DECLARATION

Prof. Dr. István Keppler (name) (student Neptun code SPL3L2)
as a consultant, I declare that I have reviewed the thesis and that I have informed the student of the requirements, legal and ethical rules for the correct handling of literary sources.

[✓ I recommend] / do not recommend[1] the final thesis to be defended in the final examination.

The thesis contains a state or official secret: yes [✓ no]*[2]

Date: 2024 year 04 month 22 day

[illegible]

insider consultant

[1] The appropriate one should be underlined.
[2] The appropriate one should be underlined.

TECHNICAL INSTITUTE
BACHELOR IN MECHANICAL ENGINEERING
Specialisation in Engineering Informatics

THESIS

worksheet

Rocha Goncalves dos Santos Lucas Daniel

for

Title of the diploma thesis: NACA0012 Airfoil Data Replication and Shape Modification Analysis in Open-Source CFD Software

---

Task reference:
The work aims to describe the reproduction of known experimental results for the profile NACA0012 in an open-source Computational Fluid Dynamics (CFD) software. Followed by the analysis, in the same software, of the modification in the airfoil geometry. The change in geometry is performed by the altering of parameters in the mathematical expression defining the NACA 4 series.

Contributing department: Department of Machine Construction

Outsider consultant: -

Insider consultant: Prof. Dr. István Keppler, Dr. János Tóth, MATE, Technical Institute

Deadline of thesis submission: 2024. April 29.

Date: 2024. February 1.

	I approve	**Received**
[illegible]	[illegible]	[illegible]
(head of deparment)	(specialist coordinator)	(student)

THESIS

Lucas Daniel Rocha Gonçalves dos Santos

2024

Hungarian University of Agriculture and Life Sciences

Szent István Campus

Tempus Foundation

Bachelor’s Degree in Mechanical Engineering

NACA0012 Airfoil Data Replication and Shape Modification Analysis in Open-Source CFD Software

Insider consultant: Prof. Dr. István Keppler

Advisor

Dr. János Tóth

Co-Advisor

Insider consultant’s Institute/department:

Technical Institute,

Department of Machine Construction

Created by: Lucas Daniel Rocha Gonçalves dos Santos

2024

Contents

1 Introduction and Objectives

The present work aims to describe the reproduction of known experimental results for the profile NACA0012 in an open-source Computational Fluid Dynamics (CFD) software, followed by the analysis, in the same software, of the modification in the airfoil geometry. The change in geometry is performed by the altering of parameters in the mathematical expression defining the NACA 4-digit series.

The purpose of the work is to be a documentation of the reproduction of experimental results, in open-source software, of airfoils and give an outline for shape modification, as a way to be an useful guideline resource for individuals in the general public interested in aeronautics and incentivize further research by those parties.

Among the shape modification problems still being studied today, is the modification of lift in an airfoil. This problem is studied today even in the development of cutting-edge technology, albeit the means to achieve it have profoundly changed since its origins.

A procedure in which a quantity is altered to meet stipulated requirements is called optimization. The optimization of aerodynamic properties is called aerodynamic shape optimization, given that the change in shape is the mean in which aerodynamic optimization can happen.

Currently, there are many state-of-the-art techniques for achieving such optimization, such as Genetic Algorithms and Gradient-Based optimization combined with the Adjoint Method (Dussauge et al., 2023).

Before the utilization of such algorithms, throughout a significant part of the history of aeronautics, aerodynamic optimization was done through direct modification of a geometry and verification if the increase or decrease of a given parameter, decreased or increased the one being optimized (Anderson, 2016).

The current work aims to utilize this more fundamental approach of a direct modification. Here the term direct modification is being used in opposition to a modification done through algorithms that lead to an automated change in shape.

Such direct modification will be done by the transformation of the NACA0012 to other known airfoils, utilizing the formula for the NACA 4-digit series. The work is not directly concerned with the optimization of lift, but rather the documentation of how this direct modification of airfoils can be done and give a general idea of how it affects the referred parameter.

Through this, the work aims to be an useful guideline resource for open-source CFD for individuals interested in aeronautics in the general public that may not have familiarity with CFD and could greatly benefit from this knowledge. A group that comes to mind is the one of hobbiests of radio-frequency airplane models.

By incentivizing those individuals to utilize open-source CFD resources, the ecosystem can be further developed by stimulating study and development of new tools, which in turn can advance the field of aeronautics as whole.

The choosing of an open-source software was also done having this objective. Similar intentions were also described by the group responsible for OpenFOAM, the open-source CFD software chosen for this work. (OpenCFD Ltd., 2024a)

2 Literature Review

2.1 Governing Equations of Fluid Mechanics

Being the focus of this work the documentation of the reproduction in CFD software of experimental data related to airfoils and subsequently change in their geometry, it comes to reason that an explanation of the physical foundations of fluid mechanics is presented, foundations from which a problem can be transferred to a computer and solved,

For that, it is essential to give an overview of the equations, the so called governing equations of fluid mechanics, that will be utilized computationally.

Anderson (1995) reiterates the fact that these equations express physics, or more specifically, each formula is built upon physical principles. He describes the three main equations of fluid mechanics as the expression of three principles, manifested under the circunstances interested by fluid mechanics, naturally, that being fluid flow. The principles and equations are the following:

- Conservation of Mass $\Rightarrow$ Continuity Equation
- Second Newton's Law $\Rightarrow$ Momentum Equations
- Conservation of Energy $\Rightarrow$ Energy Equation

The derivation of the equations of fluid mechanics, is often done by considering the fluid a continuum and in a manner where the microscopic struture can be neglected. For that end, the concept of a fluid element is utilized for the description of the flow. Those elements can be seen as the averages over a suitably large numbers of molecules. (Versteeg and Malalasekera, 2007a, p. 10)

In an analysis, the fluid elements can be studied isolated or in the context of a significantly large region, called control volume. It is important to highlight the fact, that although a control volume may be finite, the analysis is done fundamentally using tools of differential calculus, that assumes an infinitisimal fluid element. In that, the behavior of the many fluid elements inside the control volume are accounted.

From those, the finite volume and result is obtained through integral calculus. This will be discussed in more detail in the presentation of the equations.

Besides the relative size of the region, the volume can be static or in movement, accompanying the flow. The application to the volume of the three equations named above, in the static case, yields equations that are said to be in the conservation form.

As the name suggests, the name conservation form, of an equation describing a quantity, derives from the fact that in a given system in analysis, for that quantity, its conservation law holds true.

In turn, this entails that the quantity analyzed is being conserved through the element, i.e., that the input of a given quantity is equal to the output of the same quantity, and therefore that the total can only change as a result of a change in the flux through the frontiers in consideration. (LeVeque, 2002, p. 4)

This general description also manifests itself mathematically, where all the equations present the same general form, that will be presented at the end.

As mentioned, the conservation form is a consequence of the choosing of a fixed volume. Anderson (1995) describes the analysis for volumes in movement and the resulting non-conservation form in more details.

However due to the fact that the equations in conservation form are the ones currently used more broadly in CFD (Anderson, 1995; Versteeg and Malalasekera, 2007a, p. 14), and more strongly, because such form is the one used for the software chosen, this work will be mainly concerned with the equations derived from a fixed volume. Ultimately, the final set of equations will be in the context of a finite volume, for the same reason.

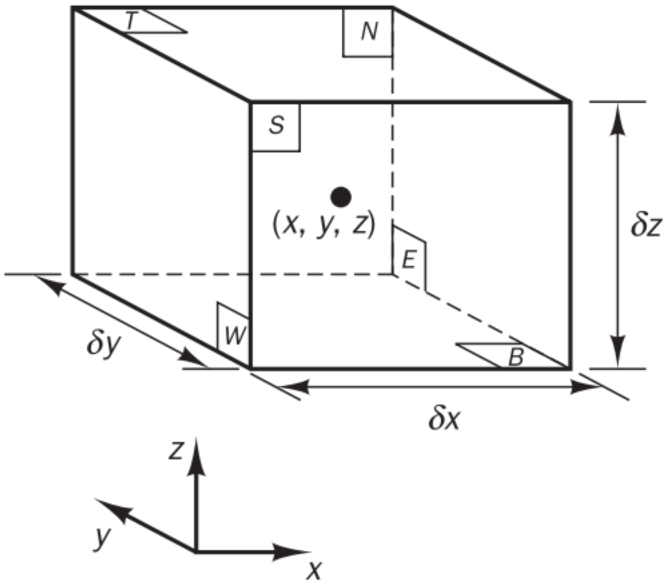
In the general context of CFD, Anderson names the importance of the conservation form as being that they provide a "*numerical and computer convenience due to the generic form*" that all share, and also being able to better model certain cases, such as some shockwave analysis.

While a finite volume consideration presents the characteristic of yielding equations in the integral form, which has the advantage, over the differential form, of not requiring mathematical continuity, and for that reason the former could be seen as a more general and fundamental than the latter. (Anderson, 1995, p. 92)

2.1.1 Outline of the Derivation of the Governing Equations

As stated, the starting point for the derivation of the equations is the consideration of an infinitesimal fluid element inside a control volume. This work will present the deriving of the continuity equation as a way to illustrate the procedure. For the Momentum Equation and Energy Equation, only a general idea of the quantities involved, the reasoning for the deduction of the equation and the final formula will be presented.

Figure 1: Fluid element
(Source: Versteeg and Malalasekera, 2007a, p. 10)



The element has its faces with the labels N, S, E, W, T, B, representing North, South, East, West, Top, and Bottom. The positive directions along the coordinate axes are specified. The center of the fluid element is situated at $(x,\ y,\ z)$.

The evaluation of the laws through the cube, give form to the equations mentioned.

For the properties involved, the following list describes the notation used:

- Density: ρ
- Pressure: p
- Temperature: T
- Vector Velocity: $\mathbf{u}$; whereas its components in x, y and z are denoted as u, v and w, respectively

All properties are functions of time and space but for short notation, this will be omitted in writing. The properties at the center of the cube will be given only by the respective symbol.

Being infinitesimal, the properties can be accurately described by the first terms in the Taylor Series, in a way that, for instance, the density on the walls W and E can be expressed,respectively by:

$$\rho - \tfrac{\partial \rho}{\partial t}\tfrac{1}{2}\delta x, \quad \rho + \tfrac{\partial \rho}{\partial t}\tfrac{1}{2}\delta x$$

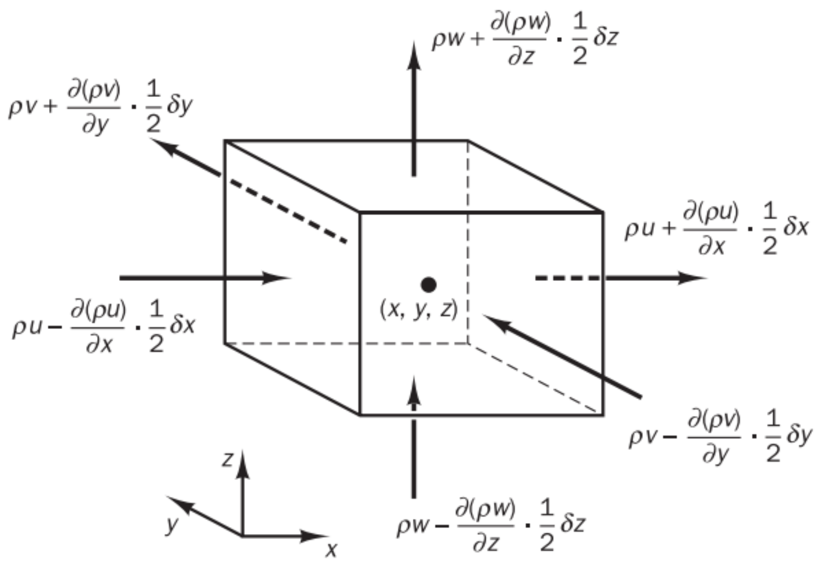
Utilizing the approach given by Versteeg and Malalasekera (2007a), the equations for momentum and energy will be derived in their non-conservative form, also, as author states, for brevity of notation and to emphasize that is based on a fluid element.

2.1.1.1 Continuity Equation

For the conservation of mass, the starting point is the evaluation of the mass balance, i.e., that the rate in which mass increases in the element is equal to the net rate of flow that enters the cube. Mathematically, the increase in mass is given by:

$$\text{Mass Increase Rate} = \frac{\partial}{\partial t}(\rho \delta x \delta y \delta z) = \frac{\partial \rho}{\partial t}\delta x \delta y \delta z \qquad (1)$$

Figure 2: Mass flows entering and leaving the fluid element
(Source: Versteeg and Malalasekera, 2007a, p. 11)



$\rho w + \frac{\partial(\rho w)}{\partial z} \cdot \frac{1}{2}\delta z$
$\rho v + \frac{\partial(\rho v)}{\partial y} \cdot \frac{1}{2}\delta y$
$\rho u + \frac{\partial(\rho u)}{\partial x} \cdot \frac{1}{2}\delta x$
$\rho u - \frac{\partial(\rho u)}{\partial x} \cdot \frac{1}{2}\delta x$
(x, y, z)
$\rho v - \frac{\partial(\rho v)}{\partial y} \cdot \frac{1}{2}\delta y$
z
y
x
$\rho w - \frac{\partial(\rho w)}{\partial z} \cdot \frac{1}{2}\delta z$

While the mass flow rate, can be obtained from analyzing the Figure 1. By definition, , the terms obtained are the product of density, area and the velocity component normal to the face.

$$
\begin{aligned}
\text{Mass Flow Rate} = &\left(\rho u - \frac{\partial(\rho u)}{\partial x}\frac{1}{2}\delta x\right)\delta y\delta z \; - \left(\rho u - \frac{\partial(\rho u)}{\partial x}\frac{1}{2}\delta x\right)\delta y\delta z \\
&+ \left(\rho v - \frac{\partial(\rho v)}{\partial y}\frac{1}{2}\delta y\right)\delta x\delta z \; - \left(\rho v - \frac{\partial(\rho v)}{\partial y}\frac{1}{2}\delta y\right)\delta x\delta z \\
&+ \left(\rho w - \frac{\partial(\rho w)}{\partial z}\frac{1}{2}\delta z\right)\delta x\delta y \; - \left(\rho w - \frac{\partial(\rho w)}{\partial z}\frac{1}{2}\delta z\right)\delta x\delta y
\end{aligned}
\tag{2}
$$

Each term accounting a face is shown in Figure 2. Positive and negative sign denote inflow and outflow, respectively.

Eq. 2 can be simplified to:

$$\text{Mass Flow Rate} = \delta x \delta y \delta z \left(\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} \right) \tag{3}$$

Equating mass increase rate and mass flow rate gives:

$$\frac{\partial \rho}{\partial t} \delta x \delta y \delta z = \delta x \delta y \delta z \left(\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} \right) \tag{4}$$

Finally, from dividing both sides by $\delta x \delta y \delta z$ and rearranging the terms to the left side, Eq. 5 is obtained:

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0 \tag{5}$$

Utilizing the divergent operator, the equation becomes:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{6}$$

Eq. 6 can be referred as the unsteady three-dimensional continuity equation at a point in a compressible fluid (Versteeg and Malalasekera, 2007a, p. 11), in its partial differential conservative form. (Anderson, 1995, p. 55)

2.1.1.2 Momentum Equation

Newton's Second Law states that the change in momentum of a particle is equal to the sum of forces acting on that particle. This will also apply to the case of interest, i.e, a fluid particle.

For each component of the momentum, the rates of increase per unit volume are given by the product of density and the so-called substantial derivative:

$$\rho \tfrac{Du}{Dt} \quad \rho \tfrac{Dv}{Dt} \quad \rho \tfrac{Dw}{Dt} \tag{7}$$

The physical meaning of the substantial derivative is that it represents the time rate of change of the quantity being differentiate, in an element, as said element moves through space (Anderson, 1995, p. 44). While mathematically is defined as:

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + \nabla \cdot (\mathbf{u}) = 0 \tag{8}$$

Once again, where **u** is the velocity vector.

Forces can be categorized between surface forces and body forces.

Surface forces directly act on the surface of the fluid element, and have only two origins:

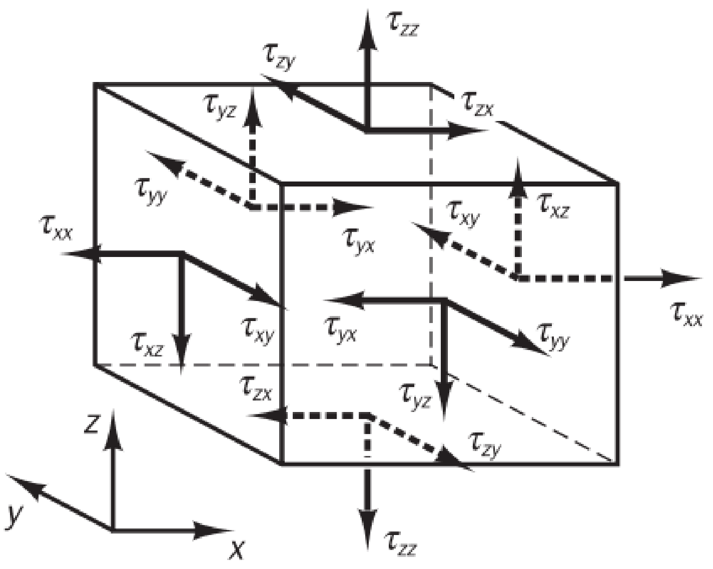
- Shear and normal stress created by surrounding fluid elements, by means of friction. These are called viscous stresses.
- Pressure distribution on the surface also created by surrounding fluid. It is worth to point out that the pressure is also an stress; a type of normal stress

Body forces act directly on the volumetric mass of the element. Examples are the electromagnetic and gravitational forces.

Similarly to the Continuity Equation, for the Momentum Equation, a type of balance is also evaluated. In this case, the force is analogous to the flow rate.

For a given surface force that the element is subjected, that force is equal to the stress being exerted on the wall times the area of the wall. Figure 3 shows the stresses in the faces of the element.

Figure 3: Stresses in the walls of an element
(Source: Versteeg and Malalasekera, 2007a, p. 14)



Body forces are usually accounted with a separate term called source term $S_M i$, where i can be one of the three directions.

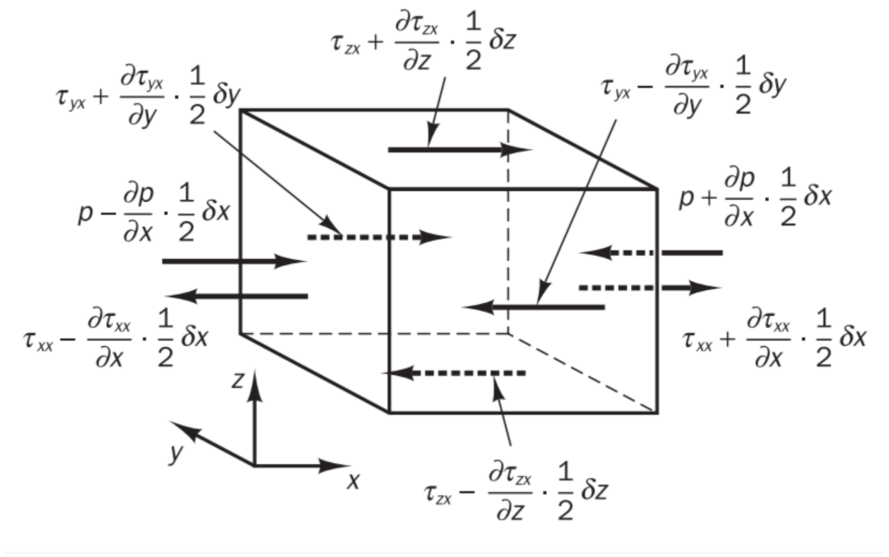
If a procedure similar to the one utilized for obtaining the flow rate in the Continuity Equation, is applied to the Figure 4, an equation for the total force in a given direction will emerge.

Therefore, in total, three equations will be derived, one for each axis, describing the total force in that axis. The total force equations can then be equated to the rate of increase of momentum, described in the beginning of this section, yielding the following set of equations:

$$\rho \frac{Du}{Dt} = \frac{\partial(-p + \tau_{xx})}{\partial x} + \frac{\partial(\tau_{yx})}{\partial y} + \frac{\partial(\tau_{zx})}{\partial z} + S_{Mx} \tag{9}$$

$$\rho \frac{Dv}{Dt} = \frac{\partial(\tau_{xy})}{\partial x} + \frac{\partial(-p + \tau_{yy})}{\partial y} + \frac{\partial(\tau_{zy})}{\partial z} + S_{My} \tag{10}$$

Figure 4: Stresses calculations for the x axis
(Source: Versteeg and Malalasekera, 2007a, p. 14)



$\tau_{zx} + \frac{\partial \tau_{zx}}{\partial z} \cdot \frac{1}{2} \delta z$
$\tau_{yx} + \frac{\partial \tau_{yx}}{\partial y} \cdot \frac{1}{2} \delta y$
$\tau_{yx} - \frac{\partial \tau_{yx}}{\partial y} \cdot \frac{1}{2} \delta y$
$p - \frac{\partial p}{\partial x} \cdot \frac{1}{2} \delta x$
$p + \frac{\partial p}{\partial x} \cdot \frac{1}{2} \delta x$
$\tau_{xx} - \frac{\partial \tau_{xx}}{\partial x} \cdot \frac{1}{2} \delta x$
$\tau_{xx} + \frac{\partial \tau_{xx}}{\partial x} \cdot \frac{1}{2} \delta x$
z
y
x
$\tau_{zx} - \frac{\partial \tau_{zx}}{\partial z} \cdot \frac{1}{2} \delta z$

$$\rho \frac{Dw}{Dt} = \frac{\partial(\tau_{xz})}{\partial x} + \frac{\partial(\tau_{yz})}{\partial y} + \frac{\partial(-p + \tau_{zz})}{\partial z} + S_{Mz} \tag{11}$$

The right side of the equations describes the total forces acting on the element, and the left-side term represents the total change in momentum. They are called, historically, the Navier-Stokes Equations.

2.1.1.3 Energy Equation

The application of the conservation of energy, also referred as the First Law of Thermodynamics, to the case of a fluid element, gives form to the following statement about the rate of change of energy in said element:

Rate of Increase of Energy of Fluid Particle	=	Net Rate of Heat Added To Fluid Particle	+	Net Rate of Work Done on Fluid Particle

In the same manner, the equation derived will be for the rate of increase of energy of fluid particle, per unit volume, expressed by:

$$\rho \frac{DE}{Dt} \tag{12}$$

The contribution of the work done on the fluid particle will originate from surface forces, also previously described on section 2.1.1.2, and are the product of the force and the velocity component in the direction of the force. The total work will be expressed by:

$$\begin{gathered} -\nabla(\rho \cdot \mathbf{u}) + \frac{\partial(\tau_{xx})}{\partial x} + \frac{\partial(\tau_{yx})}{\partial y} + \frac{\partial(\tau_{zx})}{\partial z} + \\ \frac{\partial(\tau_{xy})}{\partial x} + \frac{\partial(\tau_{yy})}{\partial y} + \frac{\partial(\tau_{zy})}{\partial z} + \\ \frac{\partial(\tau_{xz})}{\partial x} + \frac{\partial(\tau_{yz})}{\partial y} + \frac{\partial(\tau_{zz})}{\partial z} \end{gathered} \tag{13}$$

For the equation of the net flow rate of heat transfer to the fluid particle, it is conceptually useful to make an analogy with the flow rate in the Continuity Equation. An analysis of the heat transfer through the faces in Figure 5 will reveal that the two equations have a same general form, considering that the total rate of heat increase is given by the gradient of the heat flow **q** vector.

$$\text{Net Rate of Heat Added To Fluid Particle} = \nabla \cdot (\mathbf{q}) \tag{14}$$

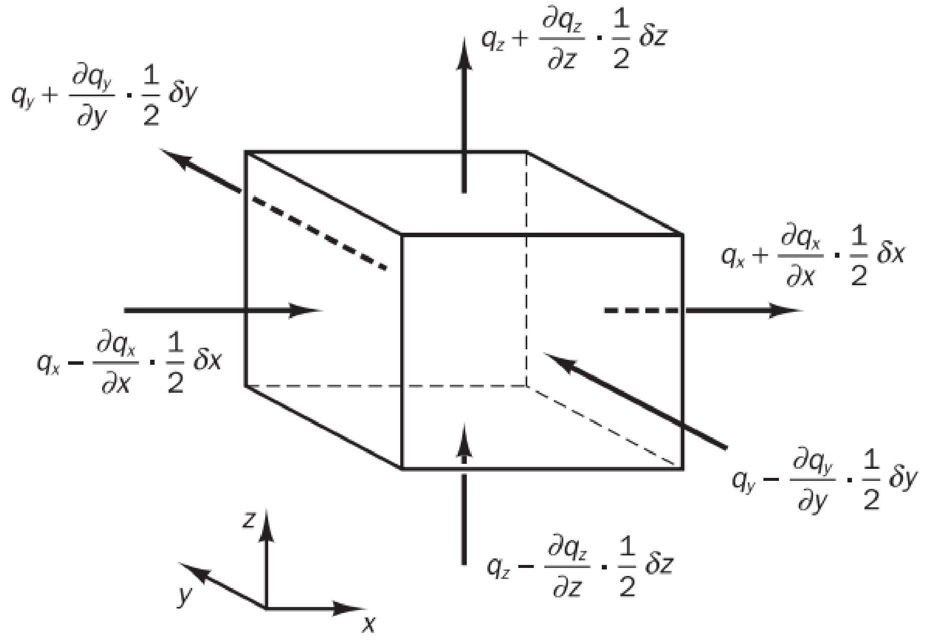
In Figure, 5, q_i denote the heat flux vector component in the direction i.

Applying the Fourier's Law of Heat conduction, the heat flux can be written in terms of the local temperature gradient:

$$q = -k\nabla(T) \tag{15}$$

Then, combining Eq. 14 and 15 yields the following equation:

Figure 5: Heat flux vector components
(Source: Versteeg and Malalasekera, 2007a, p. 18)



$q_z + \frac{\partial q_z}{\partial z} \cdot \frac{1}{2}\delta z$
$q_y + \frac{\partial q_y}{\partial y} \cdot \frac{1}{2}\delta y$
$q_x + \frac{\partial q_x}{\partial x} \cdot \frac{1}{2}\delta x$
$q_x - \frac{\partial q_x}{\partial x} \cdot \frac{1}{2}\delta x$
$q_y - \frac{\partial q_y}{\partial y} \cdot \frac{1}{2}\delta y$
$q_z - \frac{\partial q_z}{\partial z} \cdot \frac{1}{2}\delta z$
z
y
x

$$-\nabla \cdot q = \nabla \cdot \big(k\nabla(T)\big) \tag{16}$$

This equation describes the rate of heat addition to the fluid particle caused by heat conduction.

Finally, combining Eq. 13 and Eq. 16, the Energy Equation is obtained:

$$\begin{aligned}\rho\frac{DE}{Dt} = {} & -\nabla \cdot (\rho\mathbf{u}) + \\ & \Bigg[\frac{\partial(\tau_{xx})}{\partial x} + \frac{\partial(\tau_{yx})}{\partial y} + \frac{\partial(\tau_{zx})}{\partial z} + \\ & \frac{\partial(\tau_{xy})}{\partial x} + \frac{\partial(\tau_{yy})}{\partial y} + \frac{\partial(\tau_{zy})}{\partial z} + \\ & \frac{\partial(\tau_{xz})}{\partial x} + \frac{\partial(\tau_{yz})}{\partial y} + \frac{\partial(\tau_{zz})}{\partial z}\Bigg] \\ & - \nabla \cdot \big(k\nabla(T)\big) + S_E\end{aligned} \tag{17}$$

This equation assumes gravitational forces as a body force acting on the particle and doing work as the particle moves through the field. With this, the effects in potential energy are being encapsulated in the source term S_E.

2.1.2 Equations in Conservative Form for a Finite Volume

The Momentum Equation and the Energy Equation are transformed into their conservative form using the procedure described by Anderson (1995, p. 73). The procedure essencially consists in the substitution of the substantial derivative. The procedure will be exemplified utilizing the energy equation.

To reach the equation that replaces the substantial derivative, the product $\rho\frac{DE}{Dt}$ is first expanded in terms of the definition of the substantial derivative:

$$\rho\frac{DE}{Dt} = \rho\frac{\partial E}{\partial t} + \rho\mathbf{u}\cdot\nabla E = 0 \tag{18}$$

Following, $\rho\frac{\partial E}{\partial t}$ is expanded and rearranged to constitute Eq 19:

$$\rho\frac{\partial E}{\partial t} = \frac{\partial(\rho X)}{\partial t} - E\frac{\partial(\rho)}{\partial t} \tag{19}$$

Additionally, another expansion is used, now in $\rho\nabla\cdot(\mathbf{u})$. Utilizing the identity for the divergent of a product between a scalar and a vector, and rearranging the terms, Eq 20 is obtained:

$$\rho\mathbf{u}\cdot\nabla(E) = \nabla\cdot(\rho E\mathbf{u}) - E\nabla\cdot\rho(\mathbf{u}) \tag{20}$$

Replacing Eq. 19 and Eq. 20 into Eq. 18, and simplifying, will ultimately give Eq. 21, able to substitute the substantial derivative and transform the equations into their conservation form.

$$\rho\frac{DE}{Dt} = \frac{\partial(\rho E)}{\partial t} + \nabla\cdot(\rho E\mathbf{u}) \tag{21}$$

This is applied similarly to the momentum equation.

The momentum equation and the energy equation transformed to their conservation form, combined with the continuity equation, already derived in such form, yields the set of governing equations in their conservative form, given below:

Furthermore, all the equations obtained can be encapsulated within a formula for a general variable ϕ:

$$\frac{\partial \rho\phi}{\partial t} + \nabla \cdot (\rho\phi\mathbf{u}) = \nabla \cdot (\Gamma\nabla\phi) + S_\phi \tag{22}$$

Called the transport equation, its terms describe, from the left to the right the following:

- Rate of increase of ϕ of fluid element
- Net rate of flow of ϕ out of fluid element
- Rate of increase of ϕ due to diffusion
- Rate of increase of ϕ due to sources

The integration of Eq. 22 over a three-dimensional control volume (shown in Eq. 23), constitutes the base for the finite volume method (Versteeg and Malalasekera, 2007a, p. 25), the method utilized to solve the referred type of equation by the selected CFD software.

$$\int_{CV} \frac{\partial \rho\phi}{\partial t} dV + \int_{CV} \nabla \cdot (\rho\phi\mathbf{u}) dV = \int_{CV} \nabla \cdot (\Gamma\nabla\phi) dV + \int_{CV} S_\phi dV \tag{23}$$

2.2 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) essentially consists in the translation of the fundamental principles of fluid dynamics into a format in which a computer can process and obtain solutions.

Intermediately, the principles of fluid dynamics are encapsulated in the form of the known equations, i.e., those equations mathematically model a physical problem in fluid dynamics, as described in section 2.1.

The solving of these equations, before the advent of computers, were done purely using mathematical tools. This approach is called analytical. However due to the complexity of finding solutions for differential equations, specially partial differential equations, many problems could not be solved in that manner. Computers have made those solutions achievable (Anderson, 2011c).

Once a problem has been formulated in terms of the equations, computing resources use numerical methods to solve the equations using a software, therefore called CFD software, to obtain approximate solutions for the physical properties involved.

A CFD software can be divided into three main elements (Versteeg and Malalasekera, 2007b):

- Pre-processor
- Solver or Processor
- Post-processor

2.2.1 Pre-processor

The pre-processor is responsible for the construction of the geometry of relevance in a two-dimensional or three-dimensional model proper for CFD, called domain.

The domain is subdivided into smaller non-overlapping subdomains, a procedure commonly named as grid generation or meshing.

For each mesh cell, fluid's properties, prevailing conditions, governing principles of physics, initial and boundary conditions, and other associated factors are converted into mathematical constructs and formulae. More precisely, the calculations are done for each node of a cell.

The accuracy and computing cost of a mesh is directly proportional to the fineness.

Non-uniformity of the mesh is a property that can be desirable if there are regions with higher change in the quantities. In those regions, the mesh should present a more fine characteristic.

2.2.2 Processor

The processor, also called solver, is a collection of tools responsible for solving the set of equations bound to each one of cells created in the previous step.

Versteeg and Malalasekera, 2007b divides the numerical solution techniques in three main branches: finite difference, finite element and spectral methods

OpenFOAM utilizes a type of the Finite Element Method, called Finite Volume Method (FVM), and for this reason this work will only be concerned with this type of technique.

2.2.3 Post-processor

Post-processing refers to the subsequent analysis and visualization of the simulation results. Among the features in post-processing are:

- Visualization of the computational domain and grid structure: This step ensures proper mesh quality and resolution, allowing for accurate interpretation of the results.
- Vector plots: Displaying vector fields, such as velocity or pressure gradient, as arrows with varying lengths and colors to represent magnitude.
- Line and shaded contour plots: Representing scalar quantities, such as pressure, temperature, or species concentration, as lines or shaded areas with varying colors to indicate different values.
- Surface plots: Visualizing scalar quantities on 2D or 3D surfaces to better understand their spatial distribution.
- Particle tracking: Monitoring the motion of individual particles or fluid parcels to study fluid flow behavior.
- View manipulation: Interactively changing the viewpoint, translation, rotation, scaling, or hiding parts of the geometry to better analyze the results.

- Animation: Creating animations to visualize time-dependent phenomena, such as fluid flow or heat transfer.
- Custom post-processing scripts: Automating the post-processing workflow, extracting specific data, or generating custom plots.
- Report generation: Exporting results in various formats for documentation or presentation purposes.
- Comparison with experimental data: Overlaying simulation results with experimental data to assess the model's accuracy and identify discrepancies.

2.3 Finite Volume Method

The finite volume method can be divided into three main steps

In the first step, for each cell, called control volume in this method, throughout the domain, the conservation equations in their integral form are obtained.

Following, algebraic equations are calculated via the integration of the equations obtained in the first step. For this, the Gauss Theorem is often used. This step is called discretization.

At last, the algebraic equations are calculated using iterative methods.

2.4 OpenFOAM

OpenFOAM is a free and open-source continuum mechanics software developed and maintained by OpenCFD Ltd. The name is an acronym for OPEN-source Field Operation and Manipulation. The software is programmed in C++ under the license GNU GLP3, which allows the open-source characteristic and private and commercial use. The source code is available on Gitlab.

The software presents a complete set of pre-processing, processing, and post-processing tools, as well as libraries for simulating various phenomenon, including fluid flow (incompressible and compressible), turbulence, heat transfer, optimization, acoustics, chemical reactions, solid mechanics, and electromagnetism (OpenCFD Ltd., 2024a).

2.5 SimpleFOAM Solver

The SimpleFOAM consists of a incompressible, steady-state, turbulent solver within the OpenFOAM software.

The solver requires the specification of kinematic pressure, velocity, physical model for turbulence, and different solution control parameters, namely, schemes for time, gradient, divergence, laplacian, besides also requiring control parameters for the numerical methods employed in solving the pressure and velocity field. (OpenCFD Ltd., 2024b)

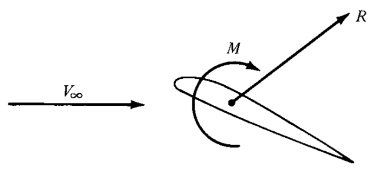
At its basis, SimpleFoam solves the continuity equation and the momentum equation, through the algorithm SIMPLE. (OpenCFD Ltd., 2024b)

2.6 Aerodynamic Forces and Moments

For every object, the aerodynamic forces and moments are the result of the pressure distribution and shear stress distribution, both acting over the body.

The integration of these distributions throughout the body, yields the resultant aerodynamic force and moment shown in Figure 6.

Figure 6: Resultant aerodynamic force and moment
(Source: Anderson, 2011a, p. 20



The notation V_∞ denotes what is called relative wind, defined as the velocity of the flow situated far ahead of a body. Concerning the flow, the term used to refer to the one in which is distant from the body is *freestream.* Due to that, V_∞ is also named freestream velocity.

From Figure 6, the force components of R can be decomposed. The consideration of a pair of components in relation to the direction of V_∞ is used to define lift and drag; respectively:

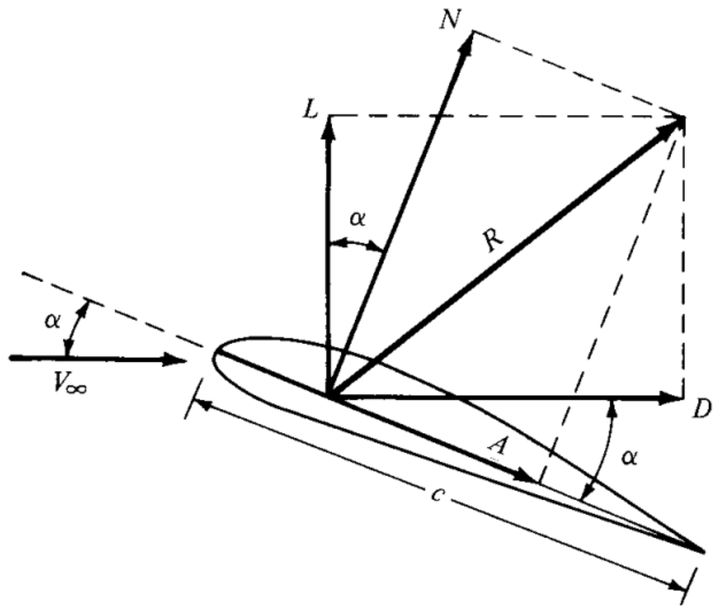
- L $\equiv$ component of R perpendicular to V_∞
- D $\equiv$ component of R parallel to V_∞

Lift and drag are represented in Figure 7. The front-most edge is called leading edge and the rear edge is called trailing edge. The line connecting the leading and trailing edges is called chord line.

The angle of attack α is described as the angle formed between the direction of the relative wind V_∞ and the chord line c.

The vectors N and A in Figure 7 denote the normal and axial forces, respectively, and are derived from considering a different direction for the decomposition of R, i.e, the direction of the chord.

Figure 7: Resultant aerodynamic force and its components
(Source: Anderson, 2011a, p. 20



The integration over the surface of the object of its pressure and stress distributions, now particularly in the direction of the defined quanties (lift, drag, etc), yields the total force of those same quantities Anderson (2011a, pp. 21–23).

The aerodynamic quantities can be synthesized into their dimensionless coefficients. Anderson (2011a, p. 23) states this is more fundamental in describing the properties. For the lift and drag this is expressed by the following definitions, respectively (Eq. 24 and 25):

$$c_L \equiv \frac{L}{q_\infty S} \tag{24}$$

$$c_D \equiv \frac{L}{q_\infty D} \tag{25}$$

Whereas, q_∞ is defined as the *dynamic pressure*, in this case, for the freestream.

$$q_\infty \equiv \frac{1}{2}\rho_\infty V_\infty^2 \tag{26}$$

And S is a reference area, that can vary for the type of body. The variation in the choosing of this area, will give a different type of coefficient, and due to this, Anderson (2011a, p. 25) highlights the importance of knowing the reference data when using a coefficient.

The use of coefficients allows for an encapsulation of the properties, derived from the shape and freestream velocity, into a quantity, and independent of other factors. For instance, to only rescale a shape must yield the same coefficients.

Other quantities can be defined, but only lift and drag will be presented, due to the scope of this work.

2.7 Airfoil and NACA Nomenclature

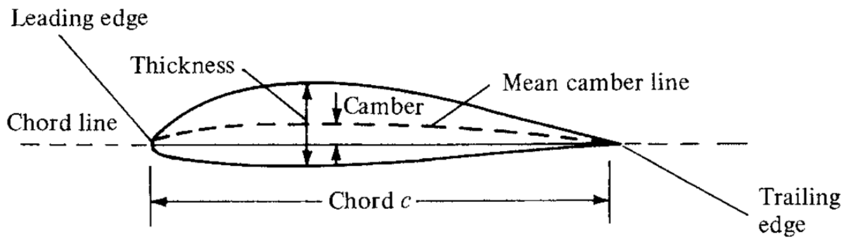
Anderson (2011b) defines an airfoil as the cross-section of a wing. The following definitions utilized here originates from the NACA naming standard.

NACA stands for National Advisory Comittee for Aeronautics, a governmental organization of the United States, created in 1915, with purpose of developing the aeronautical research (NASA, 2009).

In Figure 8 the main features of an airfoil are given. The *mean camber line* is defined as the set of points halfway between the upper an lower surfaces "*as measured perpendicular to the mean camber line itself*"(Anderson, 2011b, p. 318).

Concerning the points of the mean camber line, the front-most point is called *leading edge*, while the point located at the very rear is called *trailing edge.* The straight line connecting the leading and trailing edges is called *chord line.* The exact distance comprised by the chord line is referred simply as the *chord.* The *camber* refers to the distance between the mean camber line and the chord, measured perpendicular to the chord. Similarly, the distance measured perpendicular to the chord between the upper and lower surfaces define the thickness of the airfoil.

Figure 8: Airfoil nomenclature
(Source: Anderson, 2011b, p. 318



2.8 NACA 4-digit airfoil designation

The NACA 4-digit designation consists in a standardized airfoil naming, based on the NACA nomenclature described.

$$\text{NACA}XY(ZZ)$$

The first digit represents the percentage of the maximum camber in the airfoil. The second digit represents the horizontal coordinate of the maximum camber, given in tenths of the chord. The third and four digits represent the maximum thickness, measured in percentage of the chord. (Anderson, 2011b, p. 319)

The geometrical parameters of the standard are used in the formulas below to generate the shape (Ladson and Brooks, 1975; Moran, 1984, p. 7). Eq. 27) describes a y distance from the chord to a surface, based solely on the thickness stipulated.

$$y_t(x) = \left(\frac{t}{c}\right)\left[a_0\sqrt{\frac{x}{c}} - a_1\left(\frac{x}{c}\right) - a_2\left(\frac{x}{c}\right)^2 + a_3\left(\frac{x}{c}\right)^3 - a_4\left(\frac{x}{c}\right)^4\right] + \tag{27}$$

Where:

- t: Thickness
- c: Chord
- a_0 = 0.2969
- a_1 = -0.1260
- a_2 = -0.3516
- a_3 = 0.2843
- a_4 = -0.1036

For an symmetric airfoil, that is, one without camber, y_t corresponds exactly to the y coordinate of the upper surface.: For a symmetric airfoil, the coordinates are given by:

$$x_U = x_L = x \tag{28}$$

$$y_L = -y_t \tag{30}$$

$$y_U = y_t \tag{29}$$

For a cambered airfoil, the coordinate from 27 must be compensated. This is done by utilizing Eq. 31 in the set of equations Eq. 32 to Eq. 35

$$y_c(x) = \begin{cases} \frac{m}{p^2}(2px - x^2), & 0 \leq x \geq p \\ \frac{m}{((1-p)^2)}\big((1-2p) + 2px - x^2\big), & p \geq x \geq 1 \end{cases} \tag{31}$$

Where p denotes the maximum camber position (second digit) and m denotes the maximum camber (first digit).

The equations Eq. 32 to Eq.35 describe the x and y coordinates for a cambered airfoil.

$$x_U = x - y_t sin(\theta) \tag{32}$$

$$x_L = x + y_t sin(\theta) \tag{33}$$

$$y_U = y_c + y_t cos(\theta) \tag{34}$$

$$y_L = y_c - y_t cos(\theta) \tag{35}$$

Where:

$$\theta = arctan(\frac{dy_c}{dx}) \tag{36}$$

$$\theta = arctan(\frac{dy_c}{dx}) = \begin{cases} \frac{2pm}{p^2}(p - x), & 0 \leq x \leq p \\ \frac{2m}{(1-p)^2}(p - x), & p \leq x \leq 1 \end{cases} \tag{37}$$

2.9 NASA's Case Outline for a Turbulent Flow Through the Airfoil NACA0012

The type of problem in which this work is firstly concerned, is, in particular, the turbulent flow through an airfoil, in particular the NACA0012. The problem is further specified by the outline given by NASA under additional conditions (C. Rumsey, 2024). In addition to the outline given by NASA, the equation was slightly changed as it will be described, following the expression modification referred by Aarjav Malhotra and Kumar (2016, p. 372).

As described in the last section, the NACA 4-digit series are described by a set of formulas.

The substitution of the proper parameters for the airfoil NACA0012, yields the following formula for y_t:

$$y_t = 0.6[0.2969\sqrt{x} - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4] \tag{38}$$

In order to obtain a sharp trailing edge, the formula can be altered to the following:

$$y_t = 0.6[0.2969 * \sqrt{(}x) - 0.1260x - 0.3516x^2 + 0.2843x^3 - 0.1036x^4] \tag{39}$$

As can be noticed, the last term coefficient was altered to 0.1036 (Aarjav Malhotra and Kumar, 2016, p. 372).

The described case is then specified to be incompressible. The Reynolds number per chord is given by Re = 6 million.

An important effect in lift and drag is the effect of the farfield boundary field. Being more proeminent for high lift conditions . The minimizing of error is done by setting a significant distance between the farfield and the airfoil. In absense of this the use of a farfied vortex condition or other boundary conditions variations are recommended.

It is worth mentioning that, although experimental data for the problem is available, there is a considerable difficulty in achieving experimental 2-dimensional cases, more so for higher angles of attack, approaching stall. The literature compiling the results reflect such difficulty. Figure 9 and 10, show the variations in the distribution given by the different sources. (C. Rumsey, 2024)

At last, drag coefficients are also profoundly affected by another choice: the choosing between a tripped or untripped boundary layer for the Reynolds number range between 3 and 6 million. Comparably, lift coefficients are not affected significantly in the range in consideration. Figures 9 and 10 and illustrate this effect.

Figure 9: Lift coefficients obtained experimentally according to different sources
(Source: C. Rumsey, 2024)

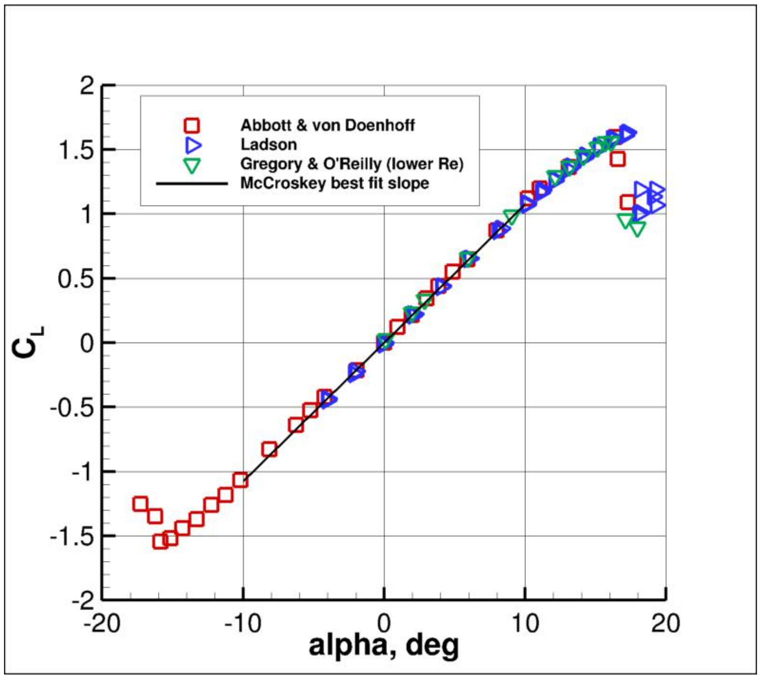
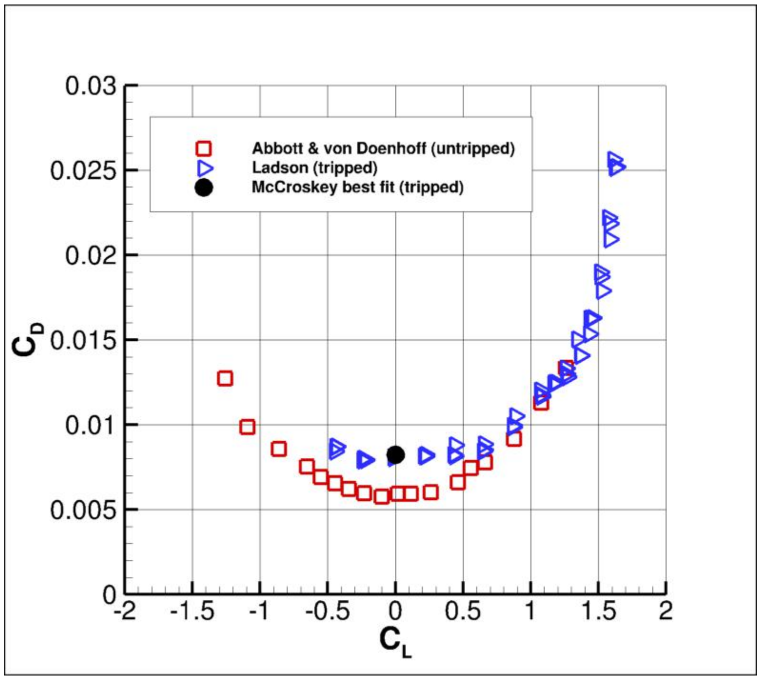


Figure 10: Drag coefficients obtained experimentally according to different sources
(Source: C. Rumsey, 2024)



3 Materials and Methodology

The choosing of open-source tools were based on the intent of allowing the free reproduction of the work by the largest number of individuals, and expects, as consequence, to encourage further contribution to the open-source CFD software ecosystem by the interested parties.

In order to provide to begineers an introduction to CFD software, and more precisely, to open-source CFD software, effort was done in order to facilitate the proper reproduction of this work.

The tools utilized can be divided in three types:

- CAD geometry software
- Mesh generation software
- CFD solver software

Open-source CFD, or more broadly, open-source in general, can be challenging compared to main commercial solutions, due to the fact that many open-source solutions do not dedicate much emphasis to Graphical User Interfaces, which most users are accustomed to in their routine interaction with software. Certain software packages may, in fact, be devoid of a Graphical User Interface, which is the case for OpenFOAM.

However, the 'open' nature of the software, allows more individuals to have an easy access to it. In addition, although it may require longer time to overcome the initial learning curve when compared to main commercial solutions, open-source software permits more versatility, not only for solving errors that may emerge, but also by allowing further development by the direct altering of the source code.

The preliminary approach attempted was done utilizing a combination of four softwares:

- FreeCAD, for geometry creation;
- Salome, for Mesh generation;

- OpenFOAM, as CFD solver
- Paraview, as CFD post-processor

This initial approach although functional, was evaluated to be slow. For a constrained change in a geometry, it was considered that a script modifying was more efficient.

Furthermore, the change to a script was also motivated by the intent to facilitate reproduction and clearer understanding of the parameters being altered, in accordance with what was described in the beginning of this section.

While a CAD geometry software allows a freer modification without further change in the base structure of how the software operates, a script for a geometry can be written explicitly in terms of known relevant parameters.

The second, and definitive, approach, consisted in utilizing a script for generating the file blockMeshDict, that specify the mesh points to be generated by OpenFOAM's blockMesh, a mesh generator that utilizes a "dictionary", a guide file, named blockMeshDict, for the creation of the grid.

The geometry creation tool and mesh points generation parts were combined into one script, named by the author as NACASMesher. This script utilizes the general formulas for the NACA 4-digit airfoils, for points creation. Based on that set of geometry points, the creation of the mesh dictionary file is conducted, also constraint by key parameters.

Summarizing, the set of tools ultimately used were:

- NACASMesher, for geometry and mesh points creation;
- blockMesh, for Mesh generation;
- OpenFOAM, as CFD solver
- Paraview, as CFD post-processor

The part for mesh points creation in the script was made by modifiying the two open-source projects created, separately, by Curiosity Fluids, 2019 and by Phan, n.d.

The software parameters are given in the list below. As it will be described in more details in the subsection for the reproduction case, some interdependencies between the variables make the final effect characteristic in the mesh difficult to predict, nonetheless, in the list is given the main role for parameters that the name requires an explanation.

- `Number_Of_Points`: Specify the number of points creating the shape of the airfoil
- `DistanceTo_Inlet`: Distance from the trailing edge to the inlet front arc.
- `DistanceTo_Outlet`: Distance from the trailing edge to the outlet vertical face
- `Angle_Of_Response`: Angle of the trailing higher refining region
- `Z_Thickness`: Thickness perpendicular to the main plane, the z-plane, of the mesh;
- `Mesh_Scale`: Scale of the mesh (multiplies all the dimensions by a factor);
- `LeadingEdge_CellSize`: Cell size at the leading edge;
- `TrailingEdge_CellSize`: Cell size at the trailing edge;
- `Middle_CellSize`: Cell size at the middle region between the leading and trailing edges;
- `SeparatingPoint_Position`: Separation point x-coordinate in chord percentage
- `BL_Thickness`: Thickness of the boundary layer
- `FirstLayer_Thickness`: Thickness of the first layer of the boundary layer
- `Expansion_Ratio`: Ratio of the size expansion in the boundary layer
- `Inlet_MaxCellSize`: Maximum longitudinal cell size at the inlet
- `Outlet_MaxCellSize`: Maximum longitudinal cell size at the outlet
- `Inlet_X_Outlet_MaxCellSize`: Vertical cell size at the 90 degrees corners between the inlet and outlet patches

- `MeshNum_OnBL`: Concentration of cells in the perpendicular direction to the surface of the airfoil and in close proximity to it, comprising the boundary layer region
- `MeshNum_OutBL`: Concentration of cells in the perpendicular direction to the surface of the airfoil, in the region immeadiatly past the boundary layer height defined by `BL_Thickness`
- `MeshNum_AtTail`: Longitudinal concentration of cells past the trailing edge (tail)
- `MeshNum_Leading`: Concentration of cells tangentially to the surface of the airfoil at the leading edge
- `MeshNum_Trailing`: Concentration of cells tangentially to the surface of the airfoil at the trailing edge

To facilitate the setting of same parameters, it was created the option to obtain the parameters from an array, besides the option to manually set each parameter.

During the reproduction of the results, the different mesh cases were inserted into a table. This table would also generate a text defining the parameter array, ready to be pasted into the script.

3.1 NACA0012 Replication Case

As stated, the first objective of the work was the reproduction of the lift coefficient c_L data for the airfoil NACA0012. This reproduction aimed to be within 10% of error. A 10% error was chosen to also facilitate reproduction and encourage mesh improvement experimentation. The error was measured against the experimental data, given by Ladson and obtanained from the NASA website C. Rumsey, 2024.

It is known that the expected theoretical result for a perfectly symmetrical airfoil in $\alpha = 0$, is that its lift should also be 0.

Due to the experimental nature of the meshers and time constraits, this work will consider as acceptable for the airfoil NACA0012, a result that yields negative values for lift at an angle of attack 0 if such values are capable of being within the referred error margin of 10% with relation to the experimental, as opposed to the theoretical prediction of $c_{L,\alpha=0} = 0$.

Therefore this work will pursue essentially a numerical curve conformation of c_L with the experimental, and by doing so it hopes to give an outline of data replication.

The case setup was done following the validation case, available at the OpenFOAM website, for the same airfoil.

The lift coefficients at 4 different angles of attack, 0, 5, 10 and 15, degrees, were analyzed.

This work was not concerned with the reproduction of the drag coefficient c_D, except for the base case for $\alpha = 0$, where c_D was used as a parameter for error control, alongside the minimization of the error for lift. This was chosen also to facilitate reproduction by begineers.

In order to achieve the reproduction of the NACA0012 data, 58 different mesh options were attempted using the script. The variation of the mesh parameters in the script were done essencially in a gray-box testing approach.

In other words, although the internal structure of the code is known, due to some more complex interdependencies between the variables, some exact results can be difficult to predict, and therefore, an educated trial and error approach was conducted, aiming for a certain mesh pattern, at the same time that the simulation result was verified to assertain if the there was improvement in the direction of the validation data. Part of the complexity arises from the fact that some constants in the original project by Phan, n.d. were allowed to be parameters in NACASMesher.

The aimed mesh pattern followed the expected for flows around airfoils, that being, a more refined grid around the surface and less refining as distance increases from it.

As stated, the different cases were tabulated. The main cases and the effect of the change of the parameters will be commented in the section of results. Ultimately, among the 58 cases, one that could keep the error within 10%, for all angles, was chosen.

3.1.1 Domain

The domain was divided in 4 groups of cells, internally called patches by OpenFoam.

The patch *inlet* consisted in the frontier region where flow enters the domain, and comprised the c shape as shown in Figure X.

The patch *outlet* comprised the back line closing the domain as shown in Figure X.

The airfoil was positioned at the center of the domain as a "hole" in the mesh, in which the walls correspond to the patch *airfoil*.

The final patch was named *sides*, corresponding to the side surfaces.

For simplification of the domain, meshing started by considering an stardard distance from the airfoil to the farfield boundary which is not as large as the one described by NASA. While being aware that this could introduce error, this was attempted before an increase in the distance, and as it will be shown, it was partially successful, being able to achieve results within the error margin stipulated.

3.1.2 Boundary Conditions

The boundary conditions are described for four quantities:

- Velocity;
- Kinematical pressure;
- Turbulent kinematic viscosity; and
- Spalart-Allmaras model modified viscosity.

For all the quantities, over the inlet and outlet, naturally, were imposed a freestream condition, i.e, that a flow originating from far away from the body and therefore not disturbed by it at its origin, is entering through these patches. This freestream condition is manifested in each of the quantities by different parameter names in OpenFOAM, however all compose the freestream condition.

Also applying for all quantities, for the sides of the domain, an empty condition was imposed, entailing a bidimensional nature to the case, where equations are not solved for the direction perpendicular to the empty patch.

The boundary conditions for velocity U are summarized in the Table 1, below:

Table 1: Boundary conditions for velocity
(Source: **website:OpenFOAM˙NACA0012**

Patch	Condition	Value $\left[\frac{m^2}{s^1}\right]$
Inlet	freestreamVelocity	U_α
Outlet	freestreamVelocity	U_α
Sides	empty	-
Aerofoil	fixedValue	(0.0, 0.0, 0.0)

The velocity in the surface of the airfoil is set to be 0, what implies what is known as no-slip condition, where no fluid slips over the surface. Although an approximation, it yields close results to reality.

The magnitude of the velocity of the frestream flow was 51.4815 m/s. As mentioned, 4 angles of atack were analyzed. From the original case given C. Rumsey, 2024, one more angle was studied; the angle of 5°. The components of the velocity for the different angles are shown in Table 2.

Table 2: Velocity vector for different angles of attack
(Source: Modified version based on **website:OpenFOAM˙NACA0012**

α	U_α
0	(51.4815, 0.00, 0.0000)
5	(51.2855, 4.4869, 0.00)
10	(50.6994, 8.9397, 0.00)
15	(49.7273, 13.3244, 0.00)

The boundary conditions for the kinematic pressure p are summarized in the Table 3.

Table 3: Boundary conditions for kinematic pressure
(Source: Modified version based on **website:OpenFOAM˙NACA0012**

Patch	Condition	Value $\left[\frac{m^2}{s^{-2}}\right]$
Inlet	freestreamPressure	0.0
Outlet	freestreamPressure	0.0
Sides	empty	-
Aerofoil	zeroGradient	-

The boundary conditions for the turbulent kinematic viscosity *nut* are summarized in the Table 4:

Table 4: Boundary conditions for kinematic viscosity
(Source: Modified version based on **website:OpenFOAM˙NACA0012**

Patch	Condition	Value $\left[\frac{m^2}{s^{-1}}\right]$
Inlet	freestreamPressure	$8.58 \cdot 10^6$
Outlet	freestreamPressure	$8.58 \cdot 10^6$
Sides	empty	-
Aerofoil	zeroGradient	- 0.0

The boundary conditions for the turbulence model Spalart-Allmaras modified viscosity *nuTilda* are summarized in the Table 5:

Table 5: Boundary conditions for Spalart-Allmaras model modified viscosity
(Source: Modified version based on **website:OpenFOAM˙NACA0012**

Patch	Condition	Value $\left[\frac{m^2}{s^{-1}}\right]$
Inlet	freestream	$3.432 \cdot 10^6 \approx 4\nu_{fluid}$
Outlet	freestream	$3.432 \cdot 10^6 \approx 4\nu_{fluid}$
Sides	empty	-
Aerofoil	fixedValue	0.0

3.2 NACA0012 Modification

The second practical objective, was to check the accuracy of the script in predicting the lift coefficient of other airfoils.

This stems from the general objective, given at the introduction, that aims for this work to give an outline of how direct shape modification of airfoils can be done and how it affects performance.

In more practical terms, the intention was to demonstrate how the increase of a given geometry parameter, such as the camber, affects lift.

The prediction had two constraints:

- To have as starting point the NACA0012 case
- To keep constant the mesh generation options;

Therefore, the changes consisted only in the modification of the geometry through the formula for the NACA 4-digit series.

Once again, the prediction for a given airfoil was considered successful in case of an error within 10%

Boundary conditions remained unchanged from the validation case.

The profiles selected were the airfoils NACA2412 and NACA0018. The results were compared against the ones given by Anderson (2011a, p. 99) for the NACA2412, and on the article by C. A. E. C. L. Rumsey (2017) for the NACA0018.

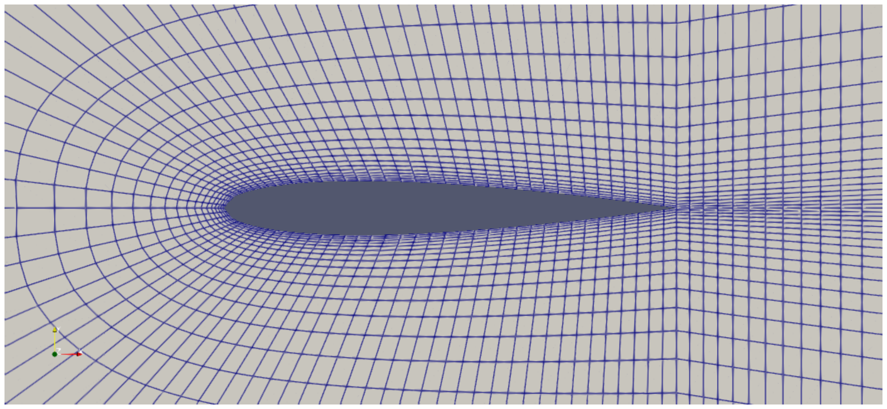
4 Results

As stated, in order to reproduce the NACA0012 lift coefficient, 58 mesh types were tested, initially, each for one simulation in OpenFOAM, for an angle of attack $\alpha = 0°$. The different mesh types will be referred by the acronym MT followed by its number (MT0, MT1, etc).

In MT0 , it was utilized the default settings given by the original project by Phan, n.d. In the original project, the number of points for the airfoil was fixed at 100.

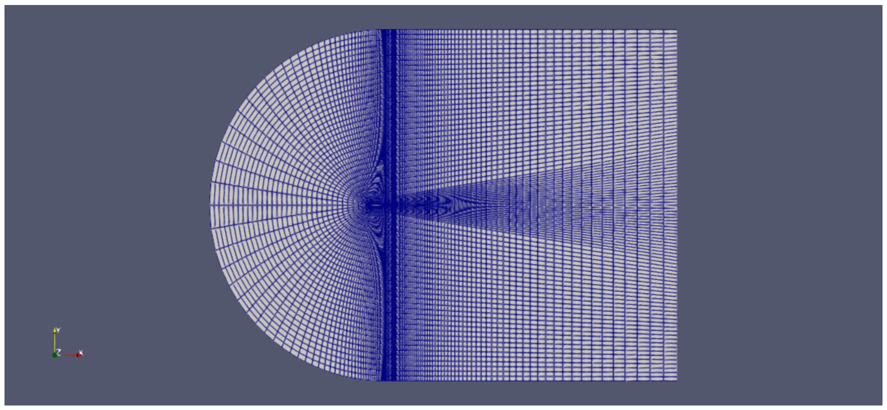
The mesh profile around the airfoil is given by Figure 11.

Figure 11: Grid around the airfoil in MT0
(Source: Own work)



A view of the entire mesh is given by Figure 12.

Figure 12: View of the entire grid in MT0
(Source: Own work)



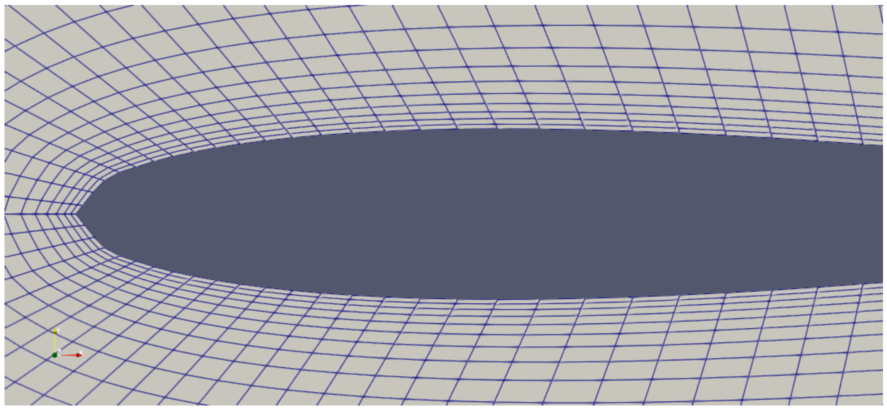
The full list of parameters are given by Table 6.

Table 6: List of parameters for MT0
(Source: Own work)

Number_Of_Points	DistanceTo_Inlet	DistanceTo_Outlet
4000	12	20
Angle_Of_Response	Z_Thickness	Mesh_Scale
0	1	1
LeadingEdge_CellSize	TrailingEdge_CellSize	Middle_CellSize
1.00E-02	3.00E-02	3.50E-02
	Separating_Point_Position	
	0.4	
BL_Thickness	First_Layer_Thickness	Expansion_Ratio
0.5	5.00E-02	1.2
Inlet_MaxCellSize	Outlet_MaxCellSize	Inlet_X_Outlet_MaxCellSize
1	1	1
MeshNum_OnBL	MeshNum_OutBL	MeshNum_AtTail
17	31	74
MeshNum_Leading	MeshNum_Trailing	
21	20	

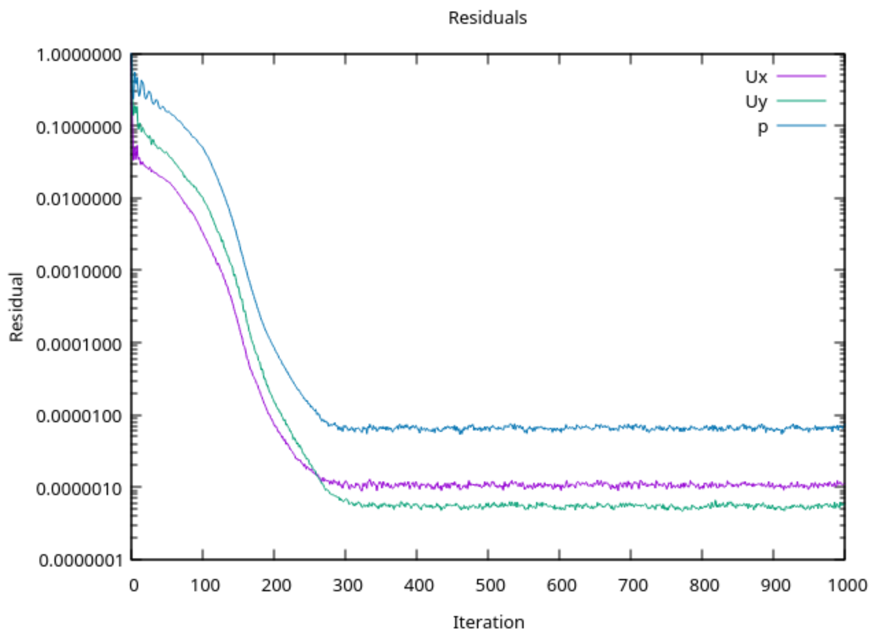
As it can be noticed in Figure 11, and in more detail, in Figure 13a the geometry feature is not fully captured by the list of 100 points, presenting sharp edges, particularlly visible on the leading edge, responsible for introducing higher errors to the model.

Figure 13: Grid closer to the airfoil's leading edge in MT0
(Source: Own work)



The convergence of the model is demonstrated by the residuals plot in Figure 14.

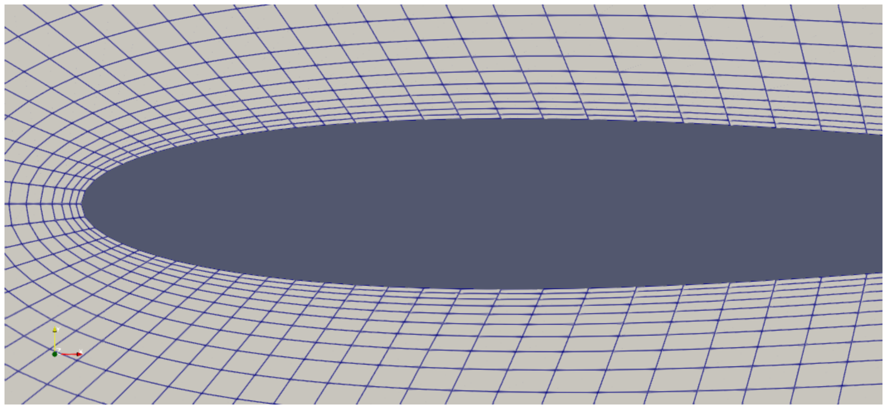
Figure 14: Example of the convergence of one of the models via residuals plot
(Source: Own work)



Due to the extensive list of parameters, the full list for following cases will be omitted.

Based on MT0, in MT1, the number of points were increased arbitrarily to 4000. The visually perceptible improvement is seem in Figure 15a .

Figure 15: Grid around the airfoil in MT1
(Source: Own work)



This change resulted in a reduction of 35.175% in the error for c_L, with relation to the original mesh type. Further increase to 10000 in the number of points, for MT2, created diminute improvement. The values and relative errors compared to the experimental data are summarized in Table 7.

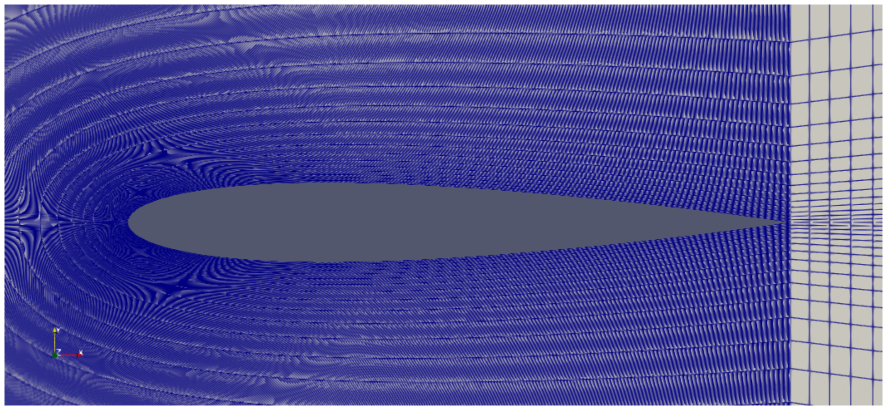
Table 7: Comparison of the effect of the number of points defining the airfoil, on MT0, MT1, MT2
(Source: Own work)

MT	Number of Points	$c_{D,\alpha=0}$	Relative Error for $c_{D,\alpha=0}$	$c_{L,\alpha=0}$	Relative Error for $c_{L,\alpha=0}$
0	100	1.28674E-02	59.021%	-1.08842E-02	50.324%
1	4000	1.26325E-02	56.118%	-8.33731E-03	15.149%
2	10000	1.26317E-02	56.108%	-8.33404E-03	15.103%

Due to the small improvement in MT2, further cases remained utilizing the 4000 points of MT1, in order to minimize computational resources.

In MT3, an increase in the number of cells tangencially to the surface of the airfoil was attempted. The parameters altered were `MeshNum_Leading` and `MeshNum_Trailing`, being multiplied by a factor of 20. The resulting mesh is shown in Figure 16a.

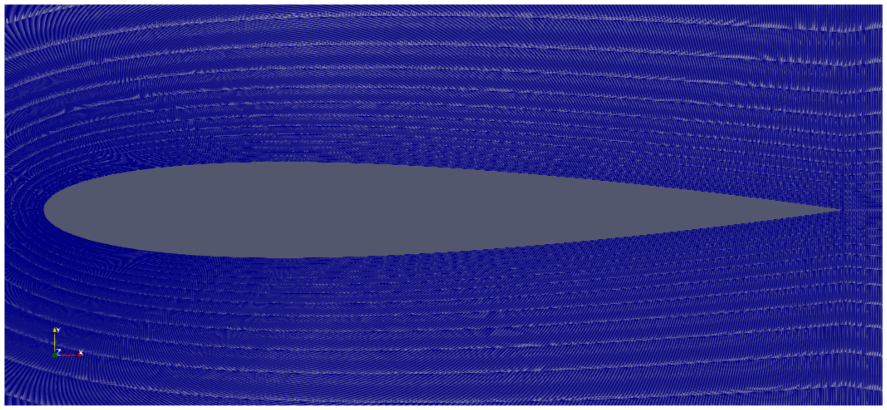
Figure 16: Grid around the airfoil in MT3
(Source: Own work)



For MT4, further increase of both parameters by a factor of 2 from MT3, represented an increase in error.

In mesh type 5, it was attempted to balance the amount of cells in the wake region, by also increasing the parameter `MeshNum_Tail` by the same accumulated factor of 40, of MT4. The error for c_L was considerably reduced. The grid profile for MT5 is shown in Figure 17.

Figure 17: Grid around the airfoil in MT5
(Source: Own work)



The drag coefficient, the parameter being considered only to minimize error, decreased its error by 1.6% from MT3 to MT4, and by 1.3% from MT4 to MT5.

Table 8 summarizes the effects on c_L of the changes in MT3-MT5.

Although a significant reduction in error was obtained for c_L in MT5, due to the persistance of a still high error, and a not significant reduction in the error for c_d, a new approach was

Table 8: Comparison of the effect of cells density tangentially to the surface, in MT3 to MT5

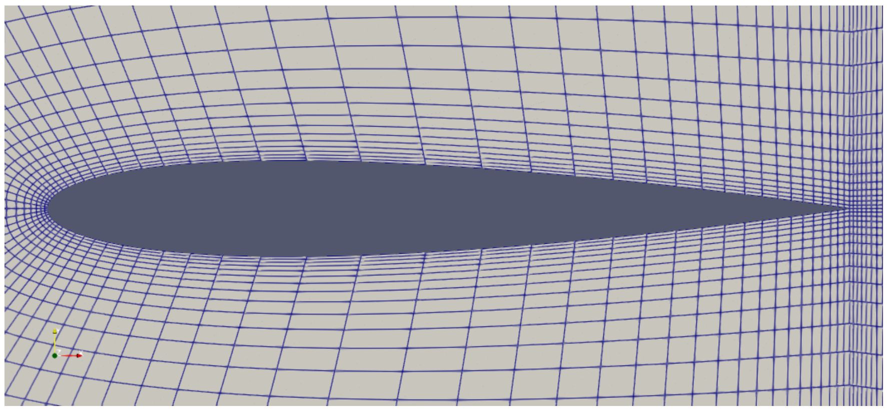
MT	`MeshNum_` `AtTail`	`MeshNum_` `Leading`	`MeshNum_` `Trailing`	$c_{L,\alpha=0}$	Relative Error for $c_{L,\alpha=0}$
3	74	420	400	-7.99366E-03	10.402%
4	74	840	800	-4.00861E-02	453.638%
5	2960	840	800	7.93604E-04	-110.961%

attempted in MT6.

All parameters were returned to the default, except for `LeadingEdge_CellSize` and `TrailingEdge_CellSize`, being both multiplied by a factor of 0.1. A significant reduction to 5.429% of error for c_L was achieved.

The change in the referred parameters have a similar effect to the `MeshNum_` types. However in this case, the refining of the mesh decreases faster as the distance increases from the trailing or leading edges, and it distorts the cells more in the direction of the edge. To have a higher refining closer to the edges is more efficient, due to the critically of these regions for the model. The grid profile is shown in Figure 18.

Figure 18: Grid around the airfoil in MT6
(Source: Own work)



Following, the parameter
`First_Layer_Thickness` was also multiplied by 0.1, in MT7, reducing the first layer to $5 \cdot 10^{-3}$. The error was further decreased to 4.966% for c_L. The error for c_D was of 30.049%.

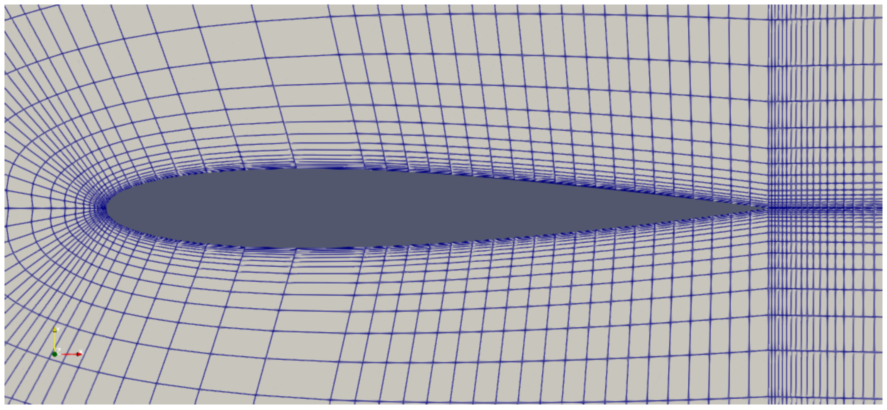
An smaller First_Layer_Thickness $= 5 \cdot 10^{-5}$, in MT8, caused an increase in the error for c_L (to 5.163%), although the error for c_D decreased (to 29.698%).

For type 9, it was checked how the software would respond to a BL_Thickness $= 0$. The mesh could not be generated.

The types 10 to 28, consisted of experimentation with the boundary layer parameters First_Layer_Thickness and Expansion_Ratio. It was attempted to increase the discretization around the surface, while keeping appropriate cell widths for upper layers. The MTs could not be solved by OpenFOAM.

Type 29 was the first one in this series investigating the boundary layer meshing, that could be run. Although consisting of only a small variation in the parameter Expansion_Ratio, from 1.2 to 1.3, this change could only operate by also reducing the parameter LeadingEdge_CellSize to $1 \cdot 10^{-5}$ and First_Layer_Thickness to $1 \cdot 10^{-4}$. The resulting mesh is shown in the Figure .

Figure 19: Grid around the airfoil in MT8
(Source: Own work)



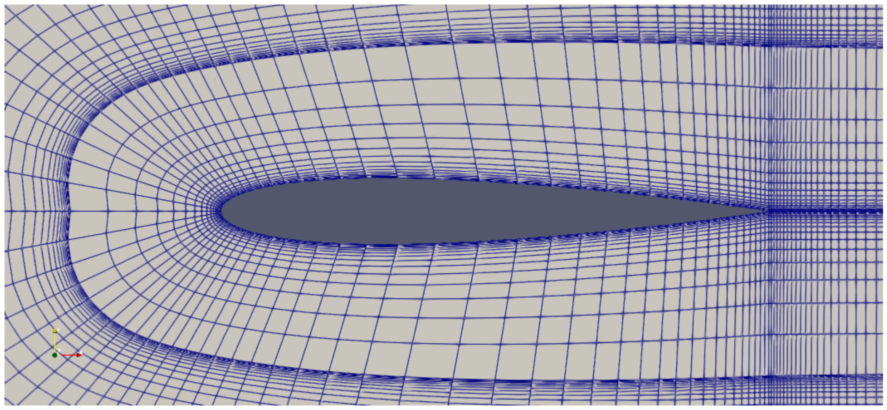
Upon conducting the tests, a highly sensitive interdependence was discerned between the variables First_Layer_Thickness and Expansion_Ratio. Similar effect was noticed in the ratio between the _CellSize variables for leading and trailing edges, and middle region of the airfoil.

The correct adjustment of these relations determined if the flow in the mesh was able to be solved by OpenFOAM.

The combination of these two parameters were further tested, culminating in a group of MT with consistently, acceptable error levels, for `First_Layer_Thickness` around 5.00E-05 and `Expansion_Ratio` around 1.3.

Among those, the type 57 was chosen due to present the smallest error for c_L. The mesh profile for MT57 is shown in Figure 20.

Figure 20: Grid around the airfoil in MT57
(Source: Own work)



The checking of the values for the other angles of attack revealed an error within 6% for all cases. The values and errors for the different angles of attack in MT57 are summarized in Table 9.

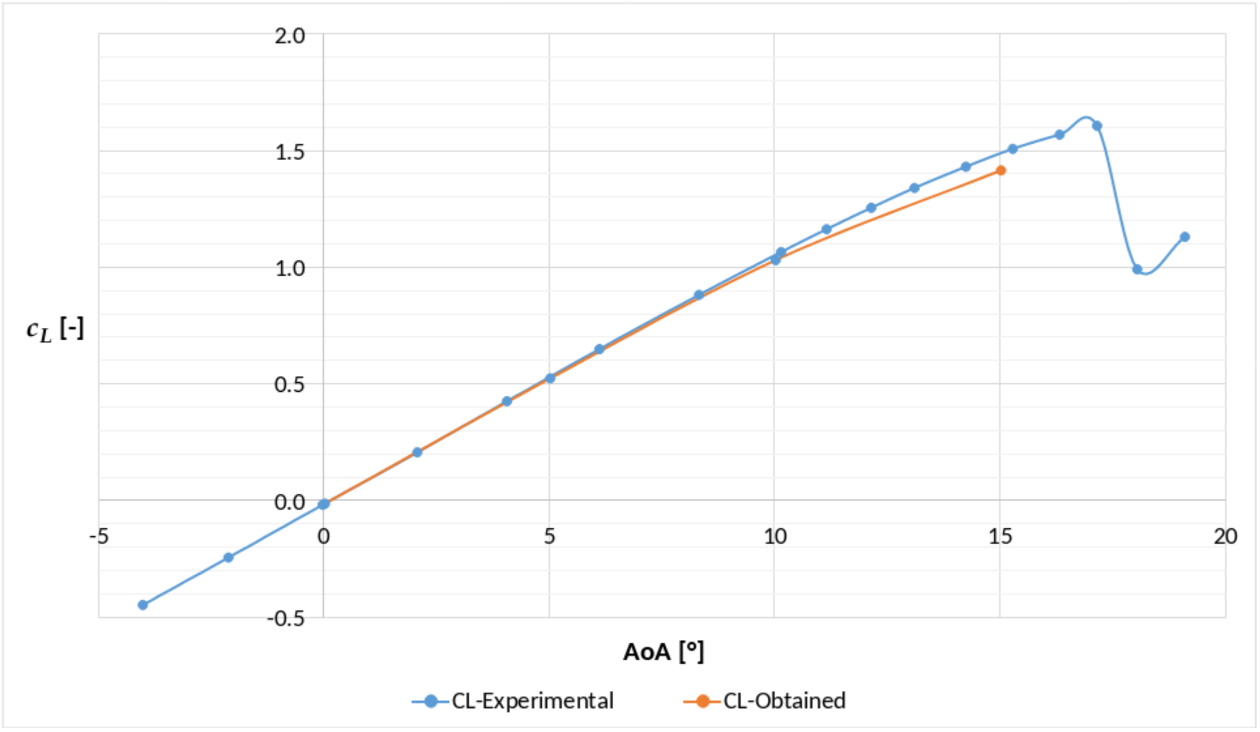
Table 9: NACA0012 lift coefficients obtained with MT57 for different angles of attack and their relative error to the experimental
(Source: Own work)

α	c_L	Relative Error for c_L	$c_{L,Experimental}$
0	-7.49210E-03	3.475%	-7.24048E-03
5	5.29512E-01	-1.216%	5.36029E-01
10	1.03626E+00	-2.111%	1.05861E+00
15	1.42081E+00	-5.232%	1.49925E+00

Figure 21, shows the distribution of the obtained c_l values, according to alpha, as well as the distribution of the experimental data, and the expected regression lines.

Due to the error values within 6%, it was confirmed that MT57 would be used as the base

Figure 21: Comparison between the experimental values and the ones obtained for c_L, for the airfoil NACA0012
(Source: Own work)



mesh type for the generation of the other NACA 4-digit series airfoils to be analyzed; the airfoils:

- NACA2412; and
- NACA0018

The velocity field in developed flow, at the maximum time of the simulation for the airfoil NACA0012, AoA $= 0$, is shown in Figure 22.

The results for the airfoil NACA2412 were able to achieve an error within 10% only for the range around $\alpha = 5°$ to $10°$. Table 10, summarizes the values obtained.

The distribution of c_l values in relation to alpha for the NACA2412 is shown in Figure 23.

The velocity field in developed flow, at the maximum time of the simulation for the airfoil NACA2412, AoA $= 0$, is shown in Figure 24.

Figure 22: Velocity field around NACA0012 for AoA = 0°
(Source: Own work)

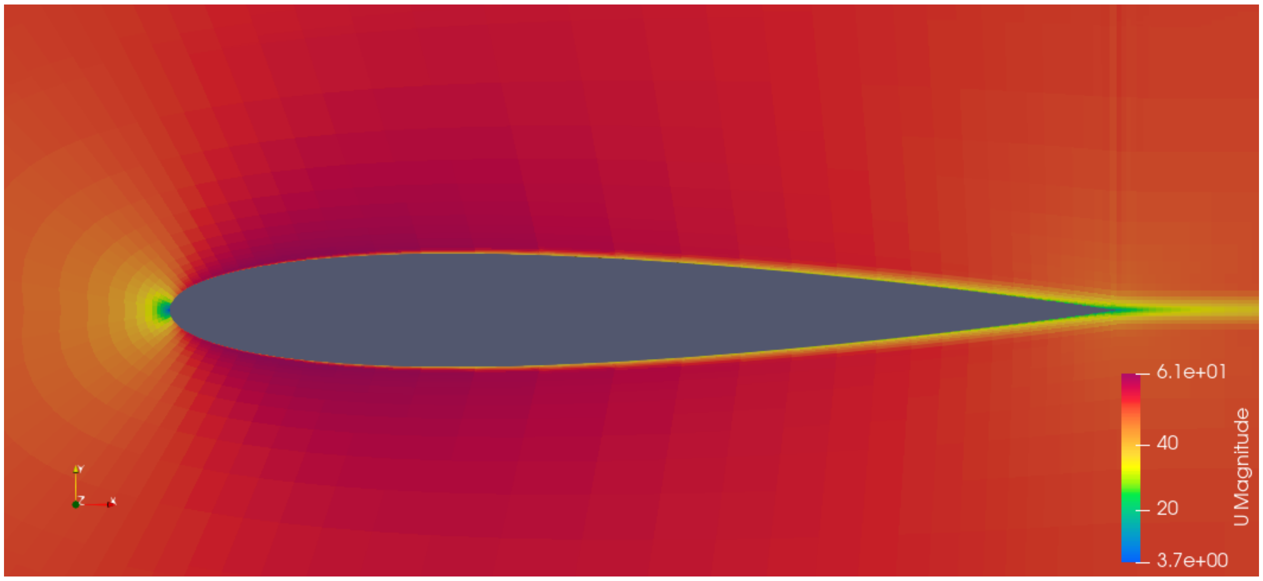


Table 10: NACA2412 lift coefficients obtained with MT57 for different angles of attack and their relative error to the experimental
(Source: Own work)

α	c_L	Relative Error for c_L	$c_{L,Experimental}$
0	-0.215	-14.071%	-0.250
5	0.748	-0.370%	0.745
10	1.244	-1.249%	1.260
15	1.421	-12.566%	1.625

The simulation of the profile NACA0018 yielded a similar error pattern, with a more pronounced deviation around the angles of 15°. The errors for the angles of 5° and 10° were of -4.967% and 1.886%, respectively.

The distribution of c_l values in relation to alpha for the NACA0018 is shown in Figure 25.

The results obtained for the airfoil NACA2412 are sufficient to demonstrate a considerable increase in lift by the presence of a camber. This fact is illustrated by Figure: 26.

Although with an error above 10% for α =0, in the range from α = 0 to α = 10, the shape of the graph is able to replicate the known result in literature of a y-translation of an airfoil

Figure 23: Comparison between the experimental values and the ones obtained for c_L, for the airfoil NACA2412
(Source: Own work)

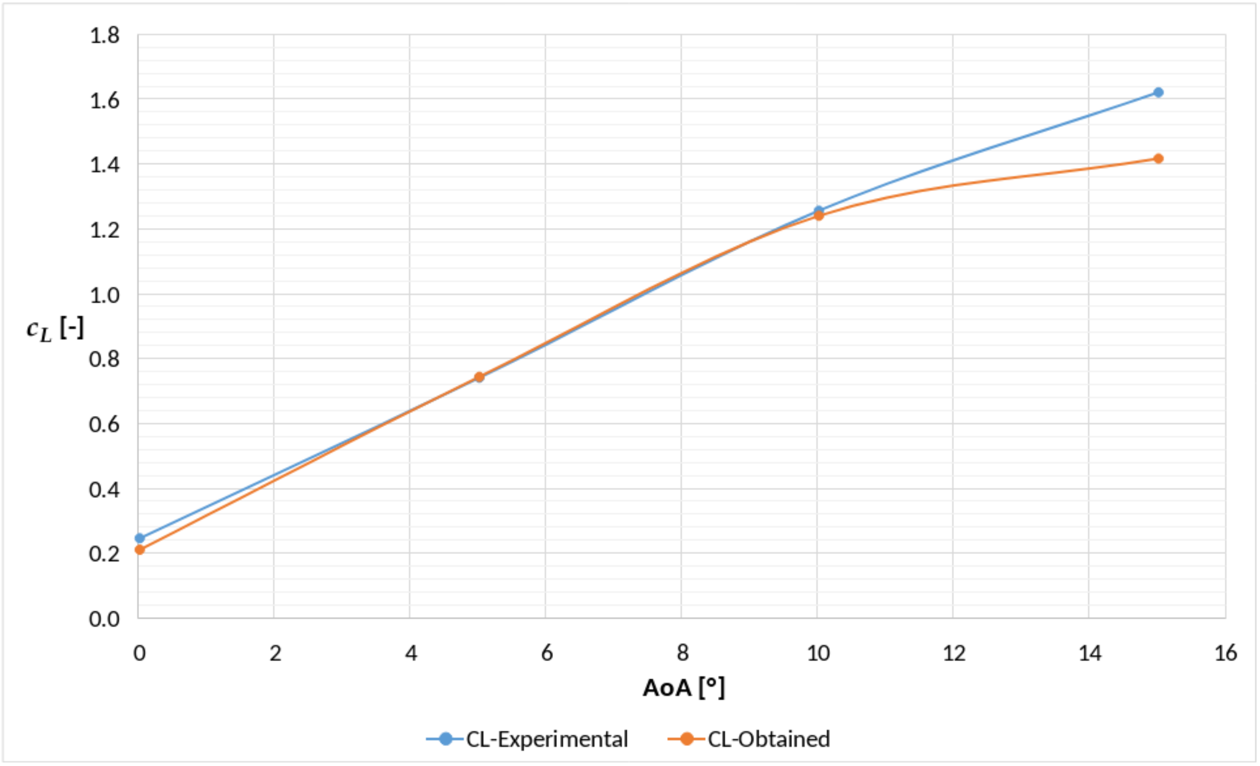
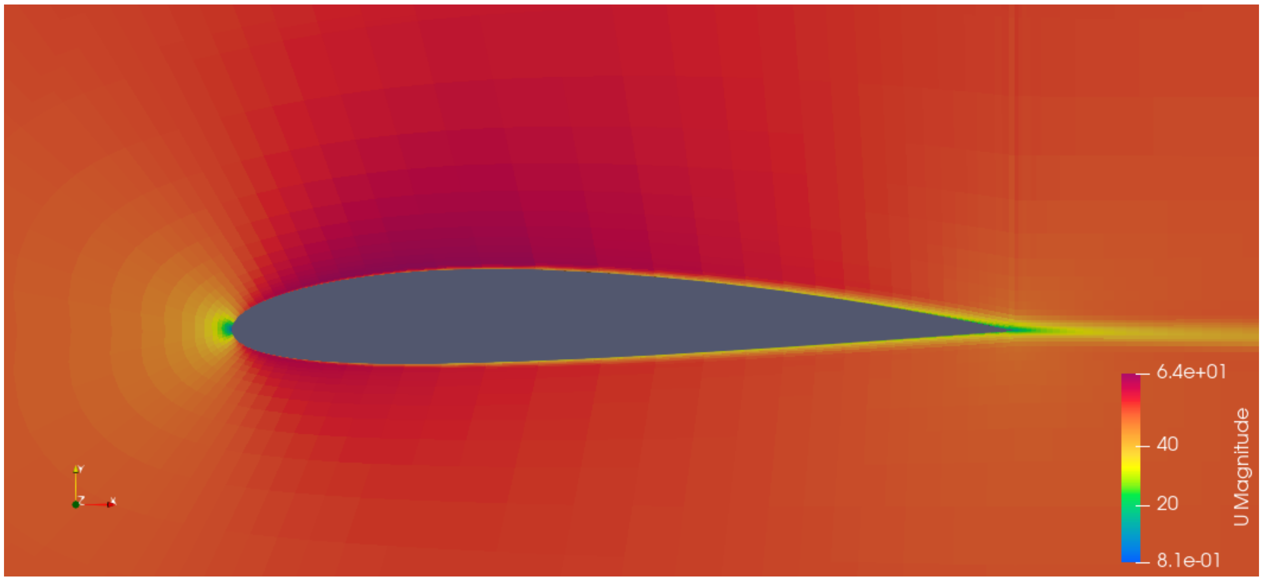
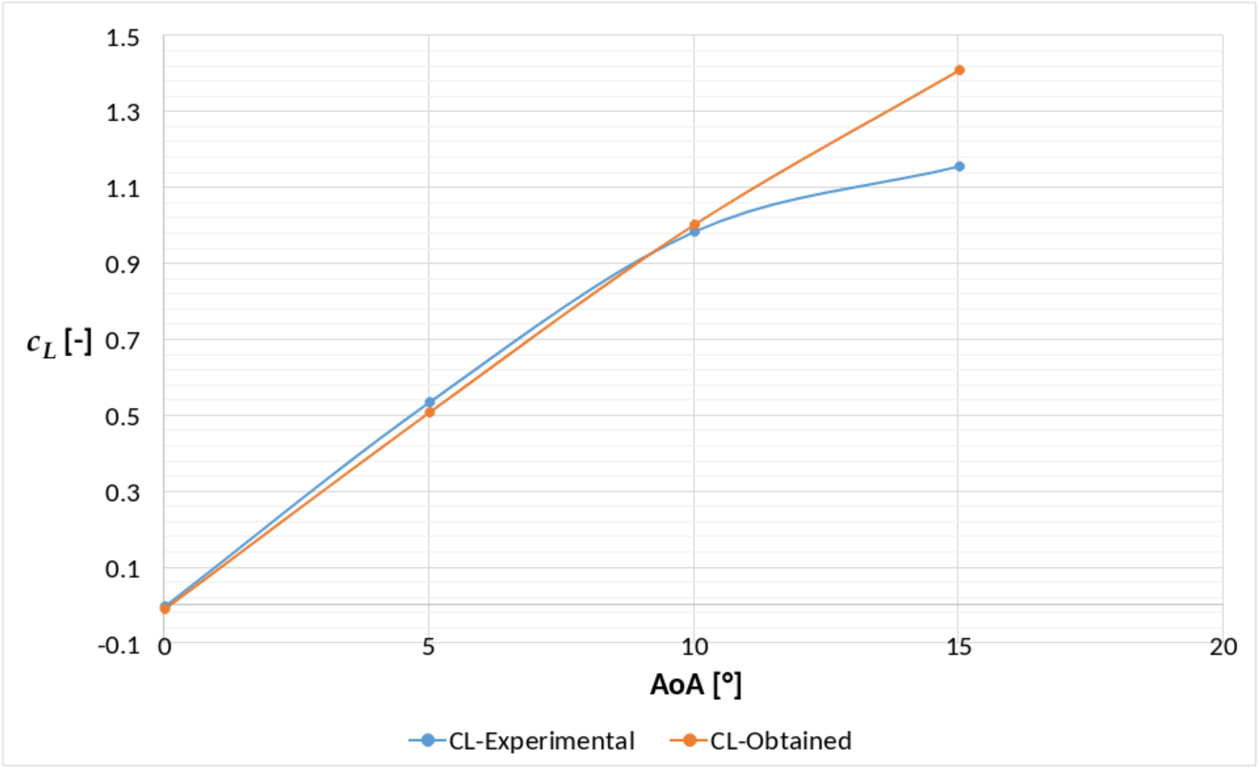


Figure 24: Velocity field around NACA2412 for AoA = 0°
(Source: Own work)



$c_L \times \alpha$ curve, for a fixed thickness, as result of an increase in the camber.

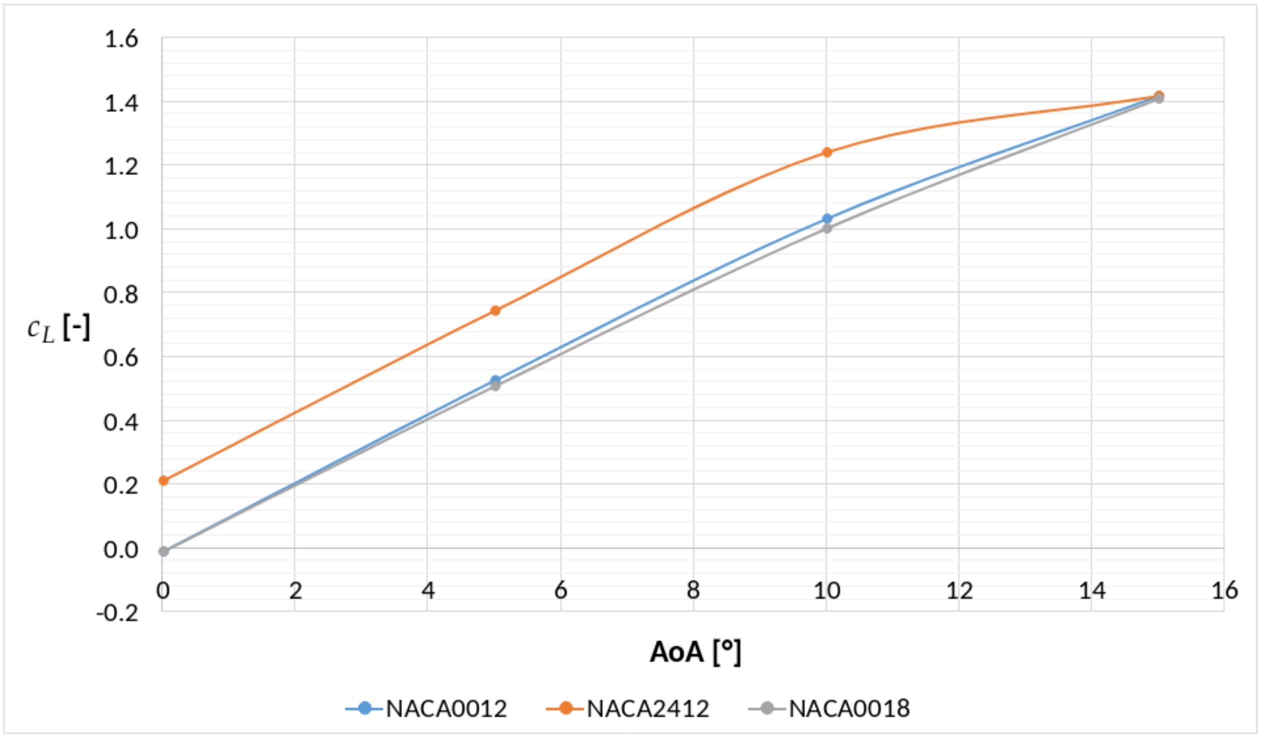
Figure 25: Comparison between the experimental values and the ones obtained for c_L, for the airfoil NACA0018



One of the possible explanations for the higher error is the effect of the farfield boundary. Although not as generally significant for lift, as for drag, the effect is more pronounced in higher lift conditions, hence the increase in error in an airfoil capable of generating higher c_L, particularly around AoA $= 15°$, where the error was more substantial.

Further investigations by increasing the distance between the airfoil and the farfield, can reveal a more proper curve, however due to time constraints, this work will limit itself to the result given.

Figure 26: Comparison between the obtained values for c_L, for the airfoils NACA0012, NACA2412 and NACA0018
(Source: Own work)



5 Conclusion

The base case, i.e., the replication of the lift coefficients for the NACA0012 was able to reach the practical objetive of maintaining an error within 10%.

Abstractly, it is believed that the procedure described is able to demonstrate the fundamentals of meshing and airfoil data replication by means of open-source tools. Therefore fulfilling the first general objective of this work

The distribution of c_l values generated a line that is a locally suitble approximation of the experimental data. An analysis for other angles can reveal the profile of this curve in more details. Due to time constraints this work limited itself to the angles of 0, 5, 10 and 15, a set able to give an outline of the curve

Altough with limited results, the same can be stated for the lift coefficient values for the airfoils NACA0018 and NACA2412 between the range of 5 to 10 degress for α

Improvements in the mesh structure through the parameters and also by modifying the script itself, can further approximate the curve to obtain more consistent error values throughout multiple airfoils, without additional manual changes. Nonetheless, the acceptable result for the range specified already alludes to the usefulness of an adaptative mesh.

The change in geometry, and its results, although properly suitable only around α 5-10, served to outline the important result known in liteture of the increase in lift caused by the increase in camber. The diminute change in the curve from the NACA0012 to the NACA0018, gives support to this demonstration of the importance of the camber in the NACA2412.

Due to the demonstration of these properties,it is believed that the second general objective of this work, of demonstrating the principles of airfoil shape modification, were therefore achieved.

Overrall, it is expected that the results obtained in this work, can serve as a basic reference for airfoil data replications and shape modification for beginners in CFD, and incentivize further development of open-source tools of the field, in particular, the script utilized in this work.

References

Aarjav Malhotra, Arpan Gupta and Pradeep Kumar (2016). “Unsteady Aerodynamics”. In: *Innovative Design and Development Practices in Aerospace and Automotive.* 1st ed. Springer Science+Business Media Singapore, pp. 369–378.

Anderson Jr., John D. (1995). “The Governing Equations of Fluid Dynamics”. In: *Computational Fluid Dynamics: The Basics with Applications.* 1st ed. McGraw-Hill. Chap. 2, pp. 37–94.

— (2011a). “Aerodynamics: Some Introductory Thoughts”. In: *Fundamentals of Aerodynamics.* 5th ed. McGraw-Hill. Chap. 1, pp. 3–98.

— (2011b). “Incompressible Flow over Airfoils”. In: *Fundamentals of Aerodynamics.* 5th ed. McGraw-Hill. Chap. 4, pp. 313–410.

— (2011c). “Invicid, Compressible Flow”. In: *Fundamentals of Aerodynamics.* 5th ed. McGraw-Hill. Chap. 2.

— (2016). “The First Aeronautical Engineers”. In: *Introduction to Flight.* 8th ed. McGraw-Hill. Chap. 1, pp. 1–52.

Curiosity Fluids (2019). *Automatic Airfoil C-Grid Generation for OpenFOAM – Rev 1.* Source: https://curiosityfluids.com/2019/04/22/automatic-airfoil-cmesh-generation-for-openfoam-rev-1/.

Dussauge, Thomas P et al. (June 2023). “A reinforcement learning approach to airfoil shape optimization”. en. In: *Sci. Rep.* 13.9753, p. 22. DOI: https://doi.org/10.1038/s41598-023-36560-z.

Ladson, Charles L. and Cuyler W. Brooks Jr. (1975). *Development of a Computer Program to Obtain Ordinates for NACA 4-Digit, 4-Digit Modified, 5-Digit and 16-Series Airfoils.* Tech. rep. Hampton, Va. 23665: NASA. Source: https://ntrs.nasa.gov/api/citations/19760003945/downloads/19760003945.pdf.

LeVeque, Randall J. (2002). “Introduction”. In: *Finite Volume Methods for Hyperbolic Problems.* 2nd ed. Cambridge: Cambridge University Press. Chap. 1, pp. 1–10.

Moran, Jack (1984). “Geometry”. In: *An Introduction To Theoretical and Computational Aerodynamics.* 1st ed. John Wiley and Sons Inc. Chap. 1, pp. 1–10.

NASA (2009). *NACA - Overview.* Source: `https://history.nasa.gov/naca/overview.html` . Last Access on: 03/15/2024.

OpenCFD Ltd. (2024a). *OpenFOAM - About.* Source: `https://doc.openfoam.com/2312/fundamentals/about/` . Last Access on: 03/02/2024.

— (2024b). *OpenFOAM - simpleFoam.* Source: `https://doc.openfoam.com/2312/tools/processing/solvers/rtm/incompressible/simpleFoam/` . Last Access on: 03/02/2024.

Phan, Thien (n.d.). *blockMesh an Airfoil.* Source: `https://www.phanquocthien.org/mesh-geometry/blockmesh/airfoil`.

Rumsey, Christopher (2024). *2DN00: 2D NACA 0012 Airfoil Validation Case.* Source: `https://turbmodels.larc.nasa.gov/naca0012_val.html` . Last Access on: 03/28/2024.

Rumsey, Christopher A. Eggert; Christopher L. (2017). *CFD Study of NACA 0018 Airfoil with Flow Control.* Tech. rep. NASA. Source: `https://ntrs.nasa.gov/api/citations/20170004500/downloads/20170004500.pdf`.

Versteeg, H. K. and W. Malalasekera (2007a). "Conservation laws of fluid motion and boundary conditions". In: *An Introduction to Computational Fluid Dynamics: The Finite Volume Method.* 2nd ed. Pearson Education Limited. Chap. 2, pp. 9–39.

— (2007b). "Conservation laws of fluid motion and boundary conditions". In: *An Introduction to Computational Fluid Dynamics: The Finite Volume Method.* 2nd ed. Pearson Education Limited. Chap. 1, pp. 1–8.

List of Figures

List of Tables

Appendix A Python script for generating the mesh - NACASMESHER

```
 #   This program is free software: you can redistribute it and/or modify
#   it under the terms of the GNU General Public License as published by
#   the Free Software Foundation, either version 3 of the License, or
#   (at your option) any later version.

#  This program is distributed in the hope that it will be useful,
#   but WITHOUT ANY WARRANTY; without even the implied warranty of
#   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#   GNU General Public License for more details.

#   You should have received a copy of the GNU General Public License
#   along with this program.  If not, see <https://www.gnu.org/licenses/>.


import numpy as np
import math
import matplotlib.pyplot as plt
import csv

#=======================================================
# NACA CURVE GENERATION FUNCTIONS
#=======================================================
def dyc_dx( x, m, p, c ):
return np.where((x>=0)&(x<=(c*p)),
((2.0 * m) / np.power(p,2)) * (p - x / c),
((2.0 * m) /np.power(1-p,2)) * (p - x / c ))

def thickness( x, t, c ):
A = 0.2969
B = -0.1260
C = -0.3516
D = 0.2843
```

```
E = -0.1036 #Closed circular trailing edge
#E = -0.1015

AxMinus1 =  A * (np.sqrt(x/c))
Bx1 =  B * (x/c)
Cx2 =  C * np.power(x/c,2)
Dx3 =  D * np.power(x/c,3)
Ex4 =  E * np.power(x/c,4)

return 5*t*c*(AxMinus1 + Bx1 + Cx2 + Dx3 + Ex4)

def camber_line( x, m, p, c ):
return np.where((x>=0)&(x<=(c*p)),
m * (x / np.power(p,2)) * (2.0 * p - (x / c)),
m * ((c - x) / np.power(1-p,2)) * (1.0 + (x / c) - 2.0 * p ))

def naca4(x, m, p, t, c=1):
if(m == 0 and p == 0):
yt = thickness(x, t, c)
return ((x, yt),
(x, -yt))
else:
dyc_dx_var = dyc_dx(x, m, p, c)
th = np.arctan(dyc_dx_var)

yt = thickness(x, t, c)
yc = camber_line(x, m, p, c)

return ((x - yt*np.sin(th), yc + yt*np.cos(th)),
(x + yt*np.sin(th), yc - yt*np.cos(th)))

#=======================================================
# CONFIGURATION
#=======================================================

#NACA CONFIGURATION
```

```
#naca0012
m = 0.02
p = 0.4
t = 0.12
c = 1.0



#MESH CONFIGURATION
vector_option = True

parameter_vector = [4000, 12, 20, 0, 1, 1, 0.001, 0.003, 0.035, 0.4, 0.4,
                                      0.00005, 1.31, 1, 1, 1, 17, 31, 74,
                                      21, 20]

if(vector_option is True):
number_of_points = parameter_vector[0]
DistanceTo_Inlet = parameter_vector[1]
DistanceTo_Outlet = parameter_vector[2]
Angle_Of_Response = parameter_vector[3]
Z_Thickness = parameter_vector[4]
Mesh_Scale = parameter_vector[5]
LeadingEdge_CellSize = parameter_vector[6]
TrailingEdge_CellSize = parameter_vector[7]
Middle_CellSize = parameter_vector[8]
SeparatingPoint_Position = parameter_vector[9]
BL_Thickness = parameter_vector[10]
FirstLayer_Thickness = parameter_vector[11]
Expansion_Ratio = parameter_vector[12]
Inlet_MaxCellSize = parameter_vector[13]
Outlet_MaxCellSize = parameter_vector[14]
Inlet_X_Outlet_MaxCellSize = parameter_vector[15]
MeshNum_OnBL = parameter_vector[16]
MeshNum_OutBL = parameter_vector[17]
MeshNum_AtTail = parameter_vector[18]
```

```
MeshNum_Leading = parameter_vector[19]
MeshNum_Trailing = parameter_vector[20]
else:
number_of_points = 4000

DistanceTo_Inlet = 12 # Distance to inlet (x chord length) - DEFAULT: 12
DistanceTo_Outlet = 20 # Distance to outlet (x chord length) - DEFAULT: 20
Angle_Of_Response = 0 # Angle of response (degree) - DEFAULT: 0
Z_Thickness = 1 # Depth in Z direction - DEFAULT: 1
Mesh_Scale = 1 # Mesh Scale - DEFAULT: 1

LeadingEdge_CellSize = 0.01 # Cell size at leading edge - DEFAULT: 0.01
TrailingEdge_CellSize = 0.03 # Cell size at trailing edge - DEFAULT: 0.03
Middle_CellSize = 0.035 # Cell size in middle - DEFAULT: 0.035
SeparatingPoint_Position = 0.4 # Separating point position (% from leading
                                         point) - DEFAULT: 0.4

BL_Thickness = 0.05 # Boundary layer thickness - DEFAULT: 0.5
FirstLayer_Thickness = 5e-7 # Fist layer thickness - DEFAULT: 0.005
Expansion_Ratio = 2.3 # Expansion ratio - DEFAULT: 1.2
#0.2; 5e-7; 2 | 1.279989e-02
#0.1; 5e-7; 2 | 1.19049999903e-02
#0.05; 5e-7; 2 | 1.171929e-02

Inlet_MaxCellSize = 2 # Inlet Max Cell Size - DEFAULT: 1
Outlet_MaxCellSize = 200 # Outlet Max Cell Size - DEFAULT: 1
Inlet_X_Outlet_MaxCellSize = 50 # Inlet_X_Outlet_MaxCellSize - DEFAULT: 1

#Secondary
MeshNum_OnBL = 17 # Parameter - DEFAULT: 17
MeshNum_OutBL = 31 # Parameter - DEFAULT: 31
MeshNum_AtTail = 220 # Parameter - DEFAULT: 74
MeshNum_Leading = 400 # Parameter - DEFAULT: 21
MeshNum_Trailing = 200 # Parameter - DEFAULT: 20
```

```
#=======================================================
# NACA POINTS CREATION FINAL FUNCTIONS
#=======================================================

number_of_points = round(number_of_points/2)
#number_of_points -= 1

x = np.linspace(1,0,number_of_points)

points = np.round(naca4(x, m, p, t, c),6)

X = np.concatenate((points[0][0],points[1][0][::-1]))
Y = np.concatenate((points[0][1],points[1][1][::-1]))

df = np.column_stack((X,Y))

pair = [0.000000, -0.000000];
#print(Coords)
index_of_midpoint = np.where((df == pair).all(axis=1))[0][0]
#print(index_of_midpoint)

#=======================================================
# MESH GENERATION FUNCTIONS AND FILE WRITTING
#=======================================================

Division_Point = SeparatingPoint_Position
ExpansionRatio_Leading = Middle_CellSize/LeadingEdge_CellSize
ExpansionRatio_Trailing = Middle_CellSize/TrailingEdge_CellSize
Inlet_ExpansionRatio_1 = TrailingEdge_CellSize/Inlet_MaxCellSize

LastLayer_Thickness = FirstLayer_Thickness*Expansion_Ratio**MeshNum_OnBL
ExpansionRatio_5 = Expansion_Ratio**MeshNum_OnBL
ExpansionRatio_6 = Inlet_MaxCellSize/LastLayer_Thickness
ExpansionRatio_7 = Outlet_MaxCellSize/TrailingEdge_CellSize
```

```
ExpansionRatio_AtOutlet = Inlet_X_Outlet_MaxCellSize*ExpansionRatio_6/
                                    Outlet_MaxCellSize*(MeshNum_OutBL+
                                    MeshNum_OnBL)/MeshNum_OutBL



ExpansionRatio1_a = LastLayer_Thickness/DistanceTo_Inlet*(ExpansionRatio_6
                                    -1)+1
ExpansionRatio1_b = math.log(ExpansionRatio_6)/math.log(ExpansionRatio1_a)

ExpansionRatio2_a = TrailingEdge_CellSize/DistanceTo_Outlet*(
                                    ExpansionRatio_7-1)+1
ExpansionRatio2_b = math.log(ExpansionRatio_7)/math.log(ExpansionRatio2_a)

ExpensionRatio3_a = LeadingEdge_CellSize/SeparatingPoint_Position*(
                                    ExpansionRatio_Leading-1)+1
ExpensionRatio3_b = math.log(ExpansionRatio_Leading)/math.log(
                                    ExpensionRatio3_a)

ExpensionRatio4_a = TrailingEdge_CellSize/(1-Middle_CellSize)*(
                                    ExpansionRatio_Trailing-1)+1
ExpensionRatio4_b = math.log(ExpansionRatio_Trailing)/math.log(
                                    ExpensionRatio4_a)

Inlet_ExpansionRatio_2 = 0.25
# Writes blockMeshDict file into system directory of current folder
f = open('system/blockMeshDict' ,'w')
f.write('/*--------------------------------*- C++
                                    -*----------------------------------*\
                                     \n')
f.write('  =========                 | \n')
f.write('  \\      /  F ield         | OpenFOAM: The Open Source CFD
                                    Toolbox \n')
f.write('   \\    /   O peration     | Website:  https://openfoam.org \n')
f.write('    \\  /    A nd           | Version:  6 \n')
f.write('     \\/     M anipulation  | \n')
```

```
f.write('
                                    \*-----------------------------------------------
                                     \n')
f.write('FoamFile \n')
f.write('{ \n')
  f.write('    version     2.0; \n')
  f.write('    format      ascii; \n')
  f.write('    class       dictionary; \n')
  f.write('    object      blockMeshDict; \n')
  f.write('} \n')
f.write('// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                                    * * * * * * // \n')
f.write('convertToMeters '+str(Mesh_Scale) +'; \n')
f.write('\n')

f.write('geometry\n')
f.write('{\n')
  f.write('}\n')
f.write('\n')

f.write('vertices\n')
f.write('(\n')
f.write('   ('+ str(0)            +   ' ' +   str(0)
                                          + ' ' +   str(0) +') \n')
f.write('   ('+ str(1)            +   ' ' +   str(0)
                                          + ' ' +   str(0) +') \n')
f.write('   ('+ str(1)            +   ' ' +   str(DistanceTo_Inlet)
                                                + ' ' + str(0) +') \n')
f.write('   ('+ str(-DistanceTo_Inlet+1)  +   ' ' +   str(0)
                                                  + ' ' + str(0) +') \n')
f.write('   ('+ str(0)            +   ' ' +   str(0)
                                          + ' ' + str(Z_Thickness) +') \n')
f.write('   ('+ str(1)            +   ' ' +   str(0)
                                          + ' ' + str(Z_Thickness) +') \n')
f.write('   ('+ str(1)            +   ' ' +   str(DistanceTo_Inlet)
                                                + ' ' + str(Z_Thickness) +'
                                    ) \n')
f.write('   ('+ str(-DistanceTo_Inlet+1)  +    ' ' + str(0)
                                                  + ' ' + str(Z_Thickness) +'
                                    ) \n')
f.write('   ('+ str(DistanceTo_Outlet+1)  +    ' ' + str(math.sin(math.pi/
                                    180*Angle_Of_Response)*(
                                    DistanceTo_Outlet+1))  + ' ' + str(0)
                                     +') \n')
f.write('   ('+ str(DistanceTo_Outlet+1)  +    ' ' + str(DistanceTo_Inlet)
                                                        + ' ' + str(0) +') \n
                                    ')
f.write('   ('+ str(DistanceTo_Outlet+1)  +    ' ' + str(math.sin(math.pi/
                                    180*Angle_Of_Response)*(
                                    DistanceTo_Outlet+1))  + ' ' + str(
                                    Z_Thickness) +') \n')
f.write('   ('+ str(DistanceTo_Outlet+1)  +    ' ' + str(DistanceTo_Inlet)
                                                        + ' ' + str(
                                    Z_Thickness) +') \n')
f.write('   ('+ str(1)              +    ' ' + str(-DistanceTo_Inlet)
                                                  + ' ' + str(0) +') \n')
f.write('   ('+ str(1)              +    ' ' + str(-DistanceTo_Inlet)
                                                  + ' ' + str(Z_Thickness) +'
                                    ) \n')
f.write('   ('+ str(DistanceTo_Outlet+1)  +    ' ' + str(-DistanceTo_Inlet)
                                                          + ' ' + str(0) +')
                                    \n')
f.write('   ('+ str(DistanceTo_Outlet+1)  +    ' ' + str(-DistanceTo_Inlet)
                                                          + ' ' + str(
                                    Z_Thickness) +') \n')
f.write('   ('+ str(1)              +    ' ' + str(0)
                                      + ' ' + str(0) +') \n')
f.write('   ('+ str(1)              +    ' ' + str(0)
                                      + ' ' + str(Z_Thickness) +') \n')


f.write('  \n')
f.write('); \n')
```

```
f.write('  \n')
f.write('\n')

f.write('blocks\n')
f.write('(\n')
f.write('  hex (0 1 2 3 4 5 6 7)  ('+str(MeshNum_Leading+MeshNum_Trailing)
                                     +' '+ str(MeshNum_OnBL+MeshNum_OutBL)
                                      + ' 1 ) //block \n')
f.write('    edgeGrading \n')
f.write('    ( \n')

f.write('    // x-direction expansion ratio \n')
f.write('    ( \n')
f.write('     (' + str(SeparatingPoint_Position) + ' ' + str(
                                     MeshNum_Leading/(MeshNum_Leading+
                                     MeshNum_Trailing)) + ' ' + str(
                                     ExpansionRatio_Leading) +' ) \n')
f.write('     (' + str(1-Division_Point) + ' ' + str(1-MeshNum_Leading/(
                                     MeshNum_Leading+MeshNum_Trailing)) +
                                     ' ' + str(1/ExpansionRatio_Leading) +
                                      ') \n')
f.write('    ) \n')
f.write('    ' + str(Inlet_ExpansionRatio_1) + ' ' + str(
                                     Inlet_ExpansionRatio_1) + ' \n')

f.write('    ( \n')
f.write('     (' + str(Division_Point) + ' ' + str(MeshNum_Leading/(
                                     MeshNum_Leading+MeshNum_Trailing)) +
                                     ' ' + str(ExpansionRatio_Leading) + '
                                     )\n')
f.write('     (' + str(1-Division_Point) + ' ' + str(1-MeshNum_Leading/(
                                     MeshNum_Leading+MeshNum_Trailing)) +
                                     ' ' + str(1/ExpansionRatio_Trailing)
                                     + ') \n')
f.write('    ) \n')
```

```
f.write('    // y-direction expansion ratio \n')
f.write('    ( \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                MeshNum_OnBL/(MeshNum_OutBL+
                                MeshNum_OnBL)) + ' ' + str(
                                ExpansionRatio_5) + ') \n')
f.write('     (' + str(1-(BL_Thickness/DistanceTo_Inlet)) + ' ' + str(1-(
                                MeshNum_OnBL/(MeshNum_OutBL+
                                MeshNum_OnBL))) + ' ' + str(
                                ExpansionRatio_6) + ') \n')
f.write('    ) \n')
f.write('    ( \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                MeshNum_OnBL/(MeshNum_OutBL+
                                MeshNum_OnBL)) + ' ' + str(
                                ExpansionRatio_5) + ') \n')
f.write('     (' + str(1-(BL_Thickness/DistanceTo_Inlet)) + ' ' + str(1-(
                                MeshNum_OnBL/(MeshNum_OutBL+
                                MeshNum_OnBL))) + ' ' + str(
                                ExpansionRatio_6) + ') \n')
f.write('    ) \n')
f.write('    ( \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                MeshNum_OnBL/(MeshNum_OutBL+
                                MeshNum_OnBL)) + ' ' + str(
                                ExpansionRatio_5) + ') \n')
f.write('     (' + str(1-BL_Thickness/DistanceTo_Inlet) + ' ' + str(1-(
                                MeshNum_OnBL/(MeshNum_OutBL+
                                MeshNum_OnBL))) + ' ' + str(
                                ExpansionRatio_6) + ') \n')
f.write('    ) \n')
f.write('    ( \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                MeshNum_OnBL/(MeshNum_OutBL+
                                MeshNum_OnBL)) + ' ' + str(
                                ExpansionRatio_5) + ') \n')
```

```
f.write('     (' + str(1-(BL_Thickness/DistanceTo_Inlet)) + ' ' + str(1-(
                                    MeshNum_OnBL/(MeshNum_OutBL+
                                    MeshNum_OnBL))) + ' ' + str(
                                    ExpansionRatio_6) + ') \n')
f.write('    ) \n')
f.write('     \n')
f.write('    // z-direction expansion ratio \n')
f.write('    1  1 1 1 \n')
f.write('    ) \n')
f.write('    \n')
f.write('    hex  (1  8 9 2 5 10  11  6)  ( '+ str(MeshNum_AtTail) + ' ' +
                                     str(MeshNum_OnBL + MeshNum_OutBL) +
                                    ' ' + '1)  //block 2 \n')
f.write('    edgeGrading \n')
f.write('    ( \n')
f.write('    // x-direction expansion ratio \n')
f.write('     ' + str(ExpansionRatio_7) + ' ' + str(ExpansionRatio_7) + '
                                    ' + str(ExpansionRatio_7) + ' ' + str
                                    (ExpansionRatio_7) +' \n')
f.write('    // y-direction expansion ratio \n')
f.write('    ( \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                    MeshNum_OnBL/(MeshNum_OutBL+
                                    MeshNum_OnBL)) + ' ' + str(
                                    ExpansionRatio_5) + ') \n')
f.write('     (' + str(1-BL_Thickness/DistanceTo_Inlet) + ' ' + str(1-(
                                    MeshNum_OnBL/(MeshNum_OutBL+
                                    MeshNum_OnBL))) + ' ' + str(
                                    ExpansionRatio_6) + ') \n')
f.write('    ) \n')
f.write('    '+  str(ExpansionRatio_AtOutlet) + '  ' + str(
                                    ExpansionRatio_AtOutlet) + ' \n')
f.write('    ( \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                    MeshNum_OnBL/(MeshNum_OutBL+
                                    MeshNum_OnBL)) + ' ' + str(
                                        ExpansionRatio_5) + ') \n')
f.write('     (' + str(1-BL_Thickness/DistanceTo_Inlet) + ' ' + str(1-(
                                        MeshNum_OnBL/(MeshNum_OutBL+
                                        MeshNum_OnBL))) + ' ' + str(
                                        ExpansionRatio_6) + ') \n')
f.write('    ) \n')
f.write('     \n')
f.write('    // z-direction expansion ratio \n')
f.write('    1  1 1 1 \n')
f.write('    ) \n')
f.write('     \n')
f.write('    hex  (3  12  16  0 7 13  17  4)  ( '+str(MeshNum_Leading+
                                        MeshNum_Trailing)+' '+ str(
                                        MeshNum_OnBL+MeshNum_OutBL) + ' 1  )
                                        //block 2 \n')
f.write('    edgeGrading \n')
f.write('    ( \n')
f.write('    // x-direction expansion ratio \n')
f.write('     ' + str(Inlet_ExpansionRatio_1) + ' \n')
f.write('    ( \n')
f.write('     (' + str(Division_Point) + ' ' + str(MeshNum_Leading/(
                                        MeshNum_Leading+MeshNum_Trailing)) +
                                        ' ' + str(ExpansionRatio_Leading) + '
                                        ) \n')
f.write('     (' + str(1-Division_Point) + ' ' + str(1-(MeshNum_Leading/(
                                        MeshNum_Leading+MeshNum_Trailing))) +
                                         ' ' + str(1/ExpansionRatio_Trailing)
                                         + ') \n')
f.write('    ) \n')
f.write('    ( \n')
f.write('     (' + str(Division_Point) + ' ' + str(MeshNum_Leading/(
                                        MeshNum_Leading+MeshNum_Trailing)) +
                                        ' ' + str(ExpansionRatio_Leading) + '
                                        ) \n')
f.write('     (' + str(1-Division_Point) + ' ' + str(1-(MeshNum_Leading/(
                                        MeshNum_Leading+MeshNum_Trailing))) +
                                              ' ' + str(1/ExpansionRatio_Trailing)
                                              + ') \n')
f.write('    ) \n')
f.write('     ' + str(Inlet_ExpansionRatio_1) + ' \n')
f.write('    // y-direction expansion ratio \n')
f.write('    ( \n')
f.write('     (' + str(1-BL_Thickness/DistanceTo_Inlet) + ' ' + str(1-(
                                     MeshNum_OnBL/(MeshNum_OutBL+
                                     MeshNum_OnBL))) + ' ' + str(1/
                                     ExpansionRatio_6) + ') \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                     MeshNum_OnBL/(MeshNum_OutBL+
                                     MeshNum_OnBL)) + ' ' + str(1/
                                     ExpansionRatio_5) + ') \n')
f.write('    ) \n')
f.write('    ( \n')
f.write('     (' + str(1-BL_Thickness/DistanceTo_Inlet) + ' ' + str(1-(
                                     MeshNum_OnBL/(MeshNum_OutBL+
                                     MeshNum_OnBL))) + ' ' + str(1/
                                     ExpansionRatio_6) + ') \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                     MeshNum_OnBL/(MeshNum_OutBL+
                                     MeshNum_OnBL)) + ' ' + str(1/
                                     ExpansionRatio_5) + ') \n')
f.write('    ) \n')
f.write('    ( \n')
f.write('     (' + str(1-BL_Thickness/DistanceTo_Inlet) + ' ' + str(1-(
                                     MeshNum_OnBL/(MeshNum_OutBL+
                                     MeshNum_OnBL))) + ' ' + str(1/
                                     ExpansionRatio_6) + ') \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                     MeshNum_OnBL/(MeshNum_OutBL+
                                     MeshNum_OnBL)) + ' ' + str(1/
                                     ExpansionRatio_5) + ') \n')
f.write('    ) \n')
f.write('    ( \n')
f.write('     (' + str(1-BL_Thickness/DistanceTo_Inlet) + ' ' + str(1-(
                                    MeshNum_OnBL/(MeshNum_OutBL+
                                    MeshNum_OnBL))) + ' ' + str(1/
                                    ExpansionRatio_6) + ') \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                    MeshNum_OnBL/(MeshNum_OutBL+
                                    MeshNum_OnBL)) + ' ' + str(1/
                                    ExpansionRatio_5) + ') \n')
f.write('    ) \n')
f.write('     \n')
f.write('    // z-direction expansion ratio \n')
f.write('    1  1 1 1 \n')
f.write('    ) \n')
f.write('     \n')
f.write('     \n')
f.write('     \n')
f.write('     \n')
f.write('     \n')
f.write('    hex  (12 14  8 16  13  15  10  17) ( ' + str(MeshNum_AtTail)
                                    + ' ' + str(MeshNum_OnBL+
                                    MeshNum_OutBL)  + ' 1) //block 4 \n')

f.write('    edgeGrading \n')
f.write('    ( \n')
f.write('    // x-direction expansion ratio \n')
f.write('     ' + str(ExpansionRatio_7) + ' ' + str(ExpansionRatio_7) + '
                                    ' + str(ExpansionRatio_7) + ' ' + str
                                    (ExpansionRatio_7) +' \n')
f.write('    // y-direction expansion ratio \n')
f.write('    ( \n')
f.write('     (' + str(1-BL_Thickness/DistanceTo_Inlet) + ' ' + str(1-(
                                    MeshNum_OnBL/(MeshNum_OutBL+
                                    MeshNum_OnBL))) + ' ' + str(1/
                                    ExpansionRatio_6) + ') \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                    MeshNum_OnBL/(MeshNum_OutBL+
                                    MeshNum_OnBL)) + ' ' + str(1/
                                    ExpansionRatio_5) + ') \n')
f.write('    ) \n')
f.write('    '+ str(1/ExpansionRatio_AtOutlet) + '   ' + str(1/
                                    ExpansionRatio_AtOutlet) + ' \n')
f.write('    ( \n')
f.write('     (' + str(1-(BL_Thickness/DistanceTo_Inlet)) + ' ' + str(1-(
                                    MeshNum_OnBL/(MeshNum_OutBL+
                                    MeshNum_OnBL))) + ' ' + str(1/
                                    ExpansionRatio_6) +') \n')
f.write('     (' + str(BL_Thickness/DistanceTo_Inlet) + ' ' + str(
                                    MeshNum_OnBL/(MeshNum_OutBL+
                                    MeshNum_OnBL)) + ' ' + str(1/
                                    ExpansionRatio_5) +  ') \n')
f.write('    ) \n')
f.write('     \n')
f.write('    // z-direction expansion ratio \n')
f.write('    1  1 1 1 \n')
f.write('    ) \n')
f.write(');\n\n')
f.write('     \n')
f.write('edges\n')
f.write('(\n')
f.write('   arc 3 2 (' + str(-DistanceTo_Inlet*math.sin(math.pi/4)+1) + '
                                    ' + str(DistanceTo_Inlet*math.sin(
                                    math.pi/4)) + '  0) \n')
f.write('   arc 7 6 (' + str(-DistanceTo_Inlet*math.sin(math.pi/4)+1) + '
                                    ' + str(DistanceTo_Inlet*math.sin(
                                    math.pi/4)) +' '+  str(Z_Thickness) +
                                    ') \n')
f.write('     \n')
f.write(' spline  1 0   \n')
f.write(' (\n')
for i in range(0, index_of_midpoint+1):
#for i in range(10):
linestring='    ('+str(X[i])+' '+str(Y[i]) +' '+ str(0)+')\n'
```

```
f.write(linestring)
#f.write('   (  0  0 0)      \n')
f.write(' )\n')
f.write(' \n')
f.write(' \n')
f.write(' spline  5 4    \n')
f.write(' (\n')
for i in range(0, index_of_midpoint+1):
#for i in range(10):
linestring='    ('+str(X[i])+' '+str(Y[i]) +' '+ str(Z_Thickness)+')\n'
f.write(linestring)
f.write(' )\n')
f.write(' \n')
f.write(' \n')
f.write(' arc 3 12  (' + str(-DistanceTo_Inlet*math.sin(math.pi/4)+1) + '
                                     ' + str(-DistanceTo_Inlet*math.sin(
                                     math.pi/4)) + '  0) \n')
f.write(' arc 7 13  (' + str(-DistanceTo_Inlet*math.sin(math.pi/4)+1) + '
                                     ' + str(-DistanceTo_Inlet*math.sin(
                                     math.pi/4)) +' '+  str(Z_Thickness) +
                                     ') \n')
f.write(' \n')
f.write(' spline  0 16   \n')
f.write(' (\n')
#for i in range(10):
for i in range(index_of_midpoint+2, len(X)):
linestring='    ('+str(X[i])+' '+str(Y[i])+' '+  str(0) + ')\n'
f.write(linestring)
f.write(' )\n')
f.write(' \n')
f.write(' \n')
f.write(' spline  4 17    \n')
f.write(' (\n')
for i in range(index_of_midpoint+2, len(X)):
#for i in range(10):
linestring='    ('+str(X[i])+' '+str(Y[i])+' '+  str(Z_Thickness) + ')\n'
```

```
f.write(linestring)

f.write(' )\n')
f.write(' \n')
f.write(' \n')
f.write(');\n')
f.write(' \n')
f.write('faces\n')
f.write('(\n')
f.write(' \n')
f.write(');\n\n')
f.write('faces\n')
f.write('(\n')
f.write(' \n')
f.write(');\n')
f.write(' \n')
f.write(' \n')
f.write('defaultPatch')
f.write('{\n')
  f.write(' name Sidesk;\n')
  f.write(' type  empty;\n')
  f.write(' \n')
  f.write('}\n')
f.write(' \n')
f.write('boundary\n')
f.write('(\n')
f.write('    inlet\n')
f.write('    {\n')
  f.write('        type patch;\n')
  f.write('        faces\n')
  f.write('        (\n')
  f.write('          (9 2 6 11)   \n')
  f.write('          (2 3 7 6  )  \n')
  f.write('          (3 12 13 7)  \n')
  f.write('          (12 15 14 13) \n')
  f.write('         );\n')
  f.write('     }\n')
f.write('    outlet\n')
f.write('    {\n')
  f.write('        type patch;\n')
  f.write('        faces\n')
  f.write('        (\n')
  f.write('          (8 9 10 11)      \n')
  f.write('          (15 8 10 14  )    \n')
  f.write('         );\n')
  f.write('     }\n')
f.write('    walls\n')
f.write('    {\n')
  f.write('        type wall;\n')
  f.write('        faces\n')
  f.write('        (\n')
  f.write('          (0 1 5 4)    \n')
  f.write('          (0 4 17 16  ) \n')
  f.write('         );\n')
  f.write('     }\n')
f.write('    interface1\n')
f.write('    {\n')
  f.write('        type patch;\n')
  f.write('        faces\n')
  f.write('        (\n')
  f.write('          (1 8 10 5)   \n')
  f.write('         );\n')
  f.write('     }\n')
f.write('    interface2\n')
f.write('    {\n')
  f.write('        type patch;\n')
  f.write('        faces\n')
  f.write('        (\n')
  f.write('          (16 17 10 8)      \n')
  f.write('         );\n')
  f.write('     }\n')
f.write(');\n')
```

```
f.write('mergePatchPairs\n')
f.write('(\n')
f.write('( interface1 interface2 )\n')
f.write(');\n')

f.close()
```