UNIVERSITY OF TWENTE.

Faculty of Engineering Technology

Italian Aerospace Research Centre

Evaluation of ice accretion on a 2D airfoil using commercial and open software.

T.T. Çakir

Internship report
December 2017

Supervisor:
prof. dr. ir. H.W.M. Hoeijmakers

Engineering Fluid Dynamics
University of Twente
Faculty CTW
Building Horst (20), N242
P.O. Box 217
7500 AE Enschede
The Netherlands

Supervisors:
dr. ir. F. Petrosino
dr. ir. G. Mingione

Fluid Mechanics Department
Centro Italiano Ricerche Aerospaziali
(CIRA)

Via Maiorise
81043 Capua (CE)
Italy

Preface

This report contains the work carried out during my internship, as part of my Master curriculum of Mechanical Engineering, at the Italian Aerospace Research Centre (CIRA).

First of all I would like to thank my supervisor at the University of Twente, Mr. Hoeijmakers and Mr. Mingione from the Italian Aerospace Research Centre, for making this internship possible and giving me this opportunity.

Next I would like to thank my supervisor at the research centre, Francesco Petrosino for giving me continuous support and helping with all the work carried out during my internship.

During my three month stay, I got the opportunity to learn many new things and therefore I am really grateful, for the tutoring and assisting by Francesco.

The skills and competences that I have acquired, will be very useful, for my future career and possibly for my Master thesis. Working among colleagues, whom themselves have their own specialization, also gave me insight in the possibilities for subjects for my Master thesis and career possibilities, regarding active research areas within Fluid Mechanics.

All in all, it was a very pleasant experience to work at CIRA, not only because of the interesting work and competences acquired, but also by the atmosphere among colleagues, and the very interesting talks we had about Italian culture and language.

Finally, I would like to thank all of my colleagues, for having a good time and giving me rides from work to home and vice versa. Especially, I would like to thank Fabrizio Morlando for taking me back and forth to work most of the time.

Summary

This report contains a comparison between commercial and open source software for ice accretion simulation. The most important goals of the research were to compare the differences between the used programs, both for fluid flow simulation as well as particle impingement simulation.
Next the phenomena of ice accretion is explained, and the work flow on how ice accretion simulations are run is explained. A short description is given about the Messinger model, which is a simple model to determine the thermodynamic balance, which in turn determines the amount of ice that accumulates on a surface.
Then a comparison method is explained that is used in subsequent chapter, to compare the iced shape of experimental results, with numerically predicted results. In Chapter 4, simulations using the potential panel method only are shown. The results for the comparisons parameter, using the different simulation options within Multi-ICE are shown. In the best case a relative difference between numerical and experimental data of 30% is achieved.
Next, using a precomputed flow field with a commercial program, an ice prediction analysis is performed. In general, the predicted ice is in better accordance with the experimental data, compared to the potential method. The reason being, that the simulation contains in general more physics and can be considered more similar to the actual physics. In the best case a relative difference of 19% between numerical and experimental data is observed.
In Chapter 6 the simulation in an open source solver is explained, the results between the commercial solver regarding the flow field, show similar flow field results, however there are slight differences. In the rest of the report, the focus lied on impingement analysis. On first sight the difference between the impingements between Multi-ICE and the open source program were expected to be due to the difference in flow field, however this was not the case.
Finally, the velocity calculation in the open source program is explained, in further detail. The corresponding files which are used are explained and looking at intermediate results, showed no wrong values for the forces acting on the particles, neither on the mass. And it is expected that the difference in particle impingements is most likely caused by the numerical integration, which is not adequately done in the open

source program.
To conclude the different programs have been compared and successful results were obtained, with similar flow fields. However, the particle impingements show significant differences, which did not allow for actual icing simulation in the open source program, rather it was attempted to understand the difference in particle impingements. It is expected the problem lies in the time stepping, which is not done in adaptive time steps in the open source code, and implementing such a feature might resolve the particle impingement calculation.

Contents

List of acronyms

LWC liquid water content

MVD median volumetric diameter

CFD computational fluid dynamics

Multi-ICE Multi-ICE version 3.3.3. (a code developed at Centro Italiano Ricerche Aerospaziali (CIRA)), which was used for the ice accretion analysis. This code allows to model the ice accretion from start to to end for a two dimensional airfoil.)

RANS Reynolds-averaged Navier-Stokes

FLUENT ANSYS FLUENT version 16.1

OpenFOAM Open Source Field Operation And Manipulation version 3.0.1

CIRA Centro Italiano Ricerche Aerospaziali

XFOIL XFOIL 6.99 (a program for the design and analysis of subsonic airfoils)

TECPLOT TECPLOT 360 2016 (a program for post-processing (computational fluid dynamics (CFD)) data)

SIMPLE Semi-Implicit Method for Pressure Linked Equations

Chapter 1

Introduction

Ice accretion is an important aspect in aircraft safety certification. Since the dangers of ice accretion can in some cases be fatal, it is of much importance, for aircraft manufacturers and designers, to prevent it. Different anti-icing systems are used currently, such as internal heating of the airfoil. However, to adequately design such systems, the ice accretion prediction is necessary and active research is being performed in this field.

1.1 Motivation

The presented work, can be considered as an evaluation of commercial and open source software, to model ice accretion. More specifically, an evaluation of Multi-ICE version 3.3.3. (a code developed at CIRA), which was used for the ice accretion analysis. This code allows to model the ice accretion from start to to end for a two dimensional airfoil.) (Multi-ICE) in combination with ANSYS FLUENT version 16.1 (FLUENT), is compared with Open Source Field Operation And Manipulation version 3.0.1 (OpenFOAM).

1.2 Framework

This work was carried out as part of an internship of the Master curriculum of Mechanical Engineering at the Engineering Fluid Mechanics department at the University of Twente. The internship was carried out at the Italian Aerospace Research Centre ()CIRA) in Capua, Italy.

1.3 Goals of the assignment

The goals of the assignment are as follows:

- The student familiarizes with the available tools at CIRA.
- Formulation of a comparison parameter that is used to compare the prediction of icing.
- An analysis of a test case is performed comparing the numerical with experimental results.
- A comparison between the commercial and open source software is performed.
- Understanding what causes the difference in impingement limits between Multi-ICE and OpenFOAM.

An additional goal that is not explicitly part of the result, however was important on getting results is:

- Data processing and translating the data from one program to the other, since the input and output files of the different programs are not directly compatible.

1.4 Report organization

The report is organized as follows. In Chapter 2, the physics of ice accretion is explained. Some definitions that will be used later on, are introduced. Then, in Chapter 3 a method to compare the performance of the numerical method, with respect to the experimental results is presented. In Chapter 4 the ice accretion using Multi-ICE only is presented, showing the different results for different simulation options that are available in Multi-ICE. Then in Chapter 5 the results for ice accretion using a precomputed flow field from FLUENT in Multi-ICE is presented. Next in Chapter 6 the simulation results using OpenFOAM are presented, however no ice accretion analysis is performed due to time limitations, rather the analysis until the particle impingements is shown. Then in Chapter 7 the calculation of particle impingements in OpenFOAM is explained in more detail. The goal was to find out what was causing the difference between the particle impingements of Multi-ICE with OpenFOAM. Finally, in Chapter 8, conclusions and recommendations are given.

Chapter 2

Ice accretion

In this chapter the physical phenomenon of ice accretion on a body is explained. Specifically, for airborne vehicles icing may occur in different stages of the flight envelope. While on the ground ice may form on an airplane, due to weather conditions and precipitation. This may also happen above the freezing point of water, because cold liquid fuel in an airplane wing, could cool down water droplets on the wing sufficiently such that they freeze [1]. After the ice forms, it influences the airplane wing performance, since the shape of the wing is off-design. Another problem that could occur is that ice that may break off and be ingested into the airplane engine, if the engine intake is behind the place where ice has formed, possibly leading to engine failure.

Another phase in which ice may form on an aircraft body is during flight. This is caused by the presence of supercooled droplets in the air. These droplets are below their freezing point, yet are liquid and thus in an unstable condition. Some of these droplets may impinge on the surface of an aircraft and freeze upon impact. Sometimes the droplets do not freeze but form a thin liquid film, which will run back over a surface and they may freeze later on, this is also referred to as run-back water. Another option is that the droplets do not freeze after running back and are carried away by aerodynamic forces.

2.1 Phase change

By processes that continually are happening in the atmosphere, such as thermal convection, liquid water droplets are transported into higher layers of the atmosphere. The droplets will cool down, as the altitude has increased, and coalesce into clouds. The presence of these droplets forms a danger for aviation, because the droplets are in an unstable situation. Although the droplets should be frozen, according to the phase diagram of water, see figure 2.1. As can be seen, the con-

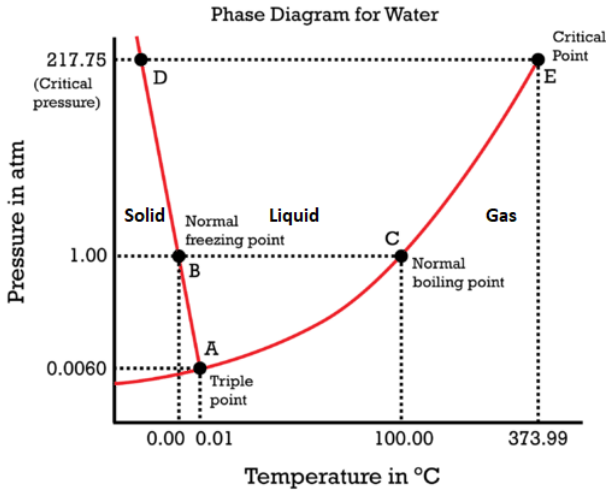


Figure 2.1: Water phase diagram.(add reference)

dition of the droplets is to the left of AB (because the pressure is lower than one atmosphere and the temperature may be below zero degrees Celsius), hence the droplets should be frozen. The reason this is not the case, is because droplets need some imitating nuclei, for example a dust particle, sand particle, a surface or even an ice particle, for the droplets to change phase. In the absence of a nuclei the droplets will remain liquid, until the droplets impinge some surface for example. Then depending on the conditions it may freeze.

Whether or not the droplets freeze depends on many factors, and in reality, a combination of freezing and run-back water is observed. The run-back water may freeze, later, but it might also be convected away by the aerodynamic forces. The parameters that influence, the amount of ice accreted, the shape and locations can be distinguished into three different categories [2].

- Atmospheric properties such as: air temperature, the ambient pressure, the amount of liquid water mass in a unit volume of air denoted by liquid water content (LWC), the relative humidity and the droplet size denoted by the median volumetric diameter (MVD).
- The conditions of the surface that are impinged such as: the surface temperature, roughness, and surface tension at the interface between air and water.
- The aerodynamics of the surrounding flow, such as: the flight velocity, angle of attack of the surface and the time that the surface is exposed to the droplets, which will be denoted by spray time.

Depending on the aforementioned parameters, in general two types of ice accretion is distinguished, namely rime ice accretion and glaze ice accretion. Rime ice occurs when the impinging droplets freeze fully upon impact, the ice looks dry and

has a white opaque colour. Glaze ice accretion, occurs when not all water freezes, but also run-back water occurs. The ice is transparent. Rime ice usually occurs at much lower temperatures, while glaze ice occurs near zero degrees Celsius, which also explains the reason why run-back water occurs. Since the droplets and ice are very near the freezing point, the heat mechanisms may just be sufficient or not for the droplets to freeze or run-back. While in glaze ice conditions, the droplets are more likely to freeze.

2.2 Ice accretion modeling

To model the phenomena of ice accretion, and specifically to model it numerically several models must be employed. First of all, the flow around the object must be modeled. This can be done using CFD. The desired end result would be the flow field, and the variables being solved for.

Then based on this flow field the particle impingement analysis is performed. For this there are several options available. Because it is a multi-phase flow, it is possible to have a certain degree of coupling between the air and water phase. In the cases considered in this report, the coupling is one-way only, with the air affecting the water, but not vice versa. However, in more complex situations a two-way coupling can be used, where the water phase also influences the air. However, this is considered computationally more expensive, and requires a coupled of way of calculating the flow field of the air.

From the above, the impingement results may follow and this gives an indication of the mass of water that hits the surface. Then based on energy and mass balances, the freezing and/or run-back of water can be computed.

Finally, for the run-back water certain models are available. For more accuracy a thin-film analysis can be performed based on CFD, but also empirical models exist that are a more simplified approach to determine the surface water.

2.3 Flow field

In this report the flow field is modeled in three ways. The first one is using a potential panel method, this is the method used inside Multi-ICE. Then the flow field is computed in FLUENT by solving the Reynolds-averaged Navier-Stokes (RANS) equations with different turbulence models. Finally, flow computations similar to those in FLUENT have been performed in OpenFOAM.

2.4 Particle impingement calculation

After calculating the flow field, the next thing is to determine the particle impingements. In Multi-ICE this is done by solving the equations of motion for the particles that are injected at a certain location. The particles are so-called Lagrangian particles, because the particles are tracked by following them in a stationary fluid flow field. A similar approach is done in the corresponding OpenFOAM solver.

The equations that solve the particle's motion are:

$$\begin{aligned} m_p \frac{d\mathbf{U_p}}{dt} &= \mathbf{F_p} \\ \frac{d\mathbf{x_p}}{dt} &= \mathbf{U_p}, \end{aligned} \tag{2.1}$$

where $\mathbf{U_p}$ and $\mathbf{x_p}$ denote the particle's velocity and position respectively, and $\mathbf{F_p}$ denotes all forces acting on the particle. Depending on how accurate it is to desired to model the forces, they can be taken into account or not. Some examples are the gravity force, buoyancy force and drag force. In reality there are many more (apparent) forces acting on the particles [2], however these are usually much smaller in order of magnitude. For this report the forces considered to act on the particle are the gravity, buoyancy and the drag force.

The equations are solved using a 4th order Runge-Kutta time integration scheme in Multi-ICE. The possible time integration schemes in OpenFOAM are the 'analytic' and Euler integration method. Some of these methods will be discussed later in the report in more detail. Then it will also become more apparent, that the differences in impingement limits are different then those predicted from Multi-ICE, which was used as reference.

2.4.1 Collection coefficient

From the impingement calculations the actual positions of the particles on the airfoil will follow. Very commonly in ice accretion studies to quantify the amount of water impinging on a surface, a so-called collection coefficient β is used for this purpose.

Aside from the x and y coordinates of the particles, a variable called curvilinear abscissa s is used and it is measured along the airfoil's contour and positive in the counterclockwise direction. In this report the convention is as follows. Starting from the bottom of the trailing edge $s = 0$ and going counter clockwise finally ending at the top of the trailing edge is where the maximum positive value s_{max} occurs.

The particles that are released at some distance from the airfoil are typically separated by some distance Δy, and they will impinge at a certain point on the

airfoil, where the distance between two impinging particles is denoted as Δs, then the collection coefficient is calculated from these two length parameters as:

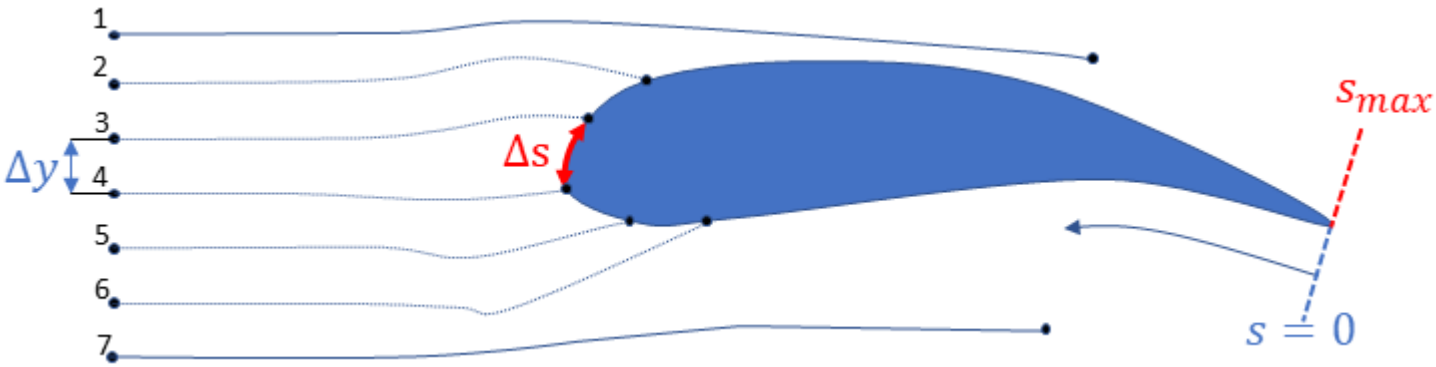


Figure 2.2: Definition of curvilinear abscissa, with $s = 0$ at the bottom of the trailing edge and positive s_{max} at the top of the trailing edge.

$$\beta = \frac{\Delta y}{\Delta s}. \tag{2.2}$$

Both at the top and bottom of the airfoil there will be limits as to where the final impingements occur. In figure 2.2, these limiting trajectories are particle number 2 and particle number 6. The collection coefficient at these limits is set to zero, so $\beta_{s_2} = \beta_{s_6} = 0$. The collection efficient will be determined in the middle between two impingements, and it is calculated according to equation 2.2

2.5 Thermodynamic model

To model the ice accretion on the airfoil, in Multi-ICE, the so-called Messinger model is used. Before being able to use the Messinger model, the flow field and the particle trajectories have to be known. The flow field is for example calculated the potential method, which is inside Multi-ICE. Then the particle impingements are calculated using a Lagrangian method approach. After calculating the flow field, and particle impingements, a thermal analysis is performed. The Messinger model provides a relatively simple, steady state approach to the icing phenomena. In short, the model is a thermal and mass balance of an unheated surface [3]. The surface is divided in multiple sections, and above or below the surface (depending on whether the top or bottom of the airfoil is considered) a control volume is made [2]. For this control volume a heat flow and mass flow balance is constructed. For the heat balance the following heat mechanisms exist:

- Heat loss by convection q_c.
- Heat loss due to sublimation of the water vapor q_s.
- Heat loss, caused by the heat absorbed when the droplets warm up after impinging on the surface, also named heat loss by fusion q_w.
- Heat gained, by the release of latent heat when the droplets impinge and solidify q_f.
- Heat gained, due to heat caused by friction inside the boundary layer q_v.
- Heat gained, due to the transfer of kinetic energy of the droplets into heat energy q_k.

In the Messinger model, the hypothesis is that these sources of heat loss and heat gain are equal, which gives a relatively simple expression than can be solved,

$$q_c + q_s + q_w = q_f + q_v + q_k, \tag{2.3}$$

which should give the equilibrium temperature of the surface. To clarify the heat flux processes, the schematic as shown by Messinger is shown in figure 2.3.

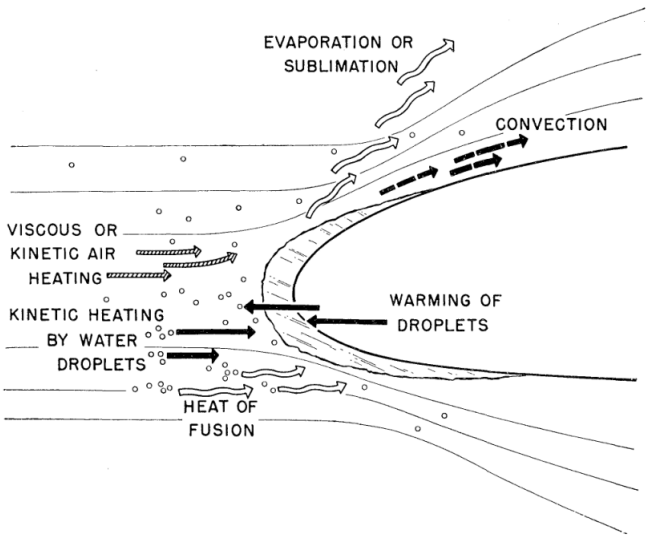


Figure 2.3: Heat transfer mechanisms of the Messinger model

.

For the mass balance a similar kind of relationship exists. The mass flows are the incoming mass due to impingement $\dot{m}_{imp}$ and incoming mass due to run-back water from upstream control volumes $\dot{m}_{in}$. The mass flows out of the control volume, are run-back water from water going out $\dot{m}_{out}$, outflow due to evaporation $\dot{m}_{ev}$ and

outflow due to freezing of water $\dot{m}_{fr}$. And the balance can be written for the mass flows as

$$\dot{m}_{out} + \dot{m}_{ev} + \dot{m}_{fr} = \dot{m}_{imp} + \dot{m}_{in} \tag{2.4}$$

Solving for both the heat flux and mass flux iteratively, and over all panels considered, the total ice accretion is computed.

In OpenFOAM there is no Messinger model present, but at CIRA a code has been implemented as an extension module for OpenFOAM. The model is a thin film layer model, and is a more computationally extensive tool. In this report, it has not been used, but only the trajectory calculation and analysis part was performed. The thin film layer model is also directly usable for the run-back water calculation, as it solves for the momentum, mass and energy equation for the film layer around the airfoil.

Chapter 3

Comparison method

In the following chapters, the different simulation methods will be presented. In Chapter 4 the results for different simulation parameter in Multi-ICE will be shown. In Chapter 5 the influence of using a precomputed flow field, which will be used as input in Multi-ICE, will be analysed.

To determine which simulation methods and parameters are the best to approximate the experimental results, aside from a qualitative a quantitative measure will be used. The qualitative measure will be given by means of figures showing the iced shape versus the clean shape of the airfoil.

The necessity to have a quantitative measure is because in many researches the predicted iced shapes are shown and analysed qualitatively only. Since, this a subjective means of analysis it is preferred to have a quantitative measure [4].

3.1 Currently used methods

Currently, there are different method to measure the iced shape. Some of these are described in [4]. The measured quantities in the mentioned article are the horn length, horn angle, stagnation point thickness, ice shape cross section and icing limits. Wright also addresses certain issues on finding the maximum thickness, which corresponds to the horn length. One method is to find the minimum distance between the currently being evaluated point on the airfoil, and from this the ice thickness is found. The second method is to simply find the length of the distance from the airfoil to the iced shape, by simply considering the unit normal to the point being evaluated. The first method appears to be better, especially if more complex shapes are formed during icing, for example when a clear horn has formed. However, for more simple shapes, the unit normal still provides good results. Since the icing case considered in this report, has a relatively simple shape, the second method was used in determining the thickness along the airfoil.

A different approach has been established by Ruff [5]. In this article a summary of methods that were used previously is given. A new method is formulated, where the iced shape is being approximated using so-called P-Fourier descriptors. These are essentially, a Fourier series approach fitted to the iced shape. The results show good accordance with geometrical comparison parameters, however the added advantage is that the process can be automated and less subjective intervention is required which is beneficial for the measurements, since human error can be present in regular measurements of the geometrical parameters.

3.2 Comparison parameter to be used

The comparison parameters that will be used in this report, are a selection from the parameters that were described and mentioned in [4] and [5]. Since the goal is not to find the best comparison parameter, but rather have a comparison parameter that is relatively easy to measure. And also, since the fact that no clear horn has formed in the case considered, the horn angle for example was not measured. This was necessary because in certain cases the iced shape had a horn, which is the measured maximum thickness, but was on the opposite side. This in turn would distort and give a bad comparison value, while other geometrical aspects were relatively closer to the experimental values.

The comparison values that will be used are as follows:

- maximum thickness T_{max}, this is calculated by measuring the length of the line normal to the point on the surface and intersects the iced shape.
- area of cross section A_{cs}, calculated by multiplying the absolute normal length (of the ice) by the width of the panel being evaluated[1]
- maximum forward width FW_{max}, this is calculated by find the maximum y value of the iced shape and the minimum y value
- maximum width in x direction at $y = 0$ denoted by $W_{y=0}$, which is calculated by drawing a straight line from the coordinate $(0, 0)$ to the left
- maximum width in direction with respect to $x = 0$ denoted as $W_{x=0}$, similar to $W_{y=0}$ however now the distance is calculated by finding the maximum value measured from $x = 0$

[1] The airfoil has a certain number of points loaded into Multi-ICE, from these points it draws panels in between them. The center points of the panels are then used to find the normal vector, and the width of the panel is used in the area calculation

- lower x extension x_{low}, which is calculated from $(0, 0)$ the the extension of the iced shape below the leading edge.
- higher x extension x_{high}, which is calculated from $(0, 0)$ the the extension of the iced shape above the leading edge.

For clarity, these measured quantities are illustrated in figure 3.1

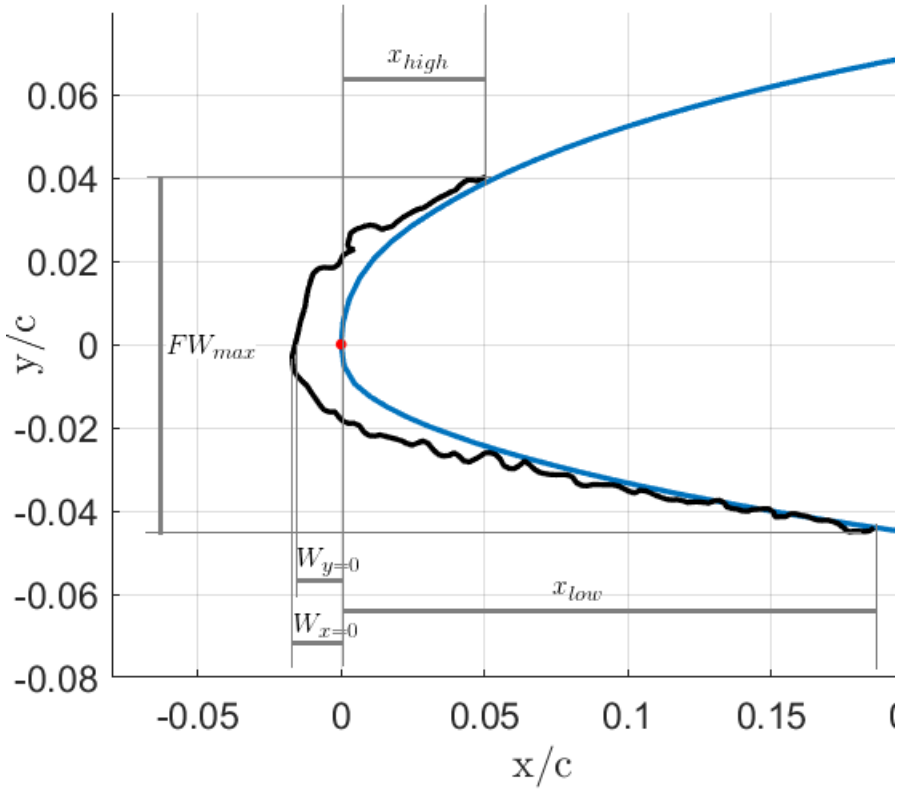


Figure 3.1: Illustration of some of the comparison parameters: maximum forward width, maximum width in x direction at $y = 0$, maximum width in x direction with respect to $x = 0$, lower x extension and higher x extension

To compare numerical simulations of the iced shape with the experimental data, the following equation is used, which is taken from [5].

$$\Delta x = \frac{|x_2 - x_1|}{\frac{1}{2}(x_2 + x_1)} \tag{3.1}$$

In this equation 3.1, x_i is for example one of the comparison parameters, such FW_{max}. The subscript is for identifying which situation to compare, so x_1 could be the experimental result of the specific comparison parameter and x_2 the numerical simulated value. The resulting value represents the average difference between experimental and numerical, with the average taken between those results, and it is desired that the value of Δx is as low as possible.

Then the computation is repeated for all other comparison parameters, and finally a weighted average is used to evaluate the overall performance of a simulation compared to the experimental result.

Example calculation of comparison parameter

Suppose in the experimental result the measured maximum forward width $FW_{max,1}$ has the dimensionless value 0.2 . And the simulated result gives a maximum forward width of $FW_{max,2} = 0.25$, then the comparison parameter gives

$$\Delta FW_{max} = \frac{|0.25 - 0.2|}{\frac{1}{2}(0.25 + 0.2)} \approx 0.22. \tag{3.2}$$

Suppose another simulation result gives an even lower value for this comparison parameter, then the other simulation is considered better, because the difference is smaller between experimental and simulated result.

Chapter 4

Ice accretion simulation using Multi-ICE

In this research assignment, doing simulations of ice accretion was an important part of the assignment. The process of evaluating ice accretion, is a complex task, because it requires to solve for the flow field around an airfoil. Then by using this flow field, iteratively water particles are released in this flow field, to find the particles that impinge on a surface. This requires to solve for the equation of motion multiple times iteratively. It can be seen therefore, it would be of great use to have some simulation software capable of doing this. Fortunately, at CIRA a program called Multi-ICE has been developed, which is capable of doing this.

Multi-ICE is a simulation software capable of computing the ice accretion on a 2D airfoil. This can be done for single, as well as for multi-element airfoils, such as a combination of an airfoil, flap and slat. The program and its work flow can be divided in the following steps:

- 1. Calculation of flow field using a potential panel method (with symmetric singularities). Alternatively the program also accepts precomputed flow fields.
- 2. Calculation of water droplet trajectories, so the amount of water that impinges onto a surface can be computed.
- 3. Calculation of the thermodynamic balance, by the different heat flux mechanisms, as explained in 2.5
- 4. Compute the amount of water that freezes, from which the new geometry of the airfoil is determined.

The program has different parameters, and choices can be made on how to do the simulation. An evaluation has been performed on a NLF-0414 airfoil, see figure 4.1. The coordinates of this airfoil can be found in Appendix B.2 and were taken

from [6]. Note that the coordinates used in this report, initially contained some minor copying errors, from the original source. Since this was found out a late stage of the internship, work was carried out with not exactly the same coordinates as found in NASA's report. Nevertheless, the influence was not expected to be much, because the errors are small, but they can be seen later in the report in the pressure coefficient figures of the airfoil.

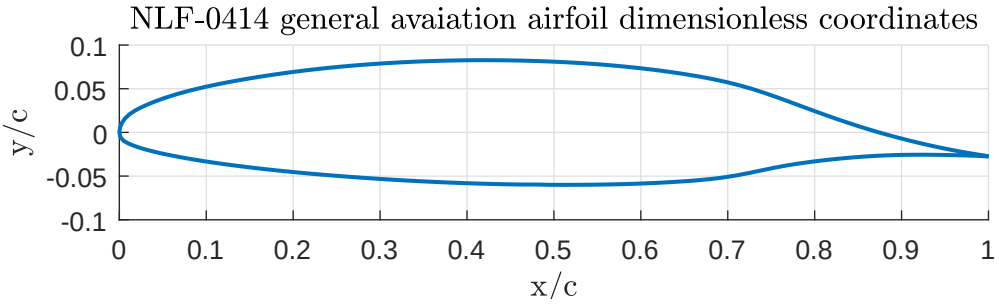


Figure 4.1: Dimensionless airfoil coordinates of NLF-0414 general aviation airfoil

4.1 Experimental results

The experimental results that were used to compare the numerical with the experimental results were taken again from [6]. The test cases that were used are numbered as run no. 629, 630 and 631 in NASA's report. The measurement results for the aerodynamic coefficients are shown in table 4.1.

Table 4.1: Aerodynamic lift and drag coefficients for test run 629 to 631

Run no.	$C_{d-clean}$	C_{d-iced}	$C_{l-clean}$	C_{l-iced}
629	0.0105	0.0131	0.470	0.447
630	0.0105	0.0154	0.467	0.437
631	0.0105	0.0207	0.469	0.433

The presented lift and drag coefficients were measured in the icing wind tunnel of NASA. In another tunnel the aerodynamic performance was tested. From these tests the pressure coefficient distribution is present as well. Since, the data was not available digitally, only some points of the C_p graph were manually digitized using [7]. Since the conditions in both wind tunnels are not exactly the same, the one which is closest to the cases 629 to 631 is presented. Also the angle of attack is zero, since no data of 0.3 degrees for the C_p distribution was available. The results from the pressure coefficient between the ones calculated in Multi-ICE and the experimental results can be seen in figure 4.2.

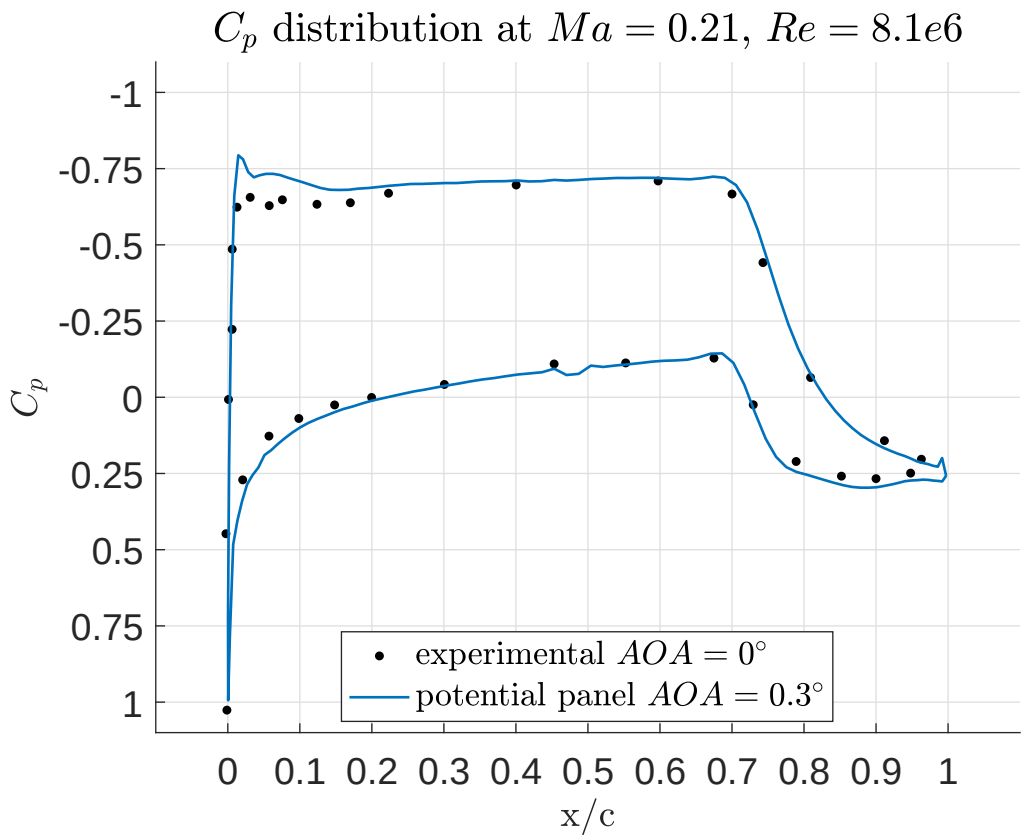


Figure 4.2: Pressure coefficient distribution around the NLF-0414 airfoil at $Ma = 0.21$ and $Re = 8.1e6$

It can be observed that at the leading edge the maximum occurring C_p is about 1 at the bottom and -0.65 at the top of the airfoil. Near the trailing edge the pressure coefficients reaches a value near 0.25. Towards the leading edge when considering the top of the airfoil, as expected the pressure coefficient decreases, meaning a decrease in pressure, which is caused by the increased velocity at the top of the airfoil. While at the bottom due to a decrease in velocity the pressure increases at the bottom. Another interesting aspect is the negative pressure coefficients at some parts even at the bottom of the airfoil, which would result in a downward force rather than a lifting force at some sections along the bottom of the airfoil. Finally, the error in one of the airfoil coordinates can be seen, with a little 'bump' at the bottom of the airfoil around $x = 0.5$. Luckily it is downstream, and the effect upstream is expected to be rather small.

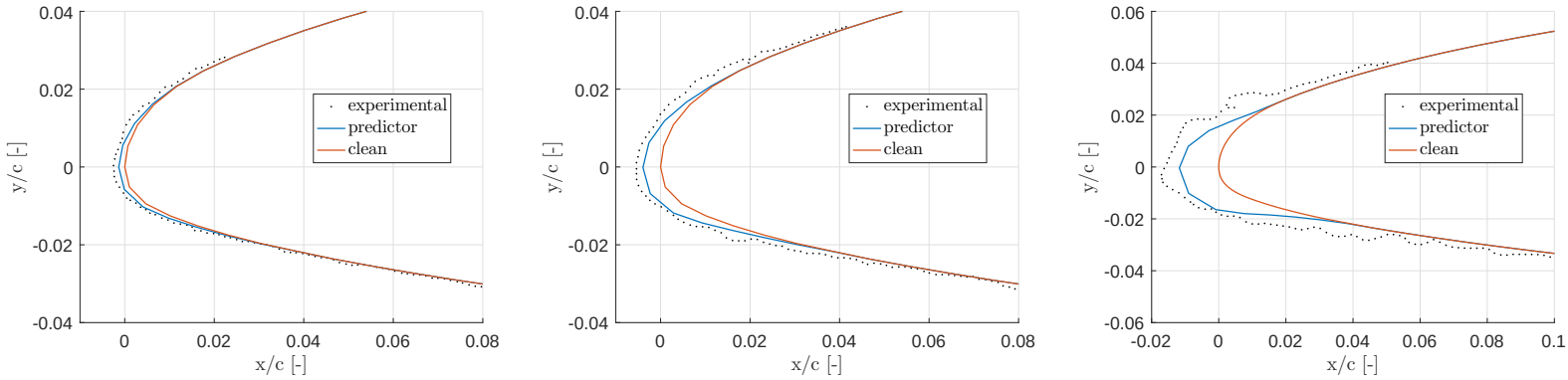
The test conditions of the cases 629, 630 and 631 have the following common parameters:

- Airfoil chord $c = 0.9m$
- Angle of attack $AOA = 0.3°$
- Mach number $M = 0.21$ and $V_\infty \approx 66.9\ m/s$

- Stagnation temperature $T_s = -15°C$
- Median volumetric diameter $MVD = 20\mu m$
- Liquid water content $LWC = 0.440\ g/cm^3$
- Static pressure $P_s = P_{atm} \approx 1.01325$

The only difference between the runs is in spraying time or also denoted as accretion time. For test case 629 the accretion time is 1.4 minutes, for case 630 this is 4.1 minutes and for test case 631 the accretion time is 15.3 minutes.

Initially, only a predictor step was performed, using the potential method to compute the aerodynamic field. The results can be seen in figure 4.3.



(a) Case 629 *(b)* Case 630 *(c)* Case 631

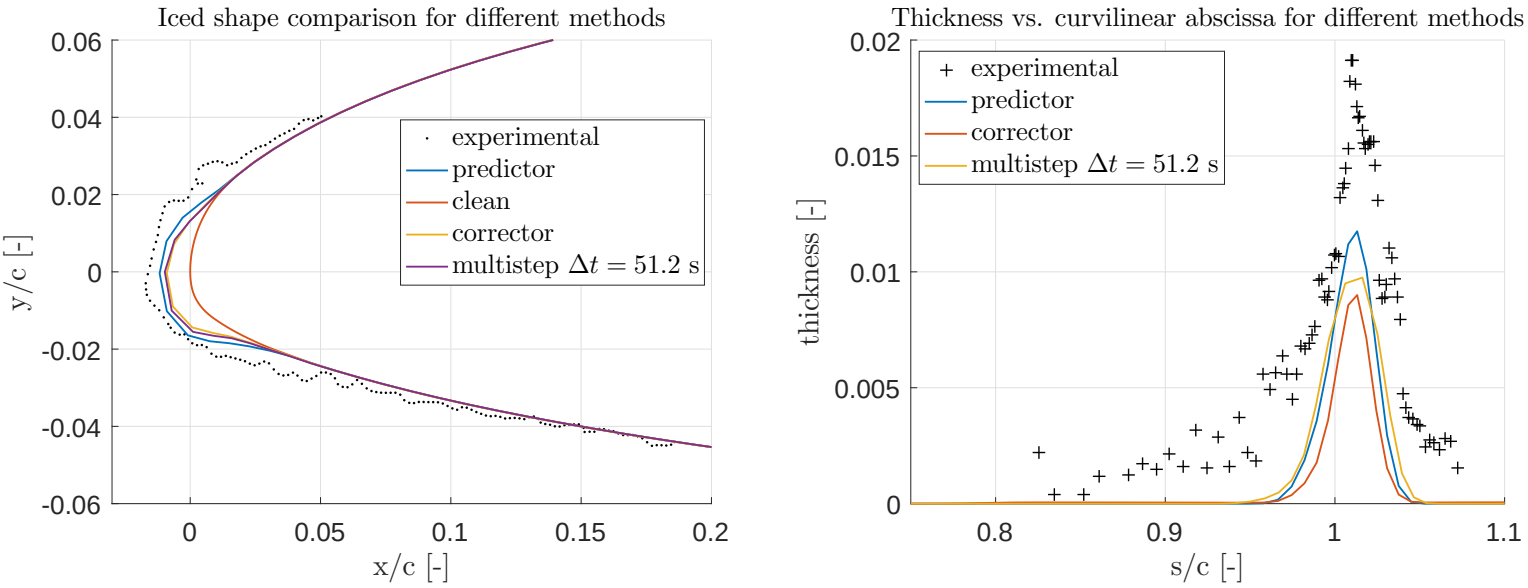
Figure 4.3: Comparison between the test cases 629,630 and 631 with the numerical predictor method in Multi-ICE.

As can be seen for small accretion times the predictor method seems to be quite close to the experimental result, however for longer accretion times, the predictor method has worse results.

4.2 Different simulation methods

In this section the different simulation methods in Multi-ICE will be demonstrated and compared. As mentioned before Multi-ICE has different possibilities to do simulations, these are: 1. predictor step only, 2. predictor and corrector step (referred to as corrector step) and 3. multi-step method, which is the predictor and corrector step applied multiple times, where a certain time span Δt is set by the user, and is repeated multiple times. To focus on the different simulation possibilities, only test case 631 will be considered hereafter, since this was the case that has the worse results with a predictor step only. Naturally, it is desirable to have a good prediction

for test case 631 as well. The results for the different simulation method is shown in figure 4.4.



(a) Iced shape

(b) Thickness vs. curvilinear abscissa

Figure 4.4: Comparison between predictor, corrector and multi-step methods for test case 631

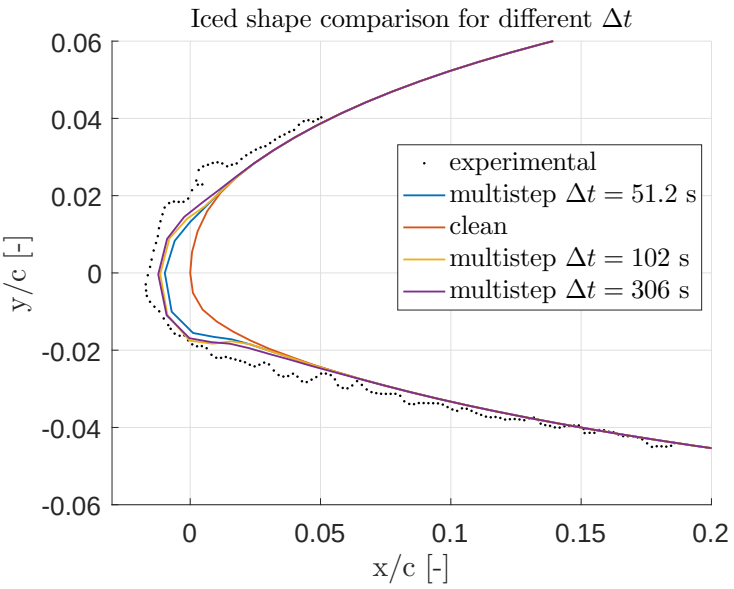
Table 4.2: Quantitative comparison between numerical and experimental iced shape for the predictor, corrector and multi-step method

Method	T_{max}	A_{cs}	FW_{max}	$W_{y=0}$	$W_{x=0}$	x_{low}	x_{high}	average
Predictor	0.39	0.98	0.55	0.33	0.37	1.2	0.96	0.69
Corrector	0.64	1.2	0.55	0.59	0.62	0.24	1.2	0.72
Multi-step ($\Delta t = 61.2s$)	0.63	1.2	0.54	0.50	0.55	0.25	1.2	0.69

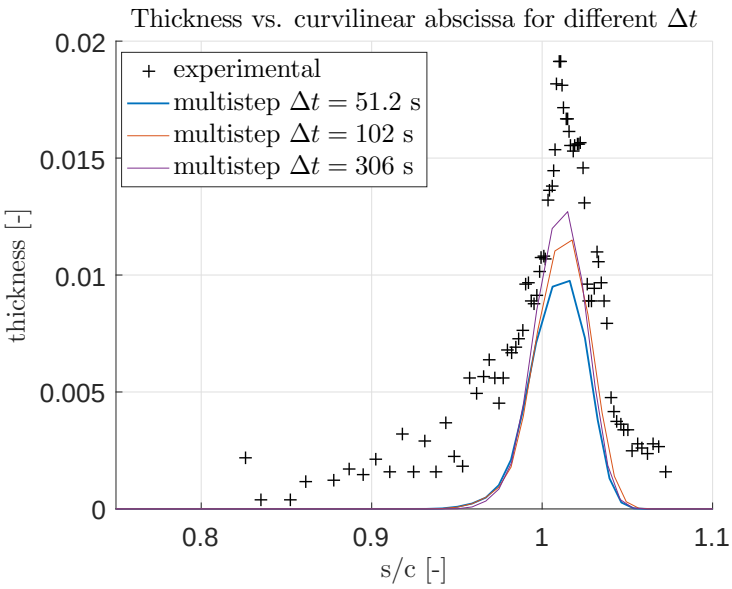
It is observed that the different simulation methods produce similar results, with small differences. However, most ice has formed when using the predictor method, which seems to be the closest to the experimental result. The multi-step approach is also better at the bottom of the leading edge compared to the corrector method, but still it is worse than the predictor method. However, also the predictor method is far away from the experimental result. These results are also observed when the quantitative comparison is considered, in table 4.2. Next the influence of the multi-step approach with different time steps will be evaluated.

4.3 Different time steps in the multi step method

In this section the influence of the time step will be evaluated for the multi-step method, to see if it is possible to improve the simulation by using a different value for Δt. The following simulation were performed: 1. three steps with $\Delta t = 306s$, 2. nine steps with $\Delta t = 102s$ and fifteen steps with $\Delta t = 61.2s$. The results are shown in figure 4.5.



(a) Iced shape



(b) Thickness vs. curvilinear abscissa

Figure 4.5: Comparison between different time steps for the multi-step method for test case 631

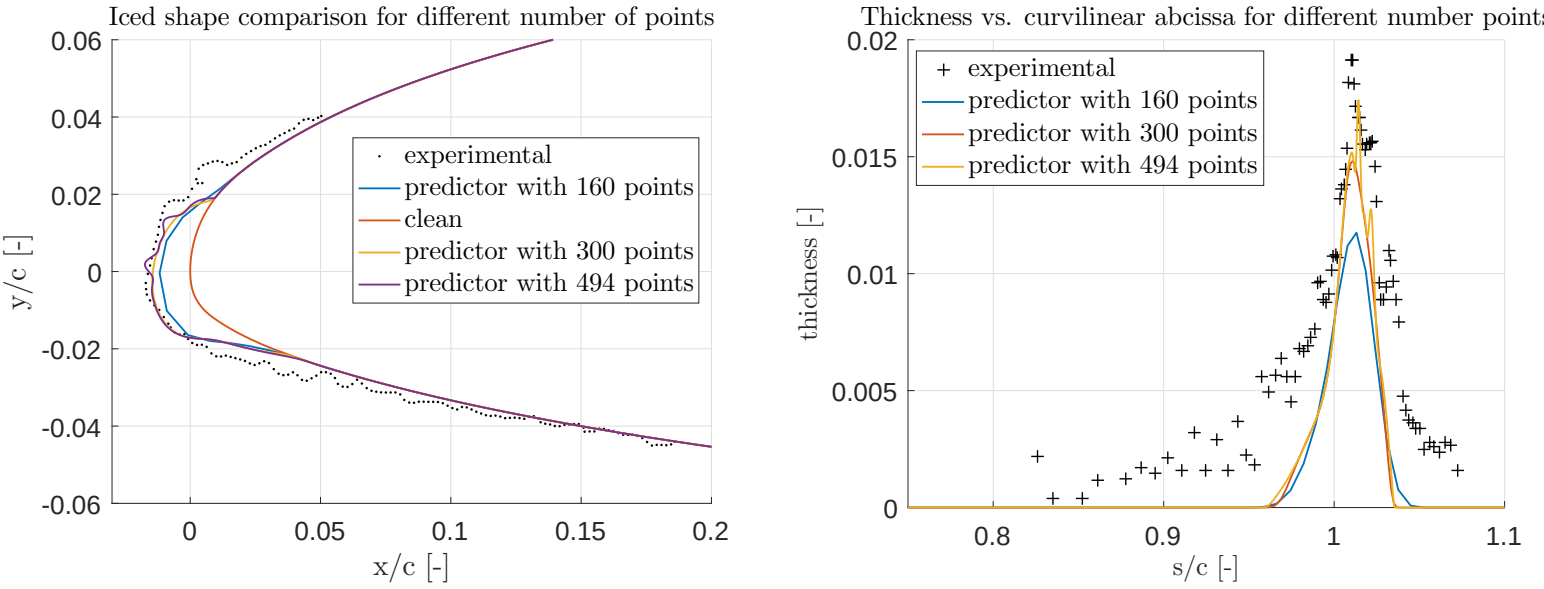
Table 4.3: Quantitative comparison between using different time steps in the multi-step method

Method	T_{max}	A_{cs}	FW_{max}	$W_{y=0}$	$W_{x=0}$	x_{low}	x_{high}	average
Multi-step ($\Delta t = 61.2s$)	0.63	1.2	0.54	0.50	0.55	0.25	1.2	0.69
Multi-step ($\Delta t = 102s$)	0.54	1.2	0.52	0.46	0.39	0.27	1.2	0.65
Multi-step ($\Delta t = 306s$)	0.47	1.2	0.50	0.44	0.37	0.32	1.2	0.64

Only small differences are observed by using a different time step. The amplitudes tend to decrease with decreasing the time step. And the largest amplitude is observed, with the largest time step. However, the amplitude of the ice thickness is still small compared to the experimental result.

4.4 Airfoil number of points

Up until now, the airfoil used was the one described by the points in Appendix B, however the number of points on this airfoil is 160 points. Since the panels, that are being computed depend on these coordinates, they are controlling the accuracy of the prediction of the ice. Having more points, will result in more panels and hence more discretization and thus more resolution around the airfoil. Therefore, using XFOIL 6.99 (a program for the design and analysis of subsonic airfoils) (XFOIL) the same airfoil was discretized into 300 points and 494 points (maximum possible points in XFOIL). This has been done by using the so-called `ppar` routine in XFOIL. The results between different number of points are compared and shown in figure 4.6, note that all simulations here are done with the predictor method only, so we can see the influence of one parameter only on the iced shape.



(a) Iced shape

(b) Thickness vs. curvilinear abscissa

Figure 4.6: Comparison between different number of points with a predictor method for test case 631

Table 4.4: Comparison between different number of points loaded into Multi-ICE.

Method	T_{max}	A_{cs}	FW_{max}	$W_{y=0}$	$W_{x=0}$	x_{low}	x_{high}	average
Predictor (160 points)	0.39	0.98	0.55	0.33	0.37	1.2	0.96	0.69
Predictor (300 points)	0.17	0.88	0.67	0.13	0.16	1.3	1.3	0.65
Predictor (494 points)	0.0039	0.83	0.66	0.024	0.018	1.2	1.3	0.59

Immediately, it can be seen that the more points we have, the better the approximation to the experimental results, this is also what was expected, because the

results have more resolution. Further, increasing the number of points to the maximum also develops a small horn at the front which was not present before. However, the horn over predicts the experimental results. The rest of the results between 300 and 494 points are very close to each other, which also indicates using more points in general is better for the accuracy.

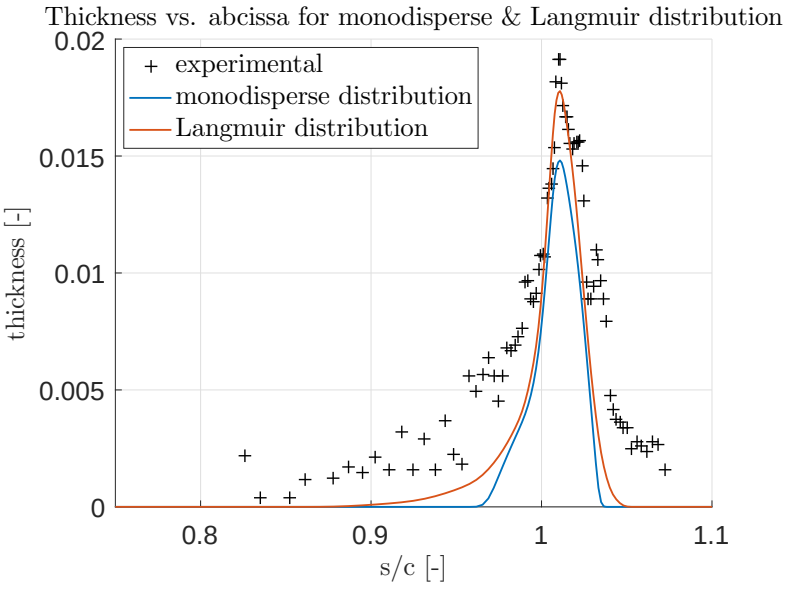
4.5 Liquid droplet diameter distribution

One other possibility is to change the input of the droplet distribution. All the above work was done for a mono disperse droplet distribution of $MVD = 20\mu m$, which is a commonly used value for ice accretion analysis. However, in reality the droplets are not mono disperse, but have a certain distribution. Another possible distribution used is a discrete Langmuir distribution, which was taken from [8]. The table with the droplet diameter and occurring percentage is shown below in table 4.5.

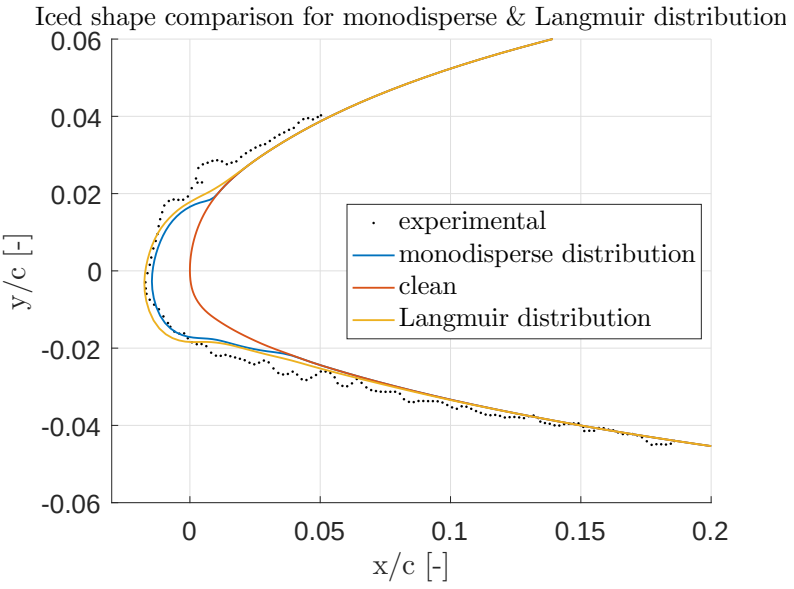
Table 4.5: Langmuir droplet distribution

$D[\mu m]$	6.2	10.4	14.2	20.0	27.4	34.8	44.4
$LWC[\%]$	5	10	20	30	20	10	5

Again a comparison with the predictor method has been made, but with the number of points is 300 now. The results are shown in figure 4.7



(a) Iced shape



(b) Thickness vs. curvilinear abscissa

Figure 4.7: Comparison between a mono disperse and a Langmuir droplet distribution with 300 points around the airfoil for test case 631

Table 4.6: Comparison between normal mono disperse droplet distribution and Langmuir droplet distribution

Method	T_{max}	A_{cs}	FW_{max}	$W_{y=0}$	$W_{x=0}$	x_{low}	x_{high}	average
Predictor with Langmuir	0.016	0.64	0.27	0.048	0.024	0.33	0.77	0.30
Predictor (300 points)	0.17	0.88	0.67	0.13	0.16	1.3	1.3	0.65

It is observed, that when using the Langmuir distribution the extension in x direction for the top (x_{top}) and bottom (x_{low}) iced surface, is longer now and is even better than the regular 300 points airfoil results with the predictor method. However, clearly at the front the amount of ice is over predicted, but overall it gives a better indication of the icing on the airfoil.

Chapter 5

Simulations using precomputed flow field of FLUENT

Aside from being able to compute the flow field around an airfoil using the built-in potential panel method, it is also possible to load in precomputed flow fields. To see what the influence is of other computational methods, in this chapter precomputed flow fields will be used to predict the ice accretion on the airfoil.

As the flow field cannot simply by recalculated within Multi-ICE, it is only possible to do a predictor analysis of the ice accretion. The following cases will be treated.

- 1. Flow field computed using the Euler equations, which is also inviscid.
- 2. Flow field computed with viscosity and the following turbulence models:
  - Spalart-Allmaras with default settings, ideal gas law and Sutherland's law
  - Standard $k - \omega$ with default settings, ideal gas law and Sutherland's law
  - Transition SST, with default settings, ideal gas law and Sutherland's law

The goal in using these different models is to see which one gives the best prediction for the lift and drag coefficient, because the lift and drag coefficient will be used to determine, whether or not convergence has been achieved.

5.1 Workflow

To get the ice accretion results in Multi-ICE by using an externally computed flow field, some steps have to be taken in between, since the output and input files are not fully compatible. Another important aspect, is that in the simulation the flow domain is very large, namely twenty chords away from the airfoil, in a C-type grid. This is necessary in order to obtain good simulation results, and capture all the fluid dynamics around the airfoil.

Of course, this comes with a price, namely the huge amount of data, which is not necessarily needed in Multi-ICE. Therefore a small domain of interest has to be extracted out of the complete solution domain. The steps required to do so will be explained in this section, however, since some parts are more easy to visualize these have been added into Appendix A.

- 1. Make a C-type grid around the airfoil, Appendix A.1
- 2. Load the grid into FLUENT and perform the simulation that is required, Appendix A.2
- 3. Now before exporting the obtained solution data, the domain of interest has to be specified, this has been explained in Appendix A.3
- 3b. Optional: extracting the pressure coefficient is more easily done in FLUENT, by exporting it as an ASCII file.
- 4. Import data into TECPLOT 360 2016 (a program for post-processing (CFD) data) (TECPLOT) and from here write an ASCII file of the positions and velocities.
- 5. Read ASCII files into Excel, and restructure in the format as required by Multi-ICE, see manual of Multi-ICE for the input structure.
- 6. Read files into Multi-ICE and perform ice accretion predictor simulation.

5.2 Comparison flow field around airfoil

Before showing the results of the ice accretion on the precomputed flow fields, the pressure coefficient distribution around the airfoil will be shown between the different simulation methods, that is the panel method, Euler and Navier-Stokes (with different turbulent models).

The C_p versus the dimensionless x coordinate are shown in figure 5.2

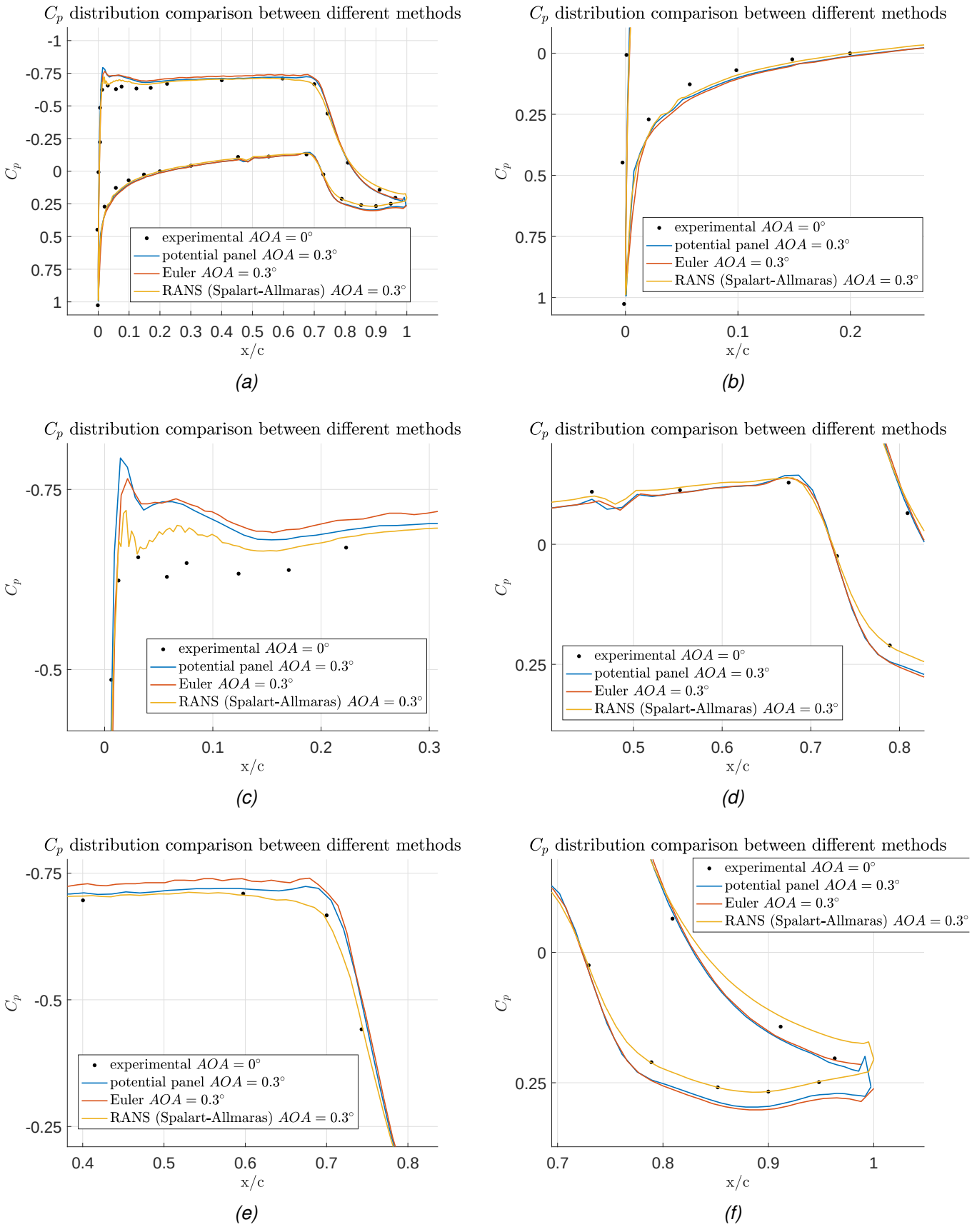


Figure 5.1: Comparison between the potential method and the Euler simulation and a viscous simulation with the Spalart-Allmaras turbulence model. a)overall C_p distribution, b)bottom leading edge, c)top leading edge, d)bottom near trailing edge, e)top near trailing edge f) trailing edge

It can be seen first of all that there is a difference between experimental and numerical data, this is because of the angle of attack is not exactly the same, but another reason is because the different methods used have different formulations. The potential panel and Euler method are inviscid, while the RANS method with (Spalart-Allmaras) turbulence model is a viscid model, and hence contains more of the physics. This is also observed, because the the result for the RANS method is in general closer to the experimental data, especially near minima and maxima of the C_p diagram. The values for the lift and drag coefficient are as follows:

Table 5.1: Comparison of the pressure coefficient distribution between different turbulence model

Run no.	C_d	C_l
Potential	0.00179	0.570
Euler	0.000613	0.534
RANS (SA)	0.00966	0.488

Next the aerodynamic results for the different turbulent models will be presented. The settings that were used for the different turbulence models, were the default values in FLUENT

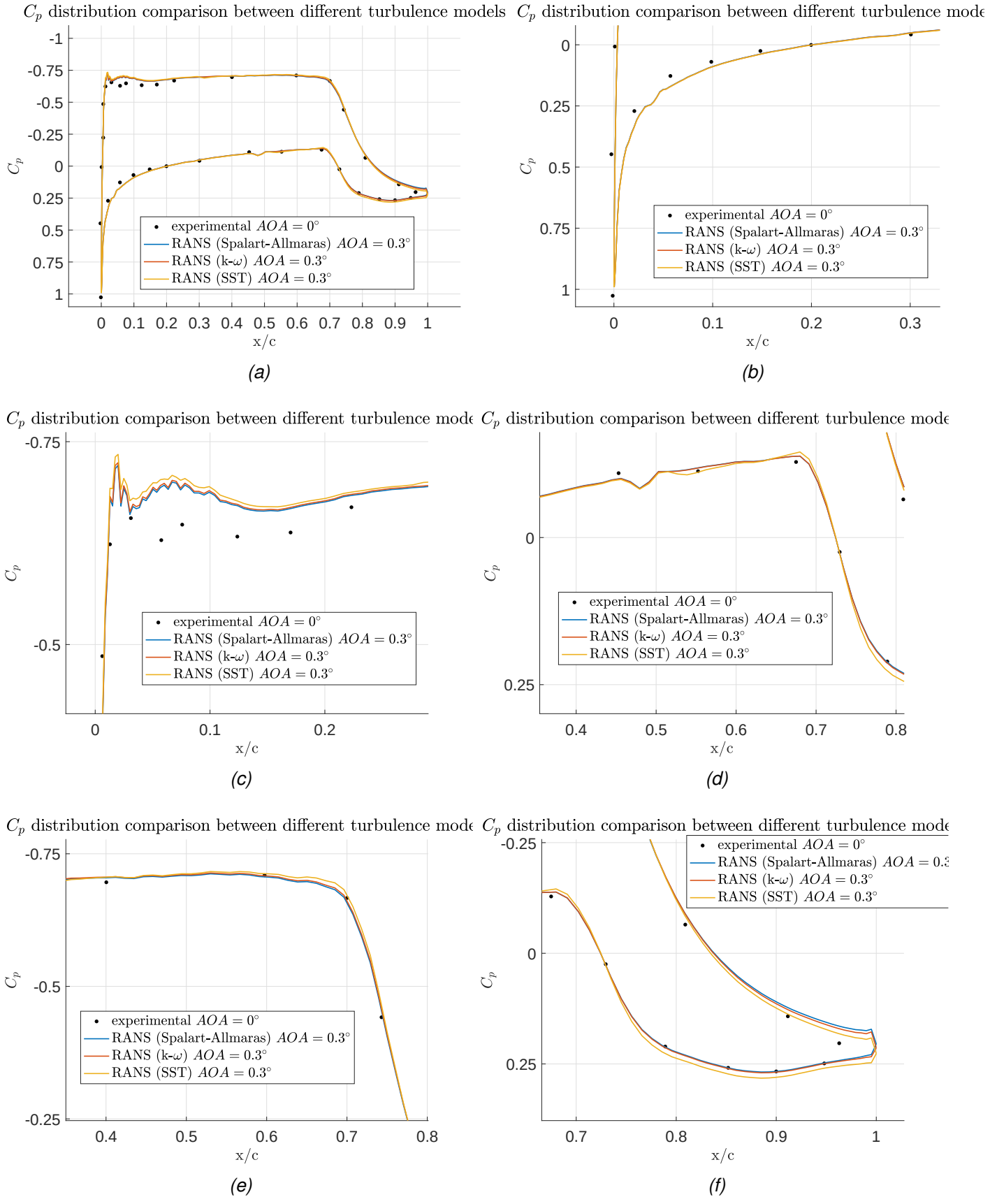


Figure 5.2: Comparison between the difference turbulence models, the Spalart-Allmaras, $k - \omega$ and SST. a)overall C_p distribution, b)bottom leading edge, c)top leading edge, d)bottom near trailing edge, e)top near trailing edge f) trailing edge

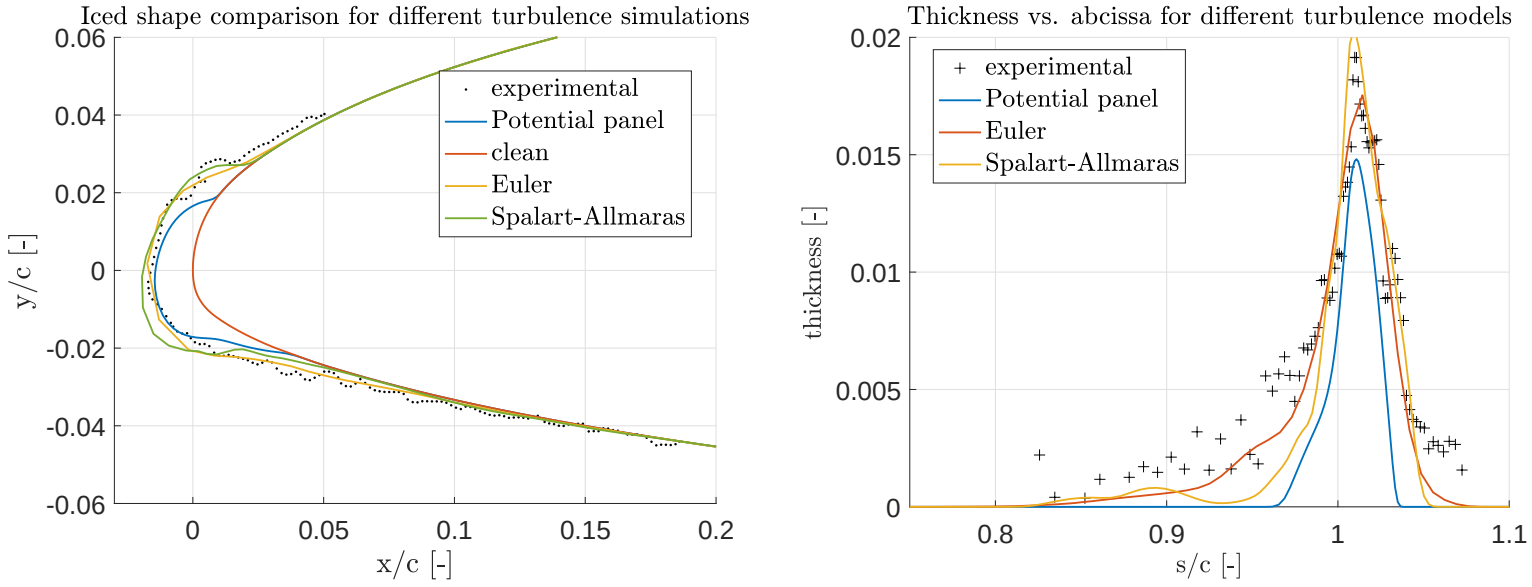
Overall, it is observed that the different turbulence models have similar results. With slight deviations at the leading and trailing edge, caused by the different turbulence models used for the closure of the RANS equations. The lift and drag coefficients are as follows:

Table 5.2: Comparison of the pressure coefficient distribution between different turbulence model

Run no.	C_d	C_l
RANS (k-ω)	0.00963	0.489
RANS (SST)	0.00478	0.506
RANS (SA)	0.00966	0.488

5.3 Comparison between the potential and precomputed methods

In this section the comparison between the potential panel method and the precomputed flow fields will be discussed. Earlier it was observed that the pressure coefficient distributions are similar for the different simulation methods. Here the ice accretion is evaluated of the different methods.



(a) Iced shape

(b) Thickness vs. curvilinear abscissa

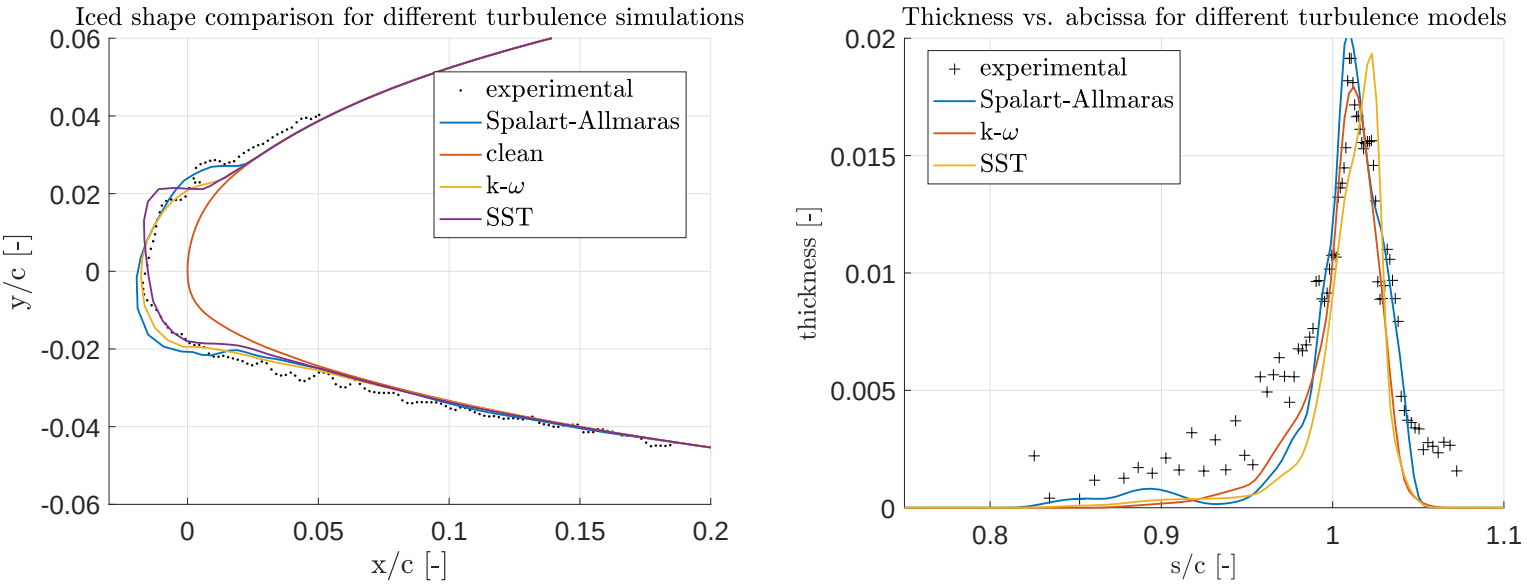
Figure 5.3: Comparison between the potential method and the Euler simulation and a viscous simulation with the Spalart-Allmaras turbulence model.

Table 5.3: Quantitative comparison between the potential panel method, the Euler simulation (FLUENT) and the Spalart-Allmaras simulation (FLUENT)

Method	T_{max}	A_{cs}	FW_{max}	$W_{y=0}$	$W_{x=0}$	x_{low}	x_{high}	average
Potential panel	0.17	0.88	0.67	0.13	0.16	1.3	1.3	0.65
Euler (inviscid)	0.11	0.38	0.15	0.17	0.13	0.15	0.42	0.22
Spalart-Allmaras(viscid)	0.15	0.29	0.14	0.15	0.13	0.0050	0.59	0.21

It can be seen that for iced shape for the Spalart-Allmaras simulations has the least deviation from the experimental results regarding the comparison parameter. This can also be observed from the figures showing the iced shape.

Next, the difference between the different turbulence models is shown.



(a) Iced shape

(b) Thickness vs. curvilinear abscissa

Figure 5.4: Comparison between the different turbulence models used.

Table 5.4: Quantitative comparison between the different turbulence models for the icing results

Method	T_{max}	A_{cs}	FW_{max}	$W_{y=0}$	$W_{x=0}$	x_{low}	x_{high}	average
$\kappa - \omega$ (viscid)	0.025	0.44	0.16	0.081	0.044	0.23	0.34	0.19
SST (viscid)	0.10	0.53	0.13	0.073	0.024	0.0050	0.50	0.19
Spalart-Allmaras(viscid)	0.15	0.29	0.14	0.15	0.13	0.0050	0.59	0.21

The difference between the turbulence models is relatively small, but the $k - \omega$ and SST turbulence models are a little bit more accurate. These are also a bit more complex than the Spalart-Allmaras turbulence model, because they contain more

transport equation to the closure problem of the RANS. Therefore, they could be considered to capture the turbulence more accurate. However, this is not true in general and different turbulence models have their limitations. No further investigation was done, to see what causes the small differences between the turbulence models.

Chapter 6

Simulations in OpenFOAM

In this chapter of the report, the process of making a simulation of the flow around an airfoil together with a simulation of droplets impingements is presented. Because all aspects of doing these simulations are done in a completely different environment, namely OpenFOAM, the different aspects such as meshing, setting up a case will also be presented. The main idea, behind doing similar simulations as to the simulations done before, is to see how well both methods compare. By methods we specify the following. The first method that was shown was to calculate the flow field around an airfoil using FLUENT, and then by extracting parts of the flow field near the airfoil and using these as input for Multi-ICE the ice accretion simulation was done. The second method is done exclusively in the OpenFOAM environment, both the flow calculations as the impingement calculations were done in OpenFOAM.

Another motivation, to look at the results of OpenFOAM, was because the results between the impingement of particles is significantly different between the results of Multi-ICE and OpenFOAM. Since, the particle impingements are one of the main important parameters in determining the correct iced shape around an airfoil, the attention in this chapter will be to try to understand what causes the different results between OpenFOAM and Multi-ICE in terms of impingement results.

6.1 Work flow

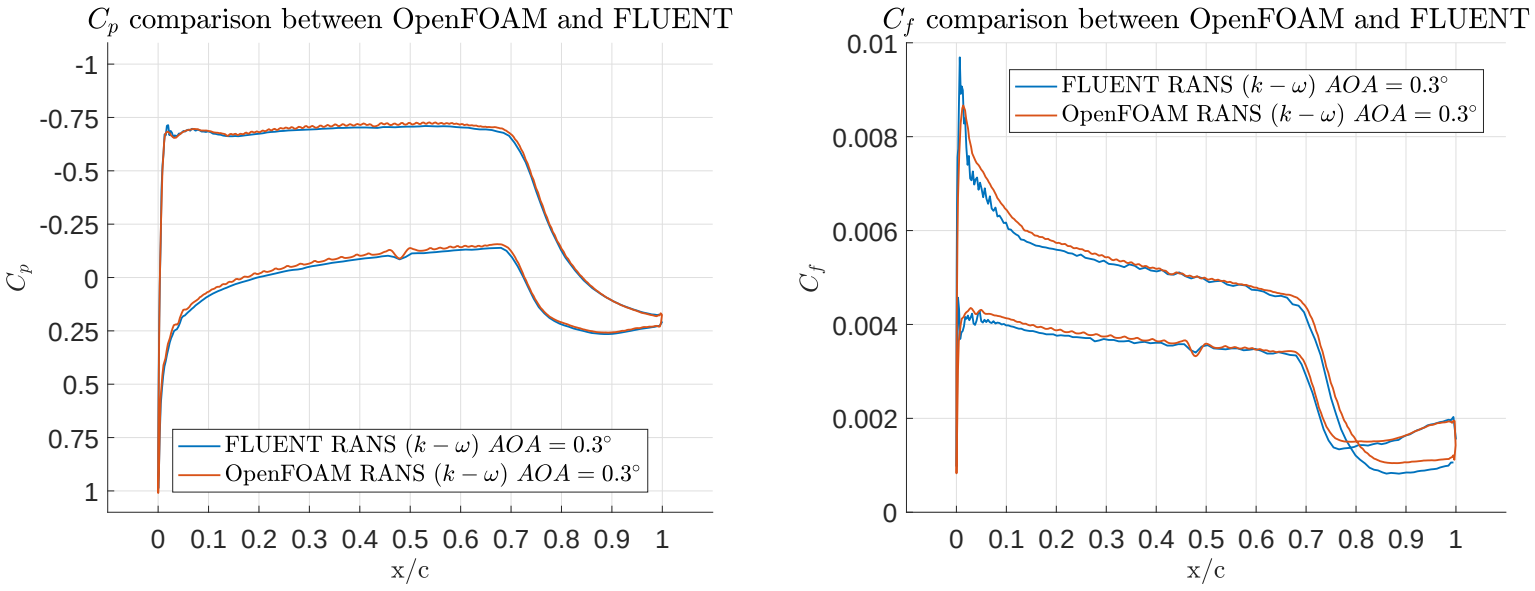
The work flow in the OpenFOAM environment consists of similar steps as the steps done in FLUENT and Multi-ICE.

- 1. First the case and the required files are set up, as explained in Appendix C.1.
- 2. Next a mesh is created around the airfoil using the possible applications inside OpenFOAM.[1].(As explained in Appendix C.2)
- 3. Then the solver *rhoSimpleFoam* can be run, which can be considered to be closest to the simulations performed in FLUENT, this is explained in Appendix C.3
- 4. After running the flow simulation, the impingement simulation will be run. This is done using *uncoupledKinematicParcelFoam*, which injects passive Lagrangian particles in the flow, which means that the flow interaction is one-way only, the flow determines what the particles do. This simulation is ran, so the correct initial positions can be found for the next step. This is explained in Appendix C.4.
- 5. Having determined the injection locations such that all impingements are covered, the modified transient solver *reactingParcelFilmFoam* is used, which is a solver that evolves a film and performs an ice accretion evaluation as well. Because, this simulation is done as a transient problem, it is very different from the Messinger method, and that is why in OpenFOAM the work flow is slightly different. In this report, due to to time limitations this part is not performed, however, in the next chapter, some parts of the particle impingement calculation will be explained, to try to understand what causes the difference.

6.2 Pressure coefficient and friction coefficient comparison

In this section first the aerodynamic results between OpenFOAM and FLUENT are shown. The pressure coefficient and friction coefficient are compared. For simplicity, the simulations in OpenFOAM were only done using the $k-\omega$ turbulence model, hence it will be compared to the corresponding simulation done previously in FLUENT. The resulting pressure coefficient and friction coefficient plots can be seen below:

[1] It was attempted, to use the same mesh that was used in FLUENT. For this procedure, there is a function available named *fluentMeshToFoam*. However, this was not successful. The reason is, due to the usage of so-called wall-functions for the different turbulence models, the initial cell height must above the viscous sublayer and the buffer layer. Therefore a new mesh was created suitable for this situation, using the tools available in OpenFOAM



(a) C_p comparison

(b) C_f comparison

Figure 6.1: Comparison between the C_p and C_f for the simulations done in FLUENT and OpenFOAM using the $k-\omega$ turbulence model.

As can be seen, the pressure coefficient distribution is very similar and only small differences are observed. For the skin friction coefficient the differences are more significant. The reason is in FLUENT the initial cell height at the airfoil was very small with a $y+ = 1$, while with OpenFOAM a $y+ = 100$ was used in the mesh, due to usage of wall functions. This will most likely be one of the causes of the difference.

6.3 Velocity profile comparisons

Next the velocity profiles near one of the impingements is evaluated. The horizontal and vertical velocity components are plotted against x to see how they develop. In the next section the particle impingements will be shown, and there it will become apparent, that there are significant differences between the Multi-ICE and OpenFOAM particles. To see, what might cause this difference, the velocity profiles along constant lines of y through which some impingements pass through are shown. The difference could be, because the flow fields are not exactly the same, between FLUENT and OpenFOAM, hence they will be shown in this chapter.

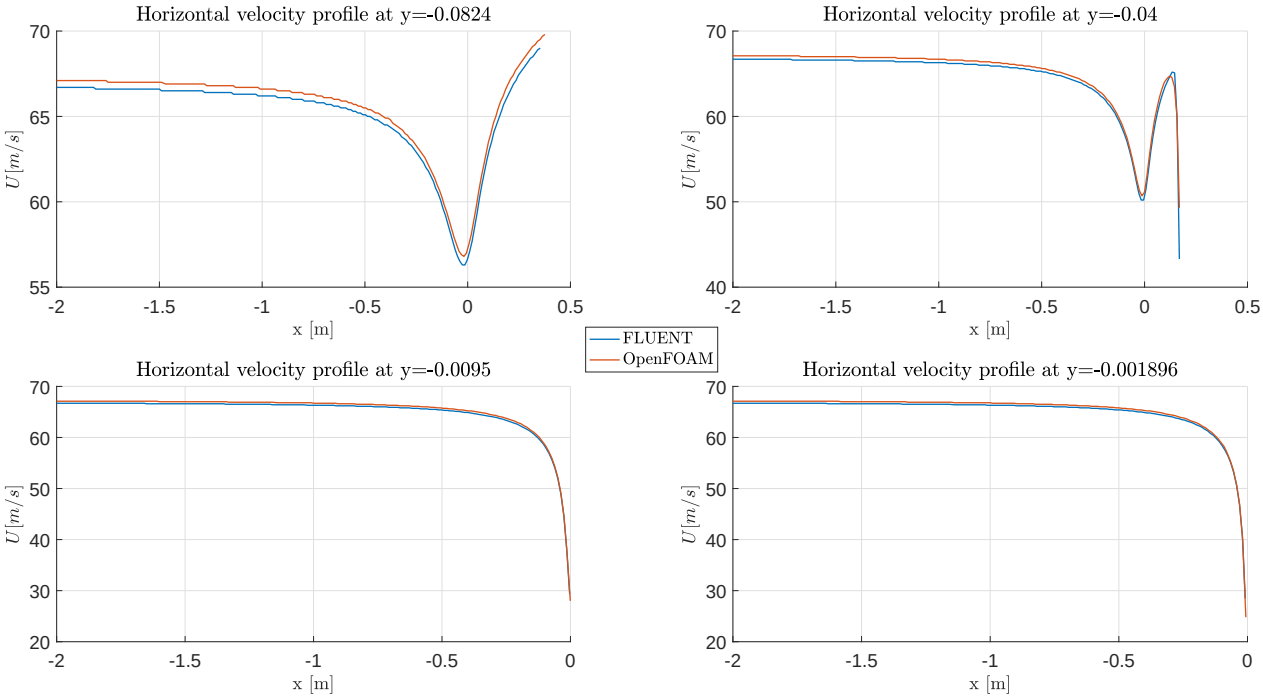


Figure 6.2: Horizontal velocity profiles

It can be seen that there are slight variations in velocity, but the overall shape is quite similar. The difference between the velocities, the average velocities from $x = -2$ to $x = 0$ was calculated, from this it followed that the relative difference between the simulation in FLUENT and OpenFOAM is 0.6% in this case. Which can be considered a small difference, since the magnitude is quite high of the velocity, around 65 m/s.

Next the vertical velocity profiles have been studied and the following results were obtained.

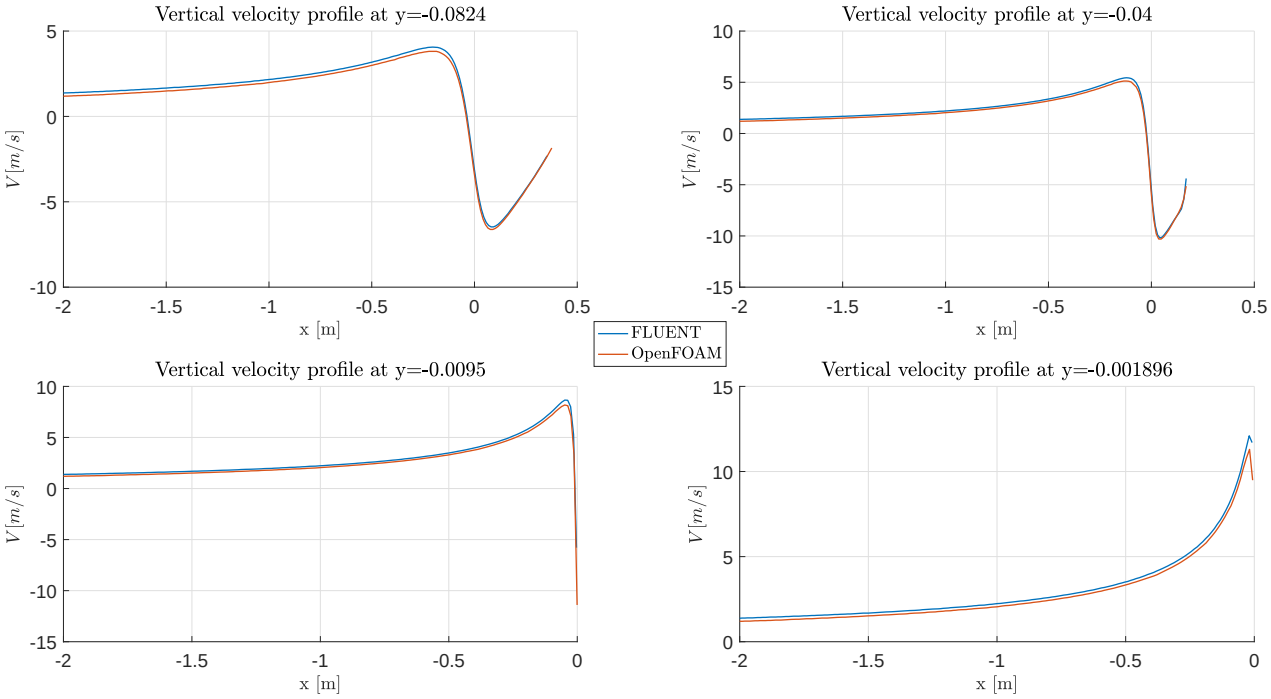


Figure 6.3: Vertical velocity profiles

Again a lot of similarity can be seen in the velocity profile development as it progresses from upstream to to the leading edge. However in this case, because the magnitude of the velocities in the vertical direction is an order of magnitude smaller. The difference between FLUENT and OpenFOAM is more significant. Again an average was calculated between $x = -2$ to $x = 0$, and it followed that an relative difference of 11% is present. This is more significant than the horizontal velocity component. And therefore may be of more influence.

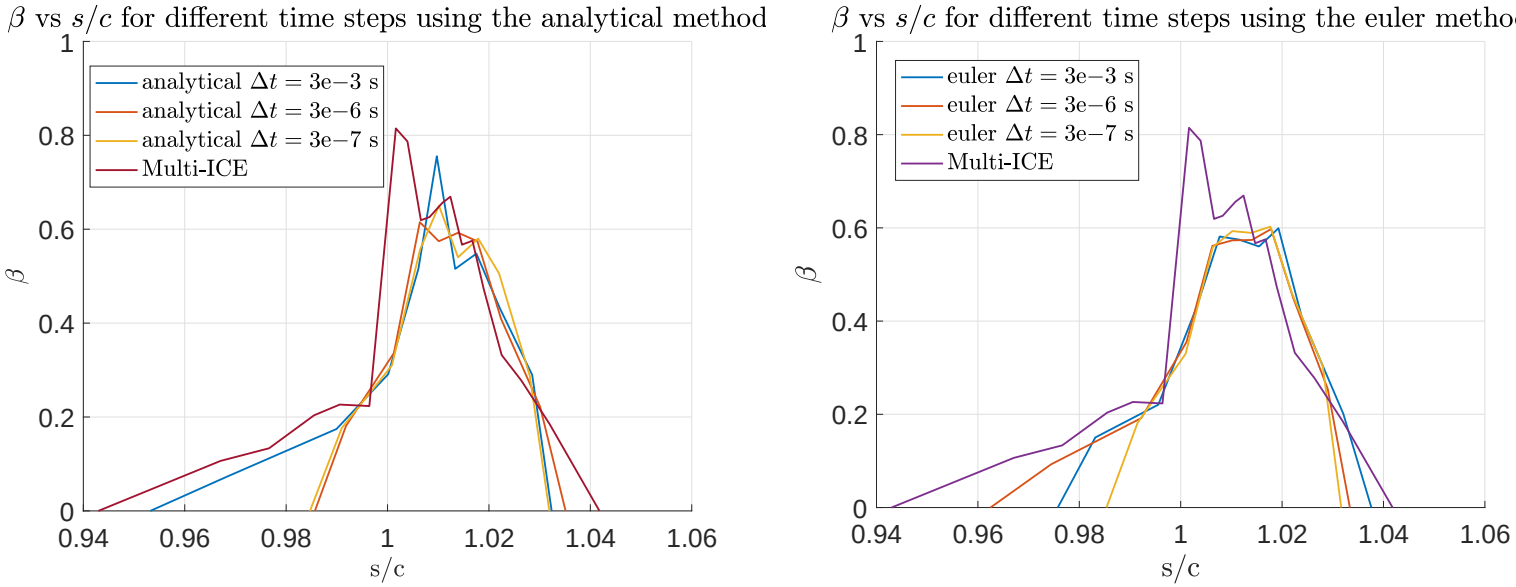
To investigate whether or not the influence of the vertical flow velocity was significant, another simulation in FLUENTwas done with a slightly slower vertical velocity field. Using this velocity field the impingement simulation was performed and compared and it was found that the impact was hardly visible, so it can be concluded that although there is a significant difference in the velocity profile in the vertical direction, it does not affect the impingement calculations significantly and further analyses were performed with the shown simulations.

6.4 Impingement analysis

After performing the flow calculation and comparing both flow fields, the next step was to perform the impingement analyses. In Multi-ICE this is done as one of the

intermediate steps in the predictor step. In OpenFOAM this is done by performing a simulation with *uncoupledKinematicParcelFoam*. The output from Multi-ICE is directly possible to show, because a file called *BETA_OR* will be outputted in the working folder. In OpenFOAM however, this data is not directly present. It has to be calculated by using the final position of the particles after the particles have impinged. For this we know the x and y coordinates of the particles that have impinged. And the x and y of the airfoil. In reality the particles will have crossed the airfoil surface, hence they are put manually by a script to the nearest point to the airfoil. Then by using a linear interpolation between the two point of an airfoil panel, the corresponding curvilinear abscissa coordinate s is found.

Similarly as was done in Multi-ICE particles are released in OpenFOAM, by using particles with a $MVD = 20\mu m$ and initial velocity of $V_{init} = 66.9$. The particles are released at a distance of two meters in front of the airfoil, at y-values such that both the top and bottom limits are ensured to be found. This has to be done manually, while in Multi-ICE this process is automated and the only importance is to check whether or not the limiting trajectories, those are the trajectories that just hit the surface at the top and bottom, are present. The results are shown below



(a) analytical

(b) euler

Figure 6.4: Comparison between using the different integration method in OpenFOAM, for different global time steps.

As can be seen for both the analytical as well as the Euler method, the impingement limits (the distance between the point with $\beta = 0$) are smaller then the predicted impingement limit in Multi-ICE. Moreover for smaller Δt with the analytical method the limits seems go to a converging value, which is significantly different than the

one in Multi-ICE. For the Euler integration method the, for small Δt it seems to resemble the limits of the analytical one, with small Δt. The results shown here, are repeatedly observed, when comparing results between the impingement in Multi-ICE and OpenFOAM. One point of critique, is the fact that in principle the flow field is not exactly the same. This adds some uncertainty to whether or not this is the source that causes the difference. To ensure this was not the case, as explained in the previous section, another flow simulation in FLUENT was performed with a slower vertical velocity, such that the vertical velocity profiles in FLUENT would end up underneath the ones shown in figures 6.3. The results however were not very different and the impingement limits remained approximately the same. To further analyze the particle trajectories and how it evolves, the particle trajectories and its velocities were obtained. For simplicity only a single particle is considered, released at approximately the same y position in both Multi-ICE and OpenFOAM. The reason it is approximately, is because in Multi-ICE the final impingements are not directly controlled since finding the limits has been automated. The results for particles released at $x = -2$ and $y = -0.073, -0.083, -0.091$ is shown in figure 6.5

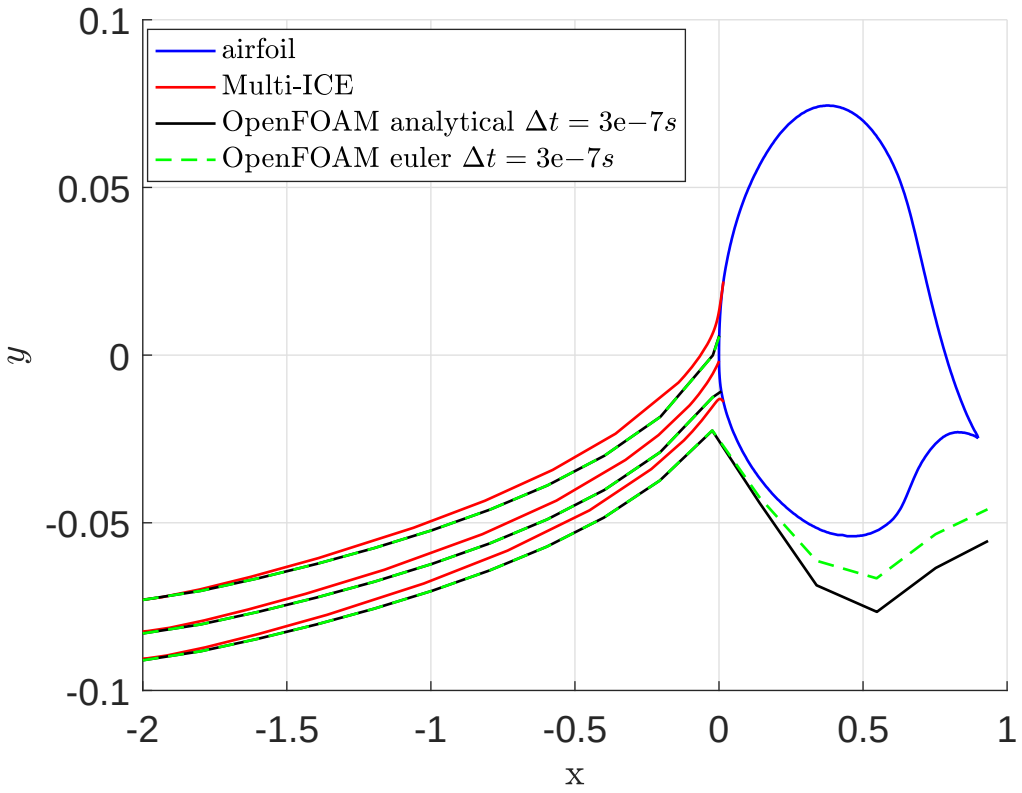


Figure 6.5: Particle trajectories

Although it may not become apparent directly from this image, the particles released in Multi-ICE remain their initial velocities much longer and are affected less

by the forces acting on the particle. For the particles in OpenFOAM, the particle's velocities start to change much earlier and it seems to follow the flow field more than the Multi-ICE particle. To see this the particle's velocities have been plotted against the x coordinate along the trajectory of the particle released at $y = -0.083$ for both the particles in OpenFOAM as well as in Multi-ICE. Along with the particle's velocity also some points along the particles' trajectories were sampled and the flow field velocity at those stations has been extracted. Note, that since the particles released in Multi-ICE and OpenFOAM do not follow the exact same trajectory, differences can be expected. The circles in figure 6.6 correspond to the flow velocity of the flow field in OpenFOAM. The crosses correspond to the flow velocity along the particle trajectory in Multi-ICE and it comes from the flow field computed in FLUENT.

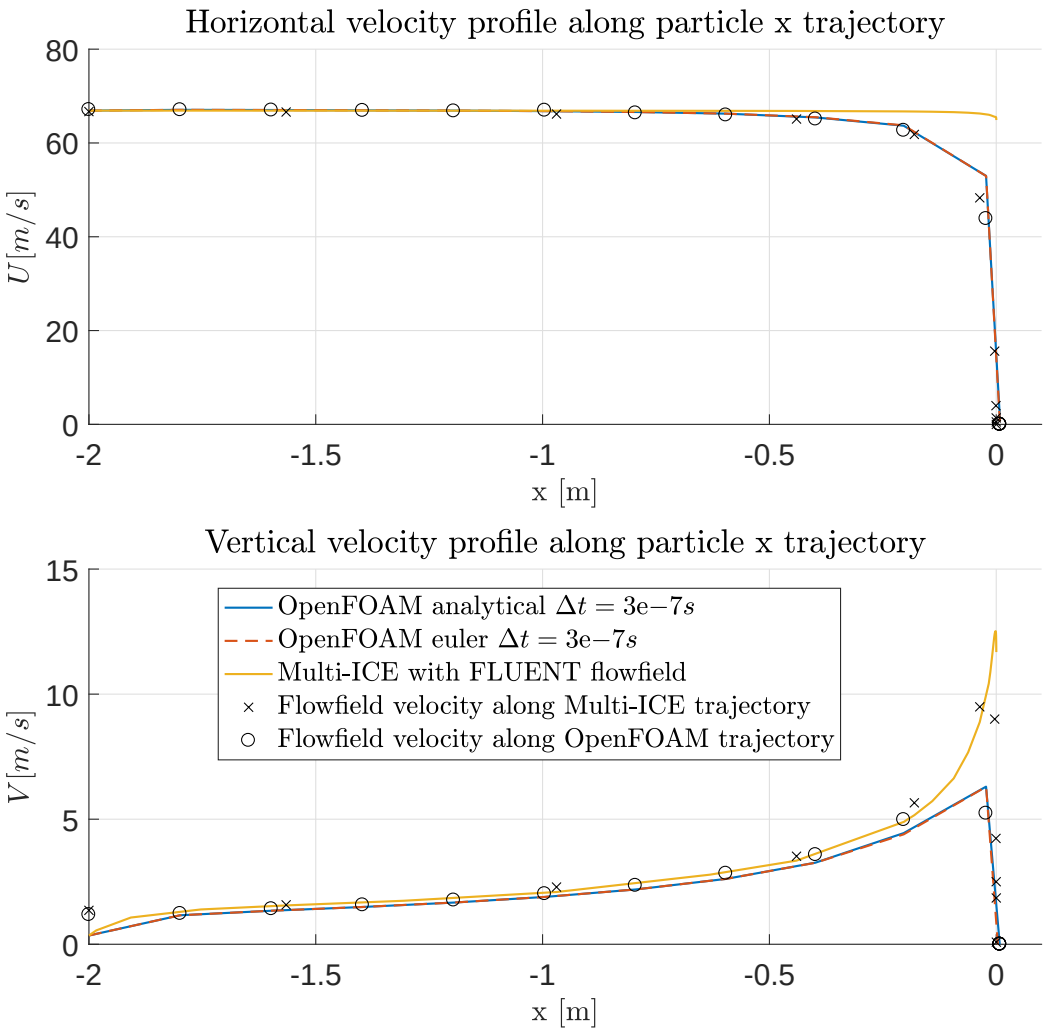


Figure 6.6: Particle trajectories

Clearly, it can be seen that the particle in OpenFOAM seems to 'follow' the flow

field velocity. This can be interpreted as follows. Either forces acting are larger than the particle mass, causing the particle to go in the direction of the flow. Or the particle mass is small, which causes the particle to change its velocity quicker and hence following the flow field. While the Multi-ICE particle remains its velocity until impact, the OpenFOAM particle tends to change its velocity until impact quite significantly, meaning it has very small inertia. To get more insight what the order of magnitude where of the particle forces and particle mass, the source code was investigated, and intermediate results, were printed while simulating. For this, the code had to be modified slightly, adding print statements. However, looking at the intermediate results for forces acting on the particle, the order of magnitude seemed to be all right and no 'suspicious' values could be identified.

6.5 Influence of contact behaviour

In the previous section the impingement analysis for a particle that sticks to the surface has been performed. Aside from having particles stick to the surface, it is possible for the particles to rebound or 'escape'. In case of rebound, as the name suggests, the particles bounce off the airfoil. While in the case of 'escape' after impact the particles are removed from the domain. It was observed in the previous section that the stick behaviour, gives quite different results especially very near the airfoil. To see whether or not this is something specific to the stick behaviour, the escape and rebound options were tried. The results, are again for a particle released at $y = -0.083$ and only the velocity components of the particles for the analytic method are shown, since the analytic and Euler are quite near each other as could be seen in the stick case. The resulting velocities along the particle trajectories can be seen in figure 6.7.

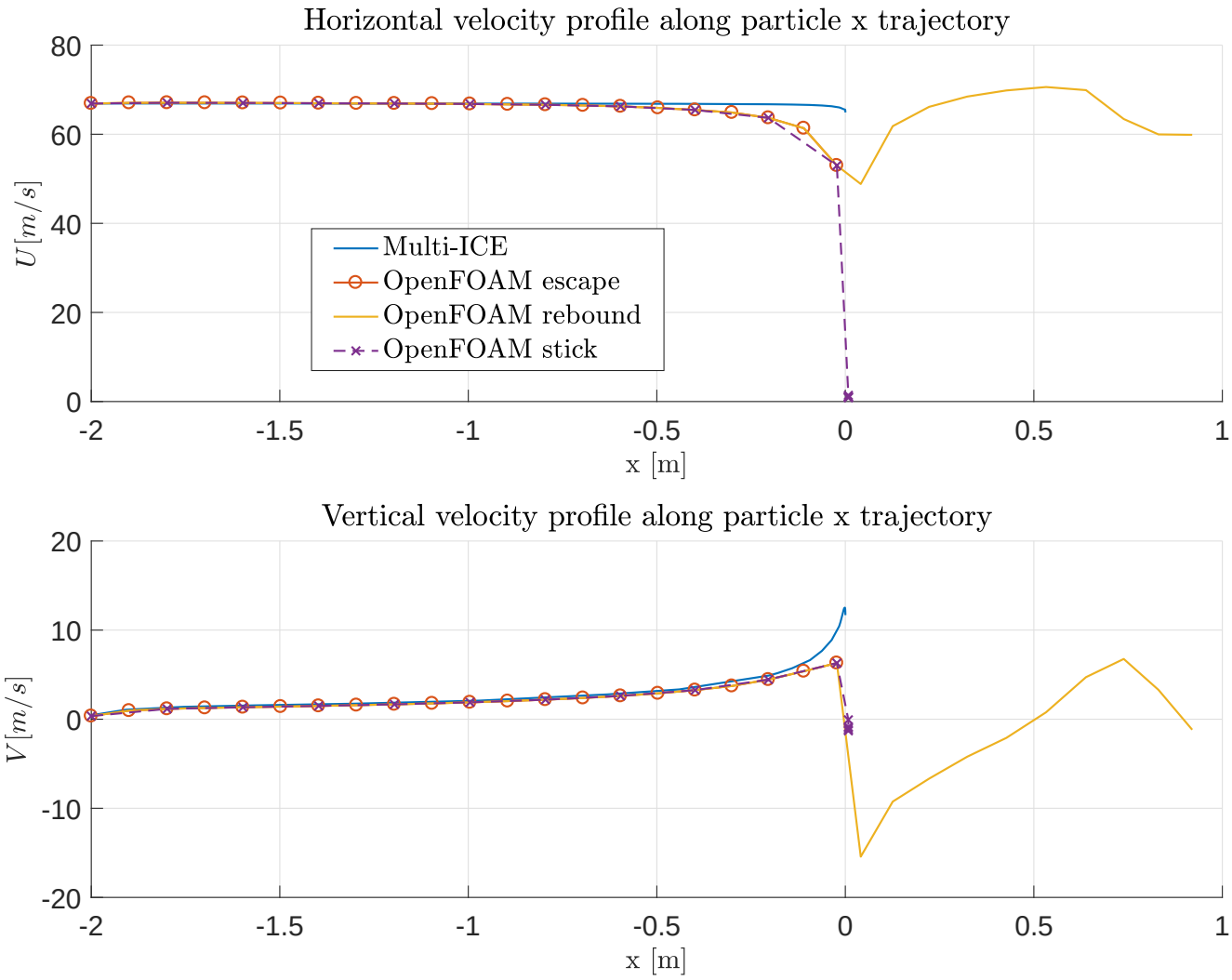


Figure 6.7: Comparison between 'stick', 'rebound' and 'escape' contact behaviour of the OpenFOAM particles.

As can be seen, the rebound and escape contact are quite similar to the stick contact, up until contact, because they still tend to follow the flow velocity more compared to the Multi-ICE particle. It may seem that the escape and rebound have a slightly higher velocity than the stick particle, but this is not the case. The escape and rebound were sampled twice as much, so if the stick particle was also sampled twice as much as the original, it would have the exact same trajectory until impact as the rebound and escape. This is also expected, since the only change is done regarding the contact behaviour. Nevertheless, the intention was to see whether or not, the behaviour of the particle would change just before impact, and there is no difference and still the particles follow the flow field more.

Chapter 7

Particle trajectory calculation in OpenFOAM

In this chapter some details about the Lagrangian particles will be explained. More specifically how *uncoupledKinematicParcelFoam* works. Although, this specific solver is not used in the ice accretion simulation, it may give insight on the particle impingement calculation of the ice accretion solver that was implemented at CIRA as part of *reactingParcelFilmFoam*. The solver *reactingParcelFilmFoam* is one of the standard solvers available in OpenFOAM. It creates a thin film around the object and simulates this film, which also includes chemical reactions and of course it contains particle injection. Without going into detail of this model, since it was not used and time was limited the focus will be on the particle impingement calculations. As was shown in the previous chapter, quite significant changes could be seen between the particle impingement of the particles in Multi-ICE and OpenFOAM. This could be seen especially by the differences in the collection coefficient. The collection coefficient, measures how much mass of water will hit the surface, therefore it is important to know this very accurately, because the ice accretion will take place after droplets hit the surface. And in case of OpenFOAM fewer particles hit the surface, because the icing limits are smaller, and hence the predicted mass accreting on the surface may be too small.

The particle impingement calculation of *reactingParcelFilmFoam*, belongs to the same type of calculation done in *uncoupledKinematicParcelFoam*. Hence, looking at *uncoupledKineMaticParcelFoam* will give insight of *reactingParcelFilmFoam* particles, since the particle calculations are the same in principle.

7.1 Relevant files

The relevant files can be found by going down from the highest level, which is the solver available under `WM_PROJECT_DIR/applications/solvers/lagrangian/uncoupledKinematicParcelFoam/uncoupledKinematicParcelFoam.C`. This function calls `kinematicCloud.evolve();` from the file `WM_PROJECT_DIR/src/lagrangian/intermediate/clouds/Templates/KinematicCloud/KinematicCloud.C`. Then `.evolve()` calls inside its own file another function named `solve()`, which calls `evolveCloud()`. Then finally `evolveCloud()` calls a function `move()` which can be found in `WM_PROJECT_DIR/src/lagrangian/intermediate/parcels/Templates/KinematicParcel/KinematicParcel.C`. Then `move()` calls `calc()`, which finally calls `calcVelocity()`, which are in the same file.

In `calcVelocity()` the particle force calculation, and subsequent time integration is performed, see Listing 1.

```
...
    // Momentum source due to particle forces
    const parcelType& p = static_cast<const parcelType&>(*this);
    const forceSuSp Fcp = forces.calcCoupled(p, dt, mass, Re, mu);
    const forceSuSp Fncp = forces.calcNonCoupled(p, dt, mass, Re, mu);
    const forceSuSp Feff = Fcp + Fncp;
    const scalar massEff = forces.massEff(p, mass);


    // New particle velocity
    //~~~~~~~~~~~~~~~~~~~~~~

    // Update velocity - treat as 3-D
    const vector abp = (Feff.Sp()*Uc_ + (Feff.Su() + Su))/massEff;
    const scalar bp = Feff.Sp()/massEff;

    Spu = dt*Feff.Sp();

    IntegrationScheme<vector>::integrationResult Ures =
        td.cloud().UIntegrator().integrate(U_, dt, abp, bp);
...
```

Listing 1: part of `calcVelocity()` in `KinematicParcel.C`

For each of the included forces, the code evaluates the force contribution, however there are two categories of forces. One type are the 'coupled' forces, and for example the drag force is considered as a 'coupled' force. The other type is 'non-coupled' and these are the gravity and buoyancy, which are taken into account simultaneously, as will be shown later. Another possible option is that an effective

mass is computed, however this is not done for the forces mentioned and the effective mass is the same as the actual mass of a spherical particle times the density.

Then two variables are computed named `abp` and `bp`, which are finally passed together with the time step to the `integrate()` function, which resides in two other files, depending on the choice of integration method.

7.2 Coupled and non-coupled forces

First it is noted that the coupled and non-coupled only refer to whether or not the forces are depending on the local flow field. So although, no actual coupling would be present between the fluid and particles, the sphere force is named a coupled force, because of its dependency on the flow field. Further the coupled forces are not directly calculated rather their implicit force coefficient is calculated, denoted by Sp. The non-coupled forces are calculated explicitly and hence are denoted by the explicit force contribution Su. The total force acting is then denoted as follows:

$$F = Sp(U_f - U_p) + Su, \tag{7.1}$$

with U_f the fluid velocity and U_p the particle velocity.

7.2.1 Coupled force

The coupled force can be found in `WM_PROJECT_DIR/src/lagrangian/intermediate/submodels/Kinematic/ParticleForces/Drag/SphereDrag/SphereDragForce.C`.

```
...
    if (Re > 1000.0)
    {
        return 0.424*Re;
    }
    else
    {
        return 24.0*(1.0 + 1.0/6.0*pow(Re, 2.0/3.0));
    }
}
...
    value.Sp() = mass*0.75*muc*CdRe(Re)/(p.rho()*sqr(p.d()));
...
```

Listing 2: part of the sphere drag calculation in `SphereDragForce.C`

First the drag coefficient is computed, based on the Reynolds number, then the corresponding implicit force coefficient Sp is calculated. The definition might not be

clear at first sight, but it can be very easily derived as follows. Consider the drag force on an sphere dispensed in a moving fluid in general, it is written as follows:

$$F_D = \frac{1}{2}\rho_f (U_f - U_p)^2 C_D A_{ref}. \tag{7.2}$$

In this equation, ρ_f denotes the fluid density, C_D the drag coefficient and A_{ref} the reference area, which is the frontal area of a sphere, hence $A_{ref} = \frac{1}{4}\pi d^2$. Rewriting 7.2, gives

$$F_D = \frac{1}{8}\rho_f (U_f - U_p) C_D \pi d^2 (U_f - U_p). \tag{7.3}$$

Next, the equation is multiplied by $\frac{\mu_f}{\mu_f}$, with μ_f the fluid viscosity, which gives:

$$F_D = \frac{1}{8}\underbrace{\frac{\rho_f (U_f - U_p) d}{\mu_f}}_{=Re}\mu_f C_D \pi d (U_f - U_p). \tag{7.4}$$

Finally, the equation is multiplied with $\frac{m_p}{m_p}$, with $m_p = \frac{1}{6}\pi d^3 \rho_p$, which gives:

$$F_D = m_p \frac{6}{\pi d^3 \rho_p}\frac{1}{8} C_D Re \mu_f \pi d (U_f - U_p). \tag{7.5}$$

Slightly rewriting the terms gives the shape similar to that in the source code, see Listing 2.

$$F_D = \underbrace{m_p \frac{3}{4}\mu_f C_D Re \frac{1}{\rho_p d^2}}_{=Sp}(U_f - U_p). \tag{7.6}$$

7.2.2 Non-coupled forces

The non-coupled forces (gravity and buoyancy) are included simultaneously in `WM_PROJECT_DIR/src/lagrangian/intermediate/submodels/Kinematic/ParticleForces/Gravity/GravityForce.C`

```
...
    value.Su() = mass*g_*(1.0 - p.rhoc()/p.rho());
...
```

Listing 3: part of the gravity and buoyancy force calculation in `GravityForce.C`

The derivation is straightforward, consider the gravity force and buoyancy force on a spherical particle:

$$F_{g+b} = F_g - F_b = \frac{4}{3}\pi R^3 \rho_p g - \frac{4}{3}\pi R^3 \rho_f g, \tag{7.7}$$

in which $g = -9.81$ denotes the gravitational acceleration, upon inserting g the sign of the left gravity contribution will be negative (downwards) and the buoyancy contribution (positive) upwards. Rewriting the equation yields

$$F_{g+b} = \frac{4}{3}\pi R^3 g(\rho_p - \rho_f) = \underbrace{\frac{4}{3}\pi R^3 \rho_p}_{m_p} g(1 - \frac{\rho_f}{\rho_p}) = Su \tag{7.8}$$

7.3 Time integration

For the time integration, two variables named `abp` and `bp` are computed, see Listing 1 from the coupled and non-coupled forces as follows:

$$\texttt{abp} = \frac{SpU_f + Su}{m_p} \tag{7.9}$$

and

$$\texttt{bp} = \frac{Sp}{m_p}. \tag{7.10}$$

Considering the equation of motion 2.1:

$$m_p \frac{dU_p}{dt} = F \tag{7.11}$$

and substituing 7.1 for F gives

$$m_p \frac{dU_p}{dt} = Sp(U_f - U_p) + Su = SpU_f + Su - SpU_p. \tag{7.12}$$

Finally, dividing by the mass left and right, gives the final form of the integrated equation:

$$\frac{dU_p}{dt} = \underbrace{\frac{SpU_f + Su}{m_p}}_{\texttt{abp}} - \underbrace{\frac{Sp}{m_p}}_{\texttt{bp}} U_p. \tag{7.13}$$

Rewriting, yields the first order non-homogeneous ordinary differential equation, with assumed constants `abp` and `bp`:

$$\frac{dU_p}{dt} + \texttt{bp}U_p = \texttt{abp}. \tag{7.14}$$

The general analytic solution of this equation is

$$U_p = \frac{\texttt{abp}}{\texttt{bp}} + Ce^{-\texttt{bp}t} \tag{7.15}$$

and the constant is determined from the fact that if $t = 0$ the particle should still have its same velocity, so we get

$$U_p = \frac{\texttt{abp}}{\texttt{bp}} + C = U_p \rightarrow C = U_p - \frac{\texttt{abp}}{\texttt{bp}}. \tag{7.16}$$

Substituing this in the general solution and rewriting yields, with $\alpha = \frac{\texttt{abp}}{\texttt{bp}}$ and $\beta = \texttt{bp}$

$$U_p = \alpha + (U_p - \alpha)e^{-\beta t} \tag{7.17}$$

Comparing 7.17 with the source code for the analytical integration scheme `WM_PROJECT_DIR/src/lagrangian/intermediate/IntegrationScheme/Analytical/Analytical.C`

```
...
    const scalar expTerm = exp(min(50, -beta*dt));

    if (beta > ROOTVSMALL)
    {
        const Type alpha = alphaBeta/beta;
        retValue.average() = alpha + (phi - alpha)*(1 - expTerm)/(beta*dt);
        retValue.value() =  alpha + (phi - alpha)*expTerm;
    }
    else
    {
        retValue.average() = phi;
        retValue.value() =  phi;
    }
...
```

Listing 4: part of the analytical integration scheme in `Analytical.C`

reveals that the same expression appears in the scheme, with `phi` corresponding to the quantity being integrated, which is correctly U_p in 7.17 and for t the local time step `dt` is passed. It is the local time step, because the time-step is not equal to the time step defined in the `controlDict` file. It is not clear yet, how this is calculated and due to time limitation this was not further investigated.

7.3.1 Euler integration method

Aside from the analytic integration method, also an Euler integration method is available and can be found in `WM_PROJECT_DIR/src/lagrangian/intermediate/IntegrationScheme/Euler/Euler.C`

```
...
    retValue.value() = (phi + alphaBeta*dt)/(1.0 + beta*dt);
    retValue.average() = 0.5*(phi + retValue.value());

    return retValue;
...
```

Listing 5: part of the Euler integration scheme in `Euler.C`

Considering the original differential equation 7.14, it is rewritten as

$$\frac{dU_p}{dt} = \texttt{abp} - \texttt{bp}U_p, \tag{7.18}$$

the Euler approximation of the derivative is

$$U_p^{n+1} = \texttt{abp}\Delta t - \texttt{bp}U_p^n\Delta t + U_p^n, \tag{7.19}$$

with the superscript $n+1$ denoting the value at the next time step and n the value at current time step. Rewriting, it slightly gives:

$$U_p^{n+1} = \texttt{abp}\Delta t + (1 - \texttt{bp}\Delta t)U_p^n, \tag{7.20}$$

and multiplying both terms by $\frac{(1+\texttt{bp}\Delta t)}{(1+\texttt{bp}\Delta t)}$ gives

$$U_p^{n+1} = \frac{\texttt{abp}\Delta t + \texttt{abpbp}\Delta t^2 + (1 + \texttt{bp}^2\Delta t^2)U_p^n}{(1 + \texttt{bp}\Delta t)}. \tag{7.21}$$

The squared terms (Δt^2) are very small since, the time step is also very small, hence they can be canceled and the final form of the equation, is that as shown in Listing 5, with `alphaBeta` referring to `abp` and `beta` referring to `bp`

$$U_p^{n+1} = \frac{\texttt{abp}\Delta t + U_p^n}{(1 + \texttt{bp}\Delta t)}. \tag{7.22}$$

Chapter 8

Conclusions and recommendations

8.1 Conclusions

The usage of different programs such as Multi-ICE, FLUENT and OpenFOAM have been shown. For FLUENT and OpenFOAM also a thorough explanation of the process of setting up and doing a simulation has been shown.

An comparison parameter has been proposed. Which can be used to compare experimental iced shape with numerically predicted iced shapes.

Using the available tools, a test case has been performed of available measurement data in the literature. The results using a turbulent flow field as input for Multi-ICE gave better results compared with the potential panel method in Multi-ICE, in the best case a relative difference of 19% for both $k-\omega$ and SST turbulence models, according to the comparison parameter.

The flow field showed quite good accordance between FLUENT and OpenFOAM. However, as these are completely different codes, some differences could be observed in the velocity profiles. This could be considered as the main reason why the impingements limits were different to start with. To see if this was the case a slower vertical velocity field was set up in FLUENT, and the influence was not significant, hence concluding the difference in impingement limits, is not caused by the flow field velocity differences.

An initial attempt has been made to explain the particle velocity calculation in OpenFOAM, which should provide some starting point to look further into what causes the differences between OpenFOAM and Multi-ICE.

8.2 Recommendations

To improve the research regarding the particle impingement calculation, it would be better to use some more simple geometry and flow field. In this way, it would be

possible to minimize other sources of errors. However, initially the goal of this report was not to find the error that caused the different in impingement limits. Therefore, only a start to what might cause this issue was possible to do. The comparison between Multi-ICE and OpenFOAM was not possible to do fully, since no ice accretion was simulated in OpenFOAM. Because of the significant differences in icing limits, it was known before even doing the simulations, that the results of OpenFOAM would be far from that of Multi-ICE. This is also the reason, why towards the end of the report the focus lied on particle impingement calculation.

To understand the particle impingement and the reason what causes the differences in OpenFOAM, the source code has to be researched in more detail, if the goal is to have a reliable code that predicts the ice accretion in OpenFOAM. One of the things that might possibly, cause 'problems', could be the time step. The time step is defined globally in the dictionary file. However, it was observed that this time step was not passed through the integration of the velocity. However a smaller time step, which is not determined yet how it is calculated, was passed in the integration step. This should in principle not be a problem, since usually in numerical integration the time step limitation is a maximum time step that should not be exceeded. But then the problem might be that the time step defined in the dictionary file, was perhaps not sufficiently small, especially near the airfoil, where the velocity changes very rapidly. In Multi-ICE an adaptive time stepping method is implemented to overcome this issue. However, setting the time step to a very small value in OpenFOAM would take a very long time to perform the integration, in case of a time step of $3e-8s$, the simulation would run for three hours already, to integrate a time of $0.045s$. All in all, it would be better to have some possibility to do an adaptive time stepping method in OpenFOAM. Also researching how the time stepping is performed would help to understand what might cause the problem.

Bibliography

[1] W. B. Wright, “Validation methods and results for a two-dimensional ice accretion code,” *Journal of Aircraft*, vol. 36, no. 5, pp. 827–835, 1999.

[2] J. M. Hospers, “Eulerian method for super-cooled large-droplet ice-accretion on aircraft wings.” Ph.D. dissertation, University of Twente, 2013.

[3] B. L. Messinger, “Equilibrium temperature of an unheated icing surface as a function of air speed,” *Journal of The Aeronautical Sciences*, pp. 29–42, Jan. 1952.

[4] S. Thomas, R. Cassoni, and C. MacArthur, “Aircraft anti-icing and de-icing techniques and modeling,” *IEEE/OSA Journal of Lightwave Technology*, vol. 28, no. 1, pp. 3–18, Jan. 2010.

[5] G. A. Ruff, “Quantitative comparison of ice accretion shapes on airfoils,” *Journal of Aircraft*, vol. 39, no. 3, pp. 418–426, 2002.

[6] H. E. Addy, *Ice accretions and icing effects for modern airfoils [microform] / Harold E. Addy, Jr.* National Aeronautics and Space Administration, Glenn Research Center ; National Technical Information Service, distributor [Cleveland, Ohio] : [Springfield, Va, 2000.

[7] A. Rohatgi, “Webplotdigitizer,” Oct, 2017. [Online]. Available: http://arohatgi.info/WebPlotDigitizer

[8] M. Papadakis, S.-C. Wong, and A. Rachman, *Large and Small Droplet Impingement Data on Airfoil and Two Simulated Ice Shapes*. National Aeronautics and Space Administration, Glenn Research Center ; National Technical Information Service, distributor [Cleveland, Ohio] : [Springfield, Va, 2007.

Appendix A

Explanation of Meshing and Simulation in Ansys

In Chapter 5 an overview of the steps that were taken to make a computation in FLUENT was given. Since, the steps are not necessarily important to understand how the work flow was, it was decided to explain the intermediate steps in this appendix.

A.1 Making a mesh in ICEM

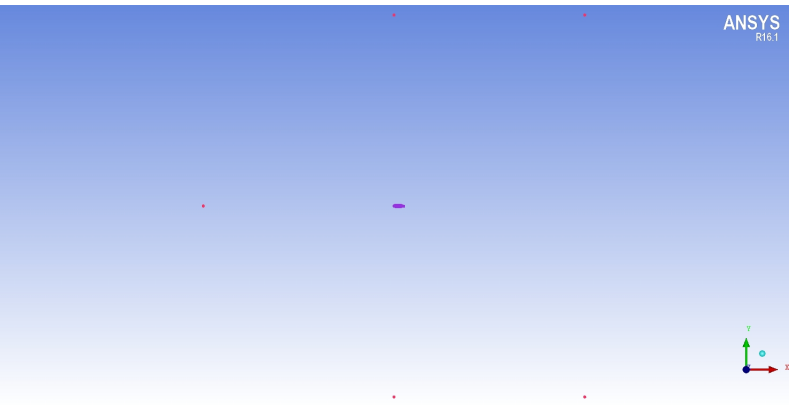
The first step in any CFD computation is to create a mesh of the domain to be analyzed. In this report we are interested in the flow around an airfoil, hence the mesh will be made around the airfoil.

The first step is to import the points in the airfoil, such as given in Appendix B, this is done by going to **File, Import Geometry, Formatted point data**, and selecting a file which has these points written into them, however additionally for each point the z coordinate must be specified and hence a column with zeros have to be added. This can be easily done by using an Excel Worksheet for example.

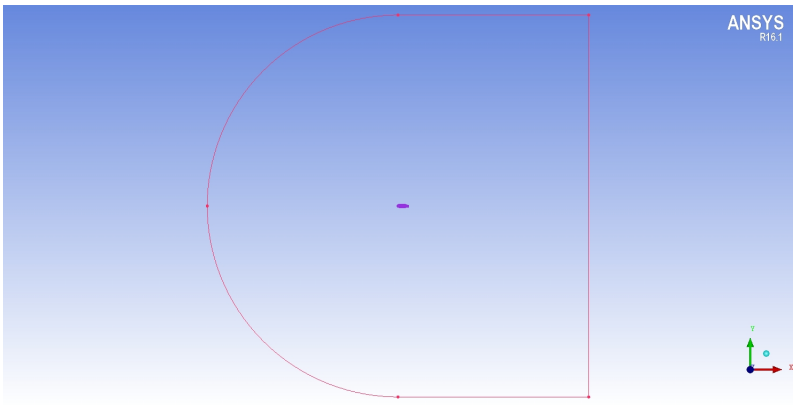
The points then have to be connected, this is done by going to the **Geometry** tab and then to **Create/Modify Curve, From Points**. Note that before connecting the points, it is convenient to change the part name to 'Airfoil' for example, this will be useful for later. Also it might not be possible to connect all points in one try, because the number of points to be connected per curve is limited. With the coordinates provided in this report, first a curve going from the trailing edge to the leading edge for the top of the airfoil and then from the leading edge to the trailing for the bottom of the airfoil was made.

Now the the farfield has to be made, a common approach is to have a so-called C-type grid, which is very commonly used for airfoil, because its shape allows the

grid lines to be orthogonal to the airfoil surface in a very easy way. By going to **Create Point, Explicit Coordinates** the following points are added in the plane $(z = 0)$: $(-20c, 0)$,$(0, 20c)$,$(0, -20c)$,$(20c, 20c)$ and $(20c, -20c)$, where c represents the chord length of the airfoil. The point in the bottom, right and top are connected with straight lines, using **Create/Modify Curve, From Points** inside a new part name called farfield. The top, left and bottom are connected using **Create/Modify Curve, Arc**, this is also added to farfield. Finally, a surface will be created from the farfield just created, using the **Create/Modify Surfaces, Simple surfaces** inside a new part called fluid. The lines from the farfield have to be selected and from this the surface is created. The result can be seen in figure A.1.



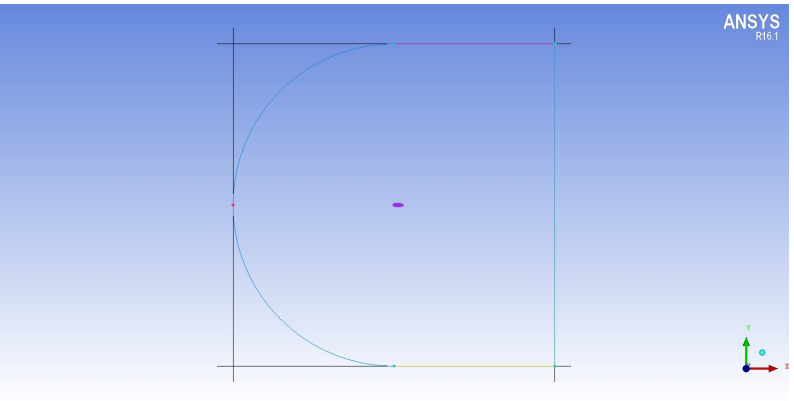
(a) Step 1: Import airfoil coordinates and create farfield points



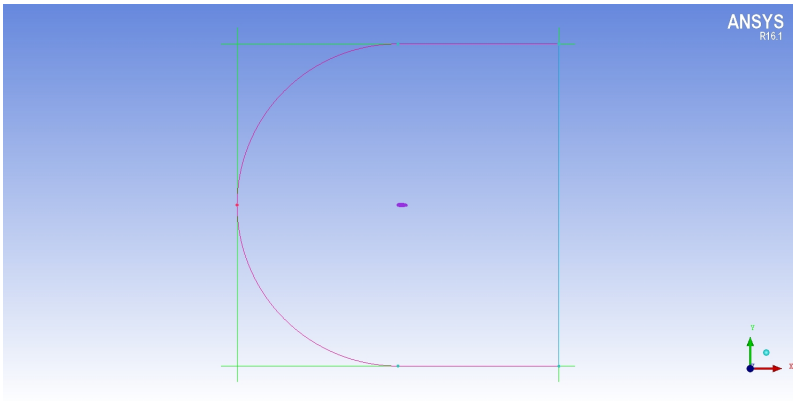
(b) Step 2: Create curves and lines for the airfoil and farfield

Figure A.1: Making the curves for the farfield and airfoil

Next by going into the **Blocking** tab, an initial block will be made by selecting **Create block**, here the part fluid must be selected, and under the **Initialize Block** 2d Planar must be selected. Now the farfield curves will be associated with the block by going to **Associate, Associate Edge to Curve**. The right curve of the farfield is associated to the right edge of the block. Then by selecting the top, left and bottom of the initial block simultaneously, these are associated to the top and bottom straight curves and the arc on the left, see figure A.2.



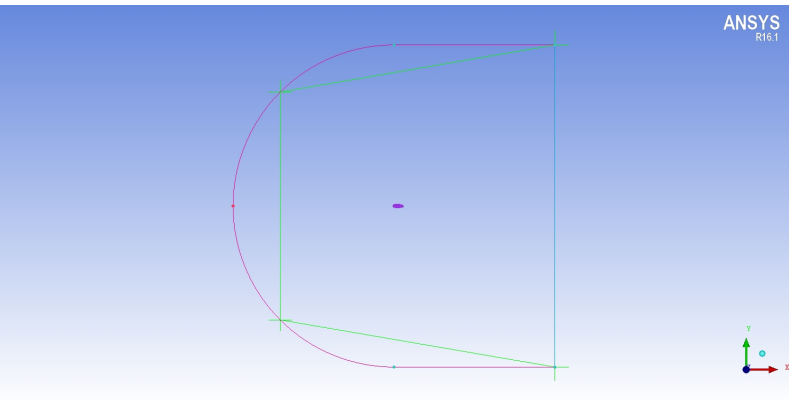
(a) Step 3: Create an initial block from the fluid surface



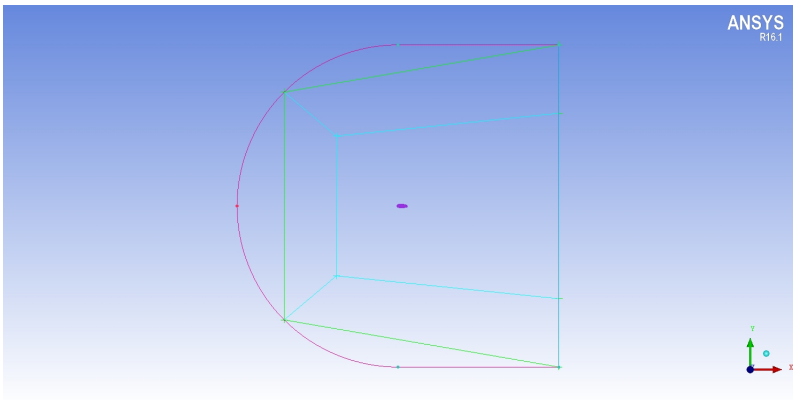
(b) Step 4: Edge(block) to Curve(geometry) association

Figure A.2: Creating an initial block

Next within **Associate** the **Snap Project Vertices** is applied. To create a C-type grid the block has to be split, this is done by going to **Split Block, Ogrid Block**. Then the block in the centers is selected using **Select Block(s)** and without clicking apply yet, the right edge is selected using **Select Edge(s)**. The result should be as shown in figure A.3.



(a) Step 5: Snap project vertices onto the created curves



(b) Step 6: Making a C-type grid

Figure A.3: Creating an initial block

Finally, another split line is added to the right of the trailing edge by using first option **Select Block**, in **Split Block**. The line above the airfoil must be clicked and by clicking and dragging the position of the split can be set behind the airfoil, see figure A.4.

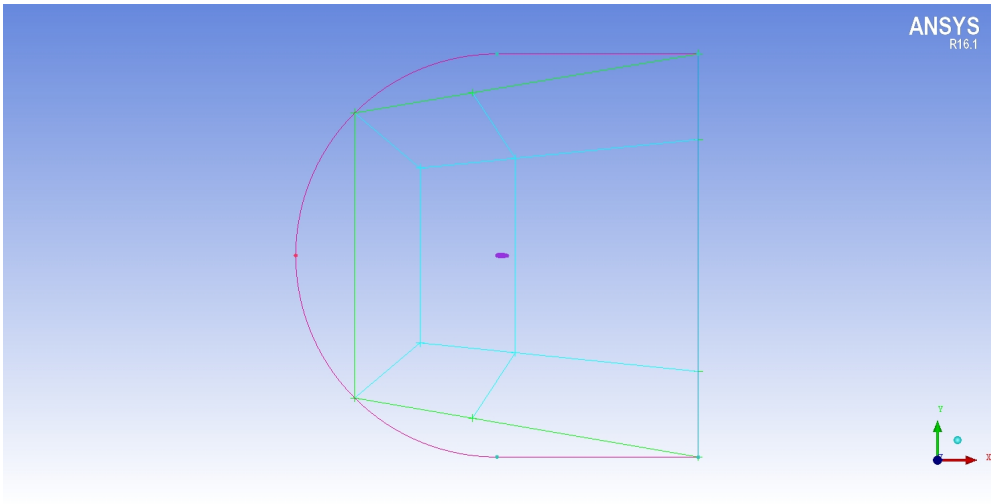
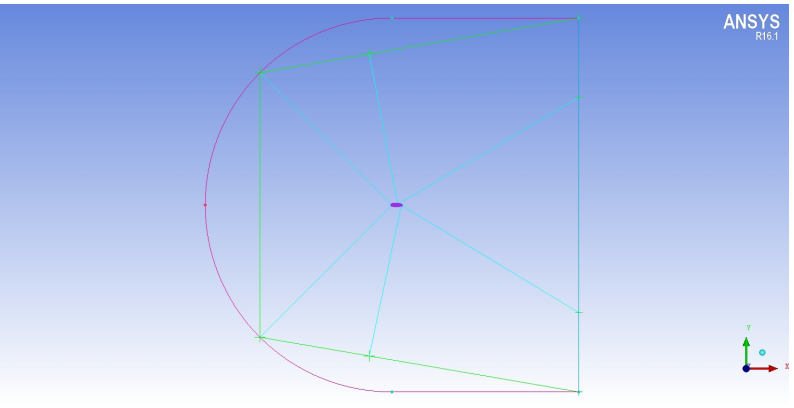
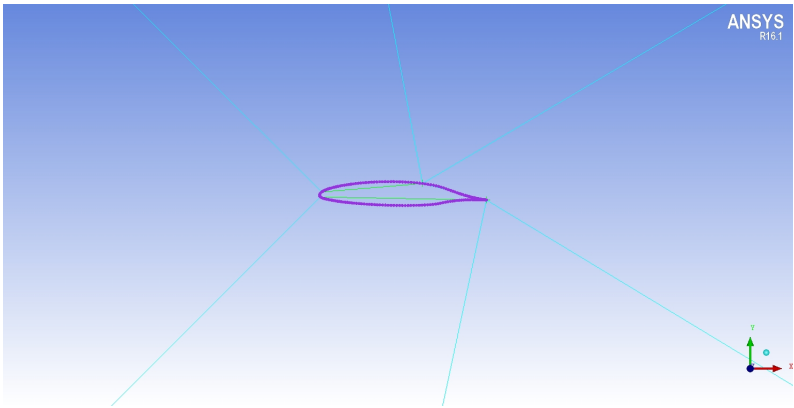


Figure A.4: Step 7: Adding a split line behind the trailing edge

Now that the C-type blocks have been created, the next step is to associate the edges of the blocks to the corresponding lines on the airfoil. The edge above the airfoil is associated to the top part of the airfoil, using **Associate, Associate Edge to Curve** and using the Project vertices option, to visualize the result. Similarly, this is repeated for the bottom part of the airfoil, but now the bottom edge is associated to the bottom of the airfoil, see figure A.5.



(a) Step 8: Associating the newly created edges



(b) Step 9: Close-up of current situation

Figure A.5: Edge association of the newly created edges, and a close-up.

Before continuing at from this point, it is a good idea to set some initial values of the mesh, this is done by going to the **Mesh** tab. Here we use **Part Mesh Setup**. In this menu we set the maximum size for the airfoil, for example to a value of 1/5 of the chord and the fluid is set to the value of the chord. Note that this step has to be done, and if not it may cause problems in the following steps. Next by going back to **Blocking** tab, and updating the mesh using **Pre-Mesh Params**, we get to see the

current mesh, as seen in figure A.6.

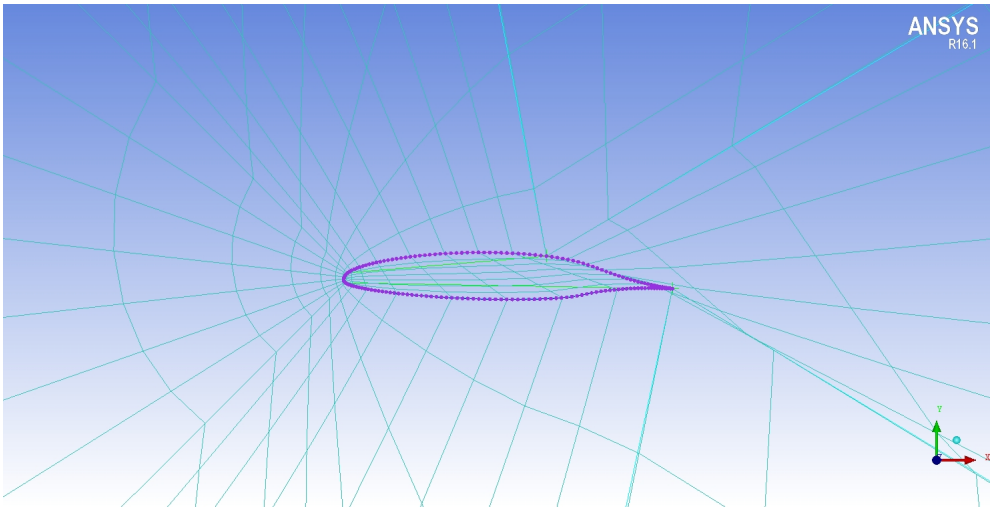


Figure A.6: Step 10: Setting initial mesh values

As can be seen there are cells inside the airfoil, these must be removed by going into **Delete Block** and selecting the block inside the airfoil. The mesh will distorted, but by going into **Pre-Mesh Params** after each time the mesh is edited, the current modification can be viewed, and the result should be that there are no distorted elements, figure A.7

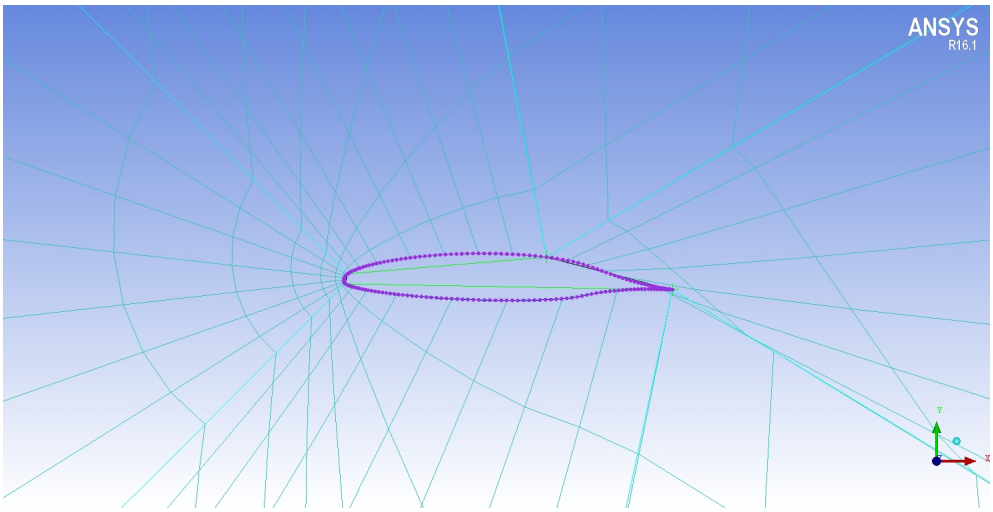
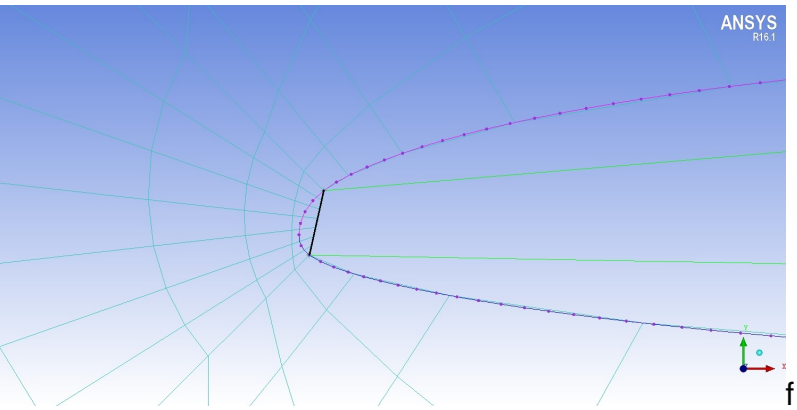
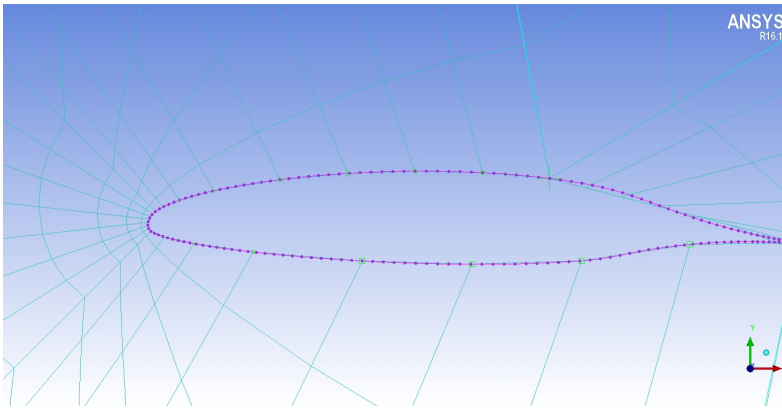


Figure A.7: Step 11: Deleting the block inside the airfoil

Now there should be small piece of a block, not associated yet to the airfoil. This part, seen as black line in figure A.8, is associated to both the top and bottom part of the airfoil, see figure A.8.



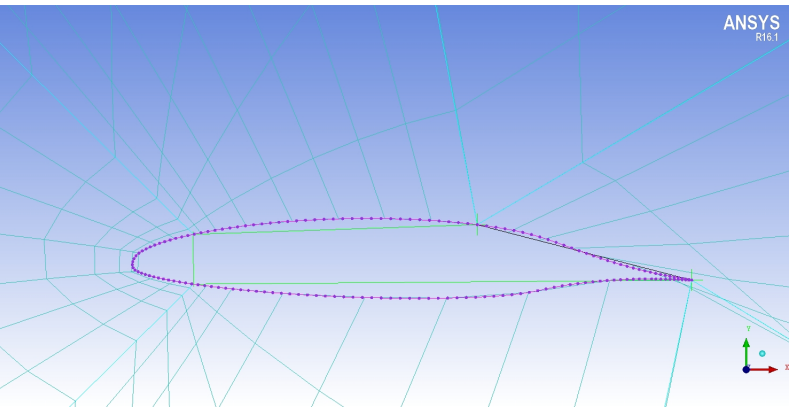
(a) Step 12: Front edge association



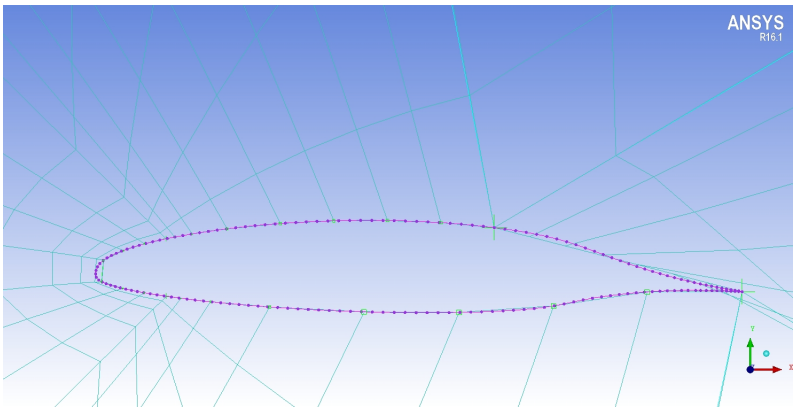
(b) Step 13: Close-up after association

Figure A.8: Edge association of the front edge in the airfoil, to the top and bottom curve of the airfoil geometry

Next, the block edges, which are currently inside the airfoil have to be fitted to the surface of the airfoil by going into **Edit Edge, Split Edge** and selecting Automatic Linear method, and clicking the edges in the airfoil one by one, see figure A.9.



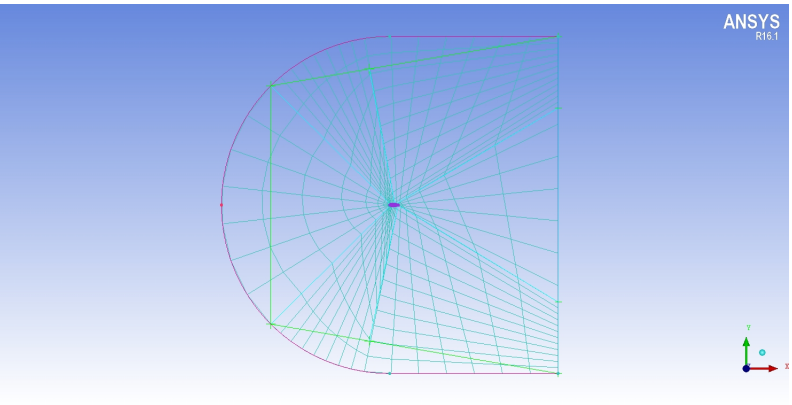
(a) Step 14: Unfitted block edges



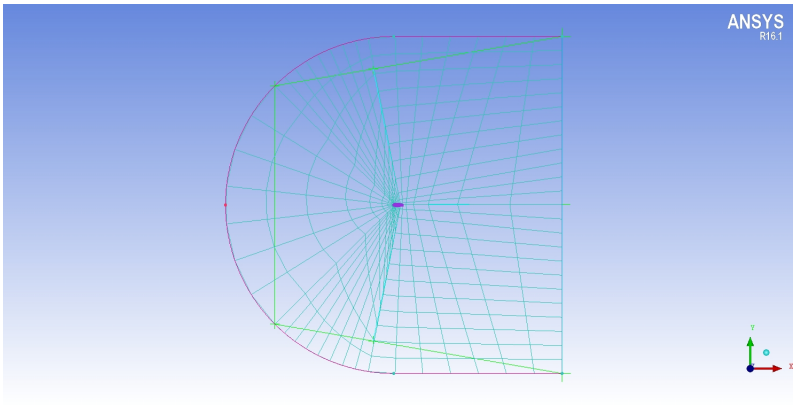
(b) Step 15: Fitting the block edges onto the airfoil geometry

Figure A.9: Linear fitting of the edges of the blocks to the airfoil geometry

For more easy control of the mesh, the fishtail behind the trailing edge is collapsed next. This is done by going into **Merge Vertices, Collapse Blocks** and selecting the right edge in the farfield and then the block to the right of the trailing edge, see figure A.10.



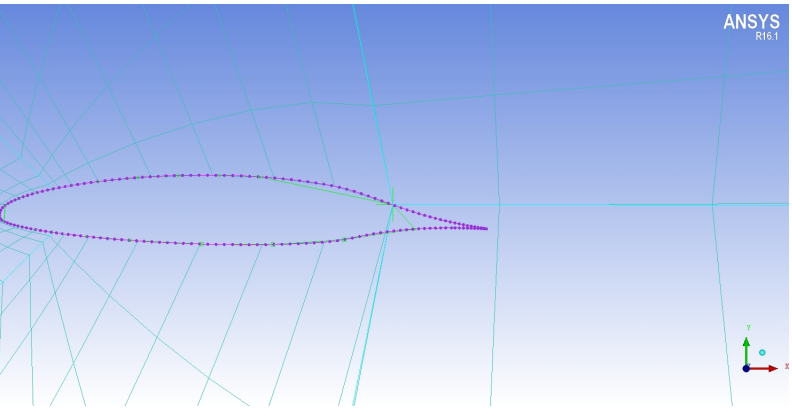
(a) Step 16: Uncollapsed block



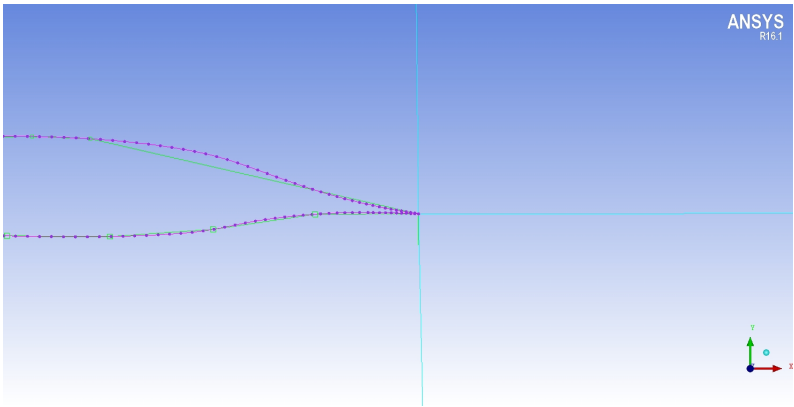
(b) Step 17: Collapsing the block behind the trailing edge

Figure A.10: Collapsing the right block

Finally, the vertex of the trailing edge, is moved to the trailing edge using the **Move Vertices, Move vertex** tool, and dragging it to the trailing edge. The result is shown in figure A.11.



(a) Step 18: Trailing edge vertex before moving



(b) Step 19: Moving the trailing edge vertex manually to the trailing edge

Figure A.11: Movement of the trailing edge vertex of the block, to the actual geometric location of the trailing edge.

Finally, the mesh spacing parameters will be set per edge of the blocks that were created. The mesh spacing near the airfoil (perpendicular to the airfoil) is very important to set correctly, an online tool [1] is used, with an $y^+ = 1$. It is important to select the spacing correctly, because otherwise the boundary layer may not be correctly simulated, which is essential in viscous simulations.

[1] https://geolab.larc.nasa.gov/APPS/YPlus/

Next by going into **Pre-Mesh Params, Edge Params**, the edges perpendicular to the airfoil are configured. By selecting only one, all four can be changed using one of the options inside the tool. By selecting **Exponential2**, we have to set the spacing to the location where the arrow is pointing towards. **Spacing 2** is set to the value computed using the tool, the ratio is set to 1.2, such that there are more points near the airfoil instead of the default value of 2. Finally, the number of points is set to for example 100, this number can be increased of course if more accurate computations are desired or to check convergence for example. Finally, before clicking apply, the **Copy Parameters** options with **To All Parallel Edges** is selected.

Now the edges, that are aligned with the airfoil surface are configured. These edges, do not necessarily need to have a y^+ value as the perpendicular edges, however it is beneficial to have a little bit smaller cells to the front of the leading edge and also towards the trailing edge, as there most of the changes of the flow will occur there.

Now the edge parameters are configured. In this case a **Bigeometric** spacing, by specifying the **spacing1** and **spacing2**, and the number of nodes the edge parameters are set. Similarly is done for the bottom, since now the lines are not parallel anymore, the settings can be copied using **To Selected Edges** inside **Pre-Mesh Params**. Also for the front edge the edge parameters have to be set, it must be noted that the transition from the front block to the top and bottom block should be smooth, the change in element size should not be too significant. For this **Match Edges** is used, which allows to copy the spacing at the start and end of an edge, to the start/end of the target edge.

Finally, the trailing edge, block edge must be configured, again it is important that near the airfoil the spacing is relatively small, hence here again the y^+ value is used, accompanied by a Exponential1 distribution.

Finally, to get some better alignment of the block's edges to the airfoil, the vertices in the farfield may be moved, and using the mentioned options in **Pre-Mesh Params** the mesh can be modified in areas where it required. The resulting mesh is shown in figure A.12.

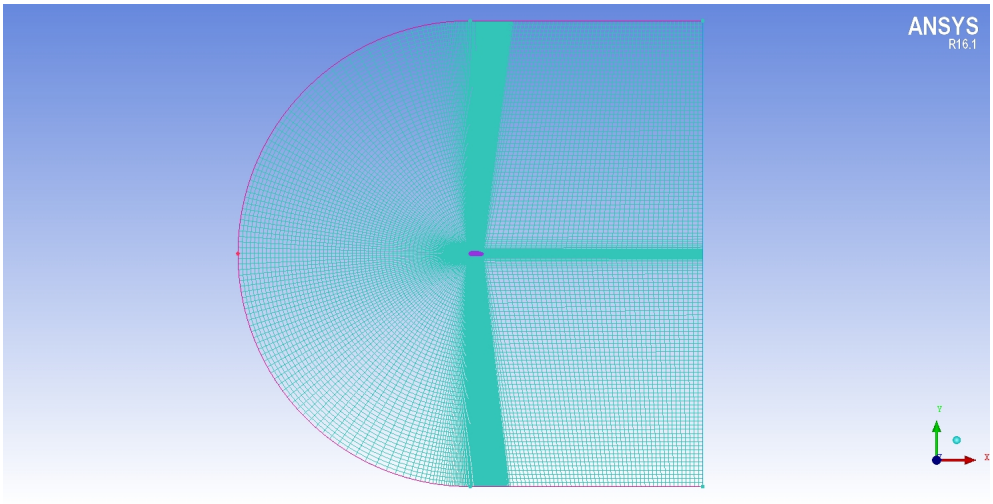


Figure A.12: Step 20: Final Mesh

Before exporting the mesh, the mesh is check using **Pre-Mesh Quality Histograms**, the criterion that will be check is the quality, and the **MinX** value is modified to -1. It is especially important that there are no negative values, which would mean some cells are have their edges overlapping forming an 'X' shaped cell. Furthermore, it is required that most cells have a good quality. Also by going into **Block Checks** the blocks may be checked for problems and fixed.

The actual mesh will now be generated, by right-clicking **Pre-Mesh** underneath **Blocking** and selecting **Convert to Unsctruc Mesh**.

A final mesh check will be performed underneath the **Edit Mesh** tab, by clicking **Check Mesh**. Everything should be okay, except for edges at the farfield, but this should be ignored. The reason this happens is because a 2d planar block was formed, but this will not give a problem in FLUENT.

The final step is to go into **Output Mesh** tab, and clicking **Solver Setup**, and selecting **ANSYS FLUENT**. The boundary conditions will be set up in **Boundary Conditions**. **Underneath Edges**, the airfoil will be shown and a wall boundary condition must be applied. For the farfield a pressure-far-field will be used. Finally, underneath **Mixed,** the fluid will be set to a fluid boundary condition. Clicking **Write Input** the .msh files can be exported, important is to select the 2d option, and if necessary the mesh can also be scaled if desired.

Some details about the created mesh, that was used in all the simulations, will be given. The mesh as shown in figure A.12 contains 45880 quadrilateral cells. The airfoil is divided in 272 wall faces. The total number of nodes is 46325.

A.2 Making a 2d simulation of the mesh generated in ICEM

After creating the computational mesh, the goal is to perform simulations with this this grid. In this section it is described how the simulation was set up in FLUENT.

First the mesh read by going to **File, Read, Mesh**. Then by going from top to bottom in the tree on the left side of the screen all parameters are set.

Within **Models, Viscous** the viscous model can be set. It is important to note that for viscous simulations it is required to have the correct y^+ value set, such that the boundary layer is simulated correctly. If a inviscid simulations will be performed, the y^+ value is not required, and a larger first cell height may be used.

In this report, three different viscosity models were used to compare with each other, these are the Spalart-Allmaras, $k-\omega$ and Transition SST model. The default values are used, since it is the purpose to use different models, rather then setting up a well-established turbulence model, which is beyond the scope of this report.

Next, by going to **Materials** in the setup tree, the fluid parameter will be set. Since, the case that is analysed in this report has a low Mach number, it is acceptable to use the ideal gas law. For the viscosity Sutherland's three coefficient viscosity law is used, without changing default values.

Next, the **Boundary conditions** section in the setup tree will be set. Since, in the meshing part most boundary conditions were already set only a few parameters have to be modified. These are within the farfield boundary condition. In the **Momentum** tab, the gauge pressure is left at 0 pressure, while in the **Operating condition** the atmospheric pressure is remained.. The Mach number is 0.21, and to simulate a 0.3 degrees angle of attack the flow has two components, for the x direction this is $\cos(0.3^\circ)$ and for the y direction $\sin(0.3^\circ)$, of which the arguments are in degrees. Finally, under the **Temperature** tab, the temperature is set to 258 K, as was the case in the experiment.

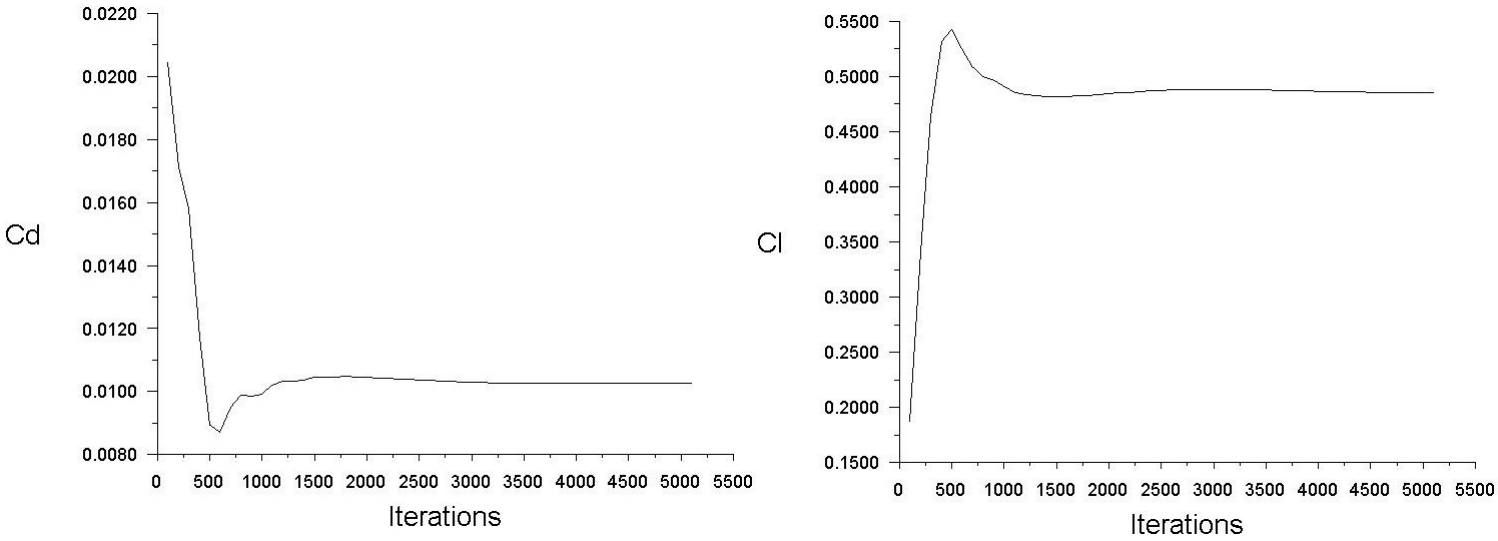
Next, by going into the **Reference value** tree, the reference values are set. Under compute from, the farfield will be selected and most values will be calculated automatically. However, to get a correct reading of the lift and drag coefficients later on, confusingly the 'area' must be changed to the chord length, which is in this case $0.9m$. Finally, at the bottom the reference zone is changed to fluid.

Next under the **Solution** section in the setup tree the schemes that will be used, will be set. In this case, the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) scheme was used. To initialize and get a stable initial result all variables below the pressure variable were set to First Order Upwind for the first 1000 iterations. After that these were changed to Second Order Upwind to get more accurate results.

In the Monitors subtree, all absolute residual criteria were set to $1e-6$, otherwise the simulation would stop prematurely, due to a relatively high residual. To monitor the convergence, rather than looking at the residuals, for 2D airfoil simulations it is more common to look at the lift and drag coefficients. These are set as follows.

By clicking create under Force Monitors, the drag coefficient is set, by clicking the airfoil part under Wall Zones. Then Print to Console, Plot and Write are checked. The force vector points parallel to the flow hence the x component is $\cos(0.3°)$ and the y component $\sin(0.3°)$. Similarly, the same is done for the lift coefficient, however now the force vector points upwards perpendicular to the flow, hence the x component has the value $-\sin(0.3°)$ and the y component the value $\cos(0.3°)$.

Next, under initialization hybrid initialization is selected and applied. Finally, the simulation will be performed. First 1000 steps will be done, using the First Order Upwind scheme, it may be useful to set the reporting interval at a value of 100, in order to reduce simulation time. After 1000 iterations, the variables will be solved using the Second Order Upwind schemes, and a desired number of iterations may be performed, while monitoring the lift and drag coefficient. When the lift and drag coefficient are constant, the simulation is considered to be finished. An example of this can be seen in figure A.13.



(a) Drag coefficient

(b) Lift coefficient

Figure A.13: Lift and drag coefficient monitors of a simulation

As can be seen, after 5000 iterations in this case the values for the lift and drag coefficient are becoming rather stable and convergence is assumed.

A.3 Exporting results

After a result in FLUENT is obtained, the next step can be to further analyze the result, by using post-processing software, such as TECPLOT. For this report, it is important, to export the results in such a way, that Multi-ICE can use the results obtained from FLUENT.

To use precomputed data in Multi-ICE two files are needed. One file containing dimensionless airfoil coordinates, corresponding dimensionless velocities and pressure coefficients. Note that since Multi-ICE only does 2D simulations, the z coordinate and velocity must be zero for all points. The exact data format can be found in the manual.

These values can be extracted from FLUENT as follows. Going to **File, Export, Solution Data**, select ASCII file format, with space as delimiter. Then click fluid and the airfoil part under Surfaces. The variables to select are pressure coefficient, velocity magnitude and the x and y velocities. The ASCII file can be opened using an Excel Worksheet. The points are ordered, so they are not placed in a random order, but the starting point, is usually not at the trailing edge. Hence, the data could have to be shifted, but this can be done very easily within Excel.

Next, the flowfield data must be extracted. As the number of points that can be loaded is limited in Multi-ICE, and also not all points are needed, because for example the flow field after the airfoil is not of interest, since no impingement can happen there. So it is good practice to cut out part of the domain rather than loading in the full domain of results. However, by doing this the topology must be remained, but fortunately a solution has been found to this, which will be explained.

From the obtained FLUENT solution, the fluid domain will be separated into a new piece, by going to **Adapt, Region**. Here an area inside the quad (which is basically a square) will be selected. The minimum and maximum x and y values of the square box are set, in this report minimum x was set to -2, the maximum to 1, the minimum y value to -0.5 and the maximum y value to 0.5. It may be necessary to change these values, if there are too many point near the airfoil, because otherwise Multi-ICE may not be able to load all results. After setting the values, the Mark button is clicked and the window is closed.

Then under **Mesh, Seperate, Cells** select the hexahedron region in the fluid and Seperate. Now by going to **File, Write, Case & Data** the file is saved together with the data.

Next the data is opened using TECPLOT. First by going to **File, Load Data Files**, and then clicking the advanced opening menu, see figure A.14.

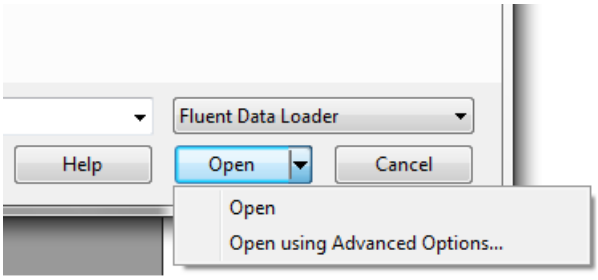


Figure A.14: Advanced opening menu

Then Load Case and Data files is selected, and the saved case and data from FLUENT is loaded, but the Averaging to Nodes arithmetic is used, such that values on the nodes are shown instead of inside the cells. After loading the files, go to **File, Write Data File** and save the file in ASCII format. Here only Field Data is selected, and in the zone, there should be two fluid zones, the one with a number has to be selected only. Then the first 5 variables, as shown in figure A.15 are selected.

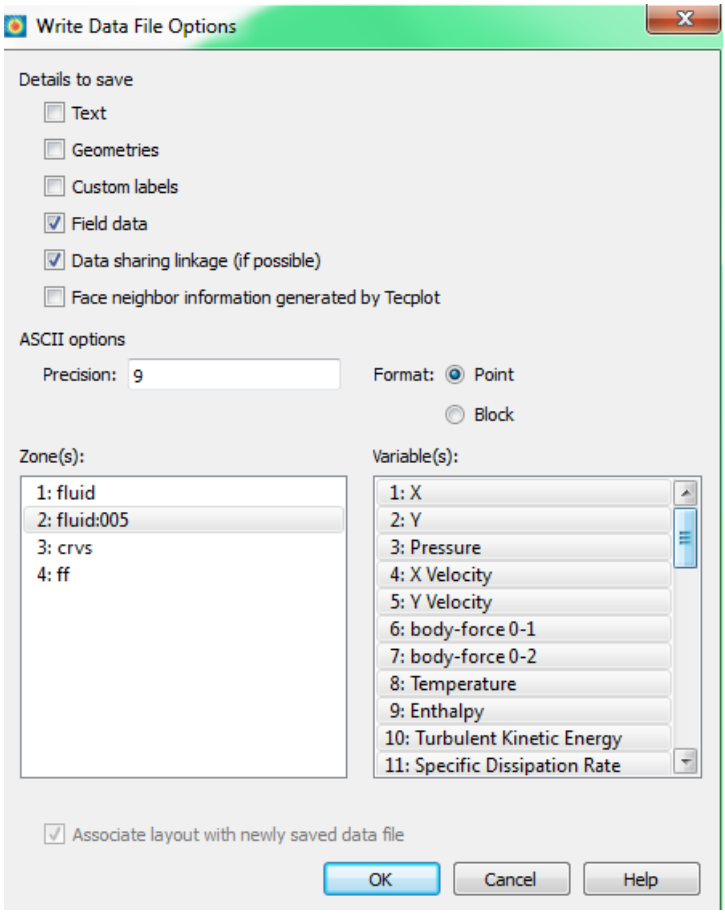


Figure A.15: Writing variables in TECPLOT

The resulting file is an ASCII file, which can be opened using an Excel Worksheet. After the header, the variables that were selected are shown. Below that

data, the connectivity data of the cells is given. For Multi-ICE, the data must be manipulated, since the dimensionless coordinates and velocities have to be specified in the input of Multi-ICE rather than the dimensional values as they are from FLUENT. The files for Multi-ICE should be saved in .dat format, and can then be loaded into Multi-ICE.

Appendix B

Tables

In the report, some experimental data was used to compare with the numerical simulations. Since, the experimental results were not available as a separate file, they were extracted using WebPlotDigitizer [7] The coordinates of the iced shape are shown in B.1. For completeness the coordinates from the airfoil as indicated in [6], are added in a table in B.2. Note that during this work, initially some airfoil coordinates were not exactly the same. This was found out at a late stage of the work, hence it was decided to continue with the slightly wrong coordinates. The corrected coordinates are denoted next to the wrong coordinates inside parentheses. The differences are relatively small and because of the meshing the exact shape will never be attained to the same exact values,due to interpolation between nodes etc. Another aspect is that most point were after the leading edge and hence would be of less influence on the flowfield in front of the airfoil. However, there is one point slightly wrong at the leading edge, but this its effect was hardly visible and very small influence of these result is expected on the overall simulations. One significant is the peak that was observed at the bottom around $x = 0.47869$,(see Figure 4.2), but because this is after the leading it was retained for all simulations done.

B.1 Experimental iced data for case 631

This section contain data of the experimental iced shape, in terms of its coordinates. These value were used to measure the comparison parameter value. The values are the non-dimensional values.

x/c	y/c
0.050304442	0.040261516
0.049042277	0.039813173
0.047206214	0.039207319
0.045366399	0.039531099

x/c	y/c
0.043528942	0.039270538
0.041694165	0.038345952
0.03986105	0.03700967
0.038024558	0.03651006
0.036188762	0.035837803
0.034352645	0.035245229
0.032517215	0.034482665
0.03068218	0.033621826
0.028846975	0.032803483
0.027009786	0.03247652
0.025173562	0.031910507
0.023337981	0.031185128
0.021502454	0.030446469
0.01966773	0.029508602
0.017833918	0.028344967
0.015996783	0.028004723
0.014159916	0.027598076
0.012318439	0.028333552
0.010478732	0.028630771
0.0086409	0.028463173
0.006804301	0.027990124
0.004967487	0.027570197
0.003247707	0.026623358
0.002339387	0.023913719
0.002918229	0.02284636
0.00464163	0.022895816
0.000698995	0.021414177
-0.0007826	0.019255796
-0.002579403	0.01841254
-0.004417771	0.018377747
-0.006256675	0.018475759
-0.008093596	0.018082392
-0.009812534	0.016926861
-0.010626238	0.015177749
-0.011406891	0.013374145
-0.011990578	0.011574458
-0.012425513	0.009523483
-0.01312419	0.007744531

x/c	y/c
-0.014019172	0.005797742
-0.014814073	0.003456773
-0.015397416	0.001572091
-0.016275065	-0.00060229
-0.017063332	-0.002959925
-0.016930777	-0.004553963
-0.016640162	-0.005948261
-0.015421109	-0.007471563
-0.013730678	-0.008743371
-0.011887326	-0.009943664
-0.010156268	-0.011791422
-0.008425769	-0.013500682
-0.007502619	-0.014466043
-0.005978274	-0.015710558
-0.00404819	-0.016256755
-0.002209983	-0.016182121
-0.000809605	-0.017441314
0.000786918	-0.01849283
0.00262743	-0.018989257
0.004465744	-0.018941184
0.00630647	-0.019490732
0.008151805	-0.021182404
0.009993388	-0.02194444
0.011832077	-0.02198933
0.013669623	-0.021750903
0.01550951	-0.022092391
0.017349325	-0.022416171
0.019189193	-0.022753232
0.021029222	-0.023130135
0.022870323	-0.023772647
0.024710513	-0.024189391
0.026547273	-0.023756183
0.028383604	-0.023216731
0.030224009	-0.023686597
0.032068861	-0.025258744
0.033912106	-0.026432476
0.035752189	-0.026822659
0.037590128	-0.026681622

x/c	y/c
0.039426245	-0.026089048
0.0411529	-0.026845908
0.042881317	-0.028039126
0.044720703	-0.028256663
0.046556659	-0.027624248
0.048392455	-0.026951991
0.050227339	-0.026053966
0.052065867	-0.026059014
0.05390595	-0.026449197
0.055535525	-0.027555585
0.056674328	-0.029079668
0.058515643	-0.029775302
0.060354547	-0.029873314
0.062190021	-0.029121374
0.064023994	-0.027997581
0.065866488	-0.028985386
0.067709733	-0.030159118
0.069551048	-0.030854752
0.071390595	-0.03111213
0.073229767	-0.031276545
0.075068403	-0.031308154
0.076906878	-0.031299922
0.078745353	-0.03129169
0.080585918	-0.031801397
0.082136518	-0.033224347
0.084041633	-0.033910842
0.085880376	-0.033969012
0.087717726	-0.03368189
0.089556254	-0.033686938
0.09139489	-0.033718548
0.093233419	-0.033723596
0.095072108	-0.033768486
0.096911869	-0.034078986
0.09875297	-0.034721498
0.100593964	-0.03533745
0.10243394	-0.035701072
0.104269842	-0.035055376
0.106108799	-0.035166668

x/c	y/c
0.107949472	-0.035702937
0.109790358	-0.036292327
0.111630709	-0.036748912
0.113471006	-0.037192217
0.115310232	-0.037369912
0.117148385	-0.037281997
0.118987932	-0.037539375
0.120828122	-0.037956119
0.122666293	-0.037872631
0.124504589	-0.037820131
0.126343547	-0.037931423
0.128182826	-0.038122398
0.130019371	-0.037636069
0.131857096	-0.03744191
0.133698251	-0.038097703
0.134460319	-0.039010661
0.136231406	-0.039387872
0.138070685	-0.039578848
0.139910821	-0.039982311
0.141749564	-0.040040481
0.143586592	-0.039673676
0.145424799	-0.039599042
0.147262845	-0.039484566
0.14910132	-0.039476334
0.150943529	-0.04039331
0.151330814	-0.041449607
0.15324566	-0.041383483
0.155084188	-0.041388532
0.156919769	-0.040663153
0.158758512	-0.040721324
0.160598541	-0.041098226
0.162437767	-0.041275921
0.164278493	-0.04182547
0.166118683	-0.042242214
0.167957318	-0.042273823
0.169795686	-0.04223903
0.171635555	-0.042576091
0.17347612	-0.043085799

x/c	y/c
0.175318757	-0.044109018
0.177161001	-0.045034848
0.178998833	-0.04486725
0.180837415	-0.044885579
0.182676212	-0.04495703
0.184513829	-0.04473631
0.185889899	-0.04403955

B.2 Dimensionless Airfoil coordinates NLF-0414 General Aviation airfoil

Note that values in between parentheses are the correct airfoil coordinates as denoted in the source. For the simulations in this report the value without parentheses were used.

x/c	y/c
1	-0.02751
0.99471	-0.02723
0.98894	-0.02695
0.98271	-0.02669
0.97602	-0.02645
0.96886	-0.02624
0.96123	-0.02604
0.95316	-0.02587
0.94463	-0.02573
0.93565	-0.02562
0.92823(0.92623)	-0.02556
0.91638	-0.02556
0.90611	-0.02565
0.89542	-0.02586
0.88432	-0.02621
0.87283	-0.02671
0.86096	-0.02739
0.84872	-0.02825
0.83613	-0.0293
0.82319	-0.03053
0.80993	-0.03196
0.79635	-0.03359
0.78249	-0.03548
0.76838	-0.03772
0.75404	-0.04036
0.73948	-0.04333
0.72468	-0.04641
0.70959	-0.04929
0.69418	-0.0517
0.67849	-0.05357
0.66255	-0.05501

x/c	y/c
0.64641	-0.05617
0.6301	-0.05715
0.61363	-0.05798
0.59703	-0.05865
0.58031	-0.05917
0.5635	-0.05955
0.54661	-0.05982
0.52967	-0.05997
0.51269	-0.06002
0.49569	-0.05998
0.47869	-0.05964(-0.05985)
0.46171	-0.05964
0.44476	-0.05935
0.42788	-0.05897
0.41107	-0.05851
0.39435	-0.05796
0.37775	-0.05732
0.36128	-0.05661
0.34495	-0.05583
0.32879	-0.05497
0.31282	-0.05405
0.29704	-0.05307
0.28148	-0.05203
0.26616	-0.05094
0.25108	-0.0498
0.23627	-0.0486
0.22174	-0.04736
0.20751	-0.04607
0.19358	-0.04474
0.17998	-0.04335
0.16672	-0.04193
0.15381	-0.04047
0.14126	-0.03896
0.12909	-0.03742
0.1173	-0.03584
0.10592	-0.03422
0.09494	-0.03255
0.08438	-0.03084

x/c	y/c
0.07426	-0.02909
0.06457	-0.02731
0.05533	-0.02549
0.04655	-0.02363
0.03824	-0.02161(-0.02169)
0.03043	-0.01966
0.02312	-0.01749
0.01635	-0.01515
0.01016	-0.01257
0.00475	-0.00949
0.00104	-0.00516
0	0
0.00069	0.00536
0.00286	0.01082
0.00653	0.01601
0.01154	0.02066
0.01762	0.02472
0.0245	0.02836
0.03202	0.03177
0.04009	0.03506
0.04869	0.03823
0.05782	0.04127
0.06745	0.04418
0.07757	0.04696
0.08816	0.04962
0.09921	0.05218
0.1107	0.05463
0.12261	0.05699
0.13494	0.05926
0.14766	0.06147
0.16076	0.0636
0.17423	0.06566
0.18805	0.06763
0.20221	0.0695
0.21671	0.07128
0.23151	0.07294
0.24661	0.07449
0.26198	0.07591

x/c	y/c
0.27762	0.0772
0.29351	0.07837
0.30962	0.0794
0.32594	0.0803
0.34245	0.08107
0.35913	0.08169
0.37597	0.08216
0.39294	0.08248
0.41002	0.08265
0.4272	0.08264(0.08267)
0.44445	0.08252
0.46175	0.08221
0.47909	0.08175
0.49645	0.08112
0.51379	0.08032
0.53111	0.07934
0.54838	0.07817
0.56558	0.0768
0.58268	0.07524
0.59967	0.07346
0.61653	0.07147
0.63322	0.06926
0.64974	0.06682
0.66605	0.06414
0.68211	0.06115
0.6979	0.05777
0.71336	0.05394
0.72845	0.04963
0.74315	0.04491
0.75749	0.03993
0.77149	0.03484
0.78519	0.02979
0.7986	0.02488
0.81175	0.02016
0.82461	0.01567
0.83719	0.01142
0.84947	0.00744
0.86143	0.00372

x/c	y/c
0.87308	0.00027
0.88438	-0.00293
0.89533	-0.00587
0.90591	-0.00859
0.91611	-0.01108
0.92591	-0.01337
0.9353	-0.01546
0.94426	-0.01737
0.9528	-0.01912
0.9609	-0.0207
0.96856	-0.02211
0.97577	-0.02338
0.98252	-0.02451
0.98881	-0.0255
0.99464	-0.02636

Appendix C

Setting up a simulation in OpenFOAM

In this appendix, the steps that are required to take to set up a case are explained. Then the process of making a mesh is explained. And finally, to make a simulation of the case that was set up is explained. For any OpenFOAM case, the work flow is similar. First a case folder is set up, which contains some folders inside it, which themselves contain the necessary dictionary files required to run the specific simulation. Depending on the solver that will be used, some files may be necessary and some may not be necessary. Then some commands are run, for example in making a flow computation, the mesh must be made using a command *blockMesh*. Then if all other necessary files are present, the simulation can be run.

C.1 Case folder

Every simulation in OpenFOAM requires to work from a so-called case folder. This folder can have any name in principle and for all simulations three folders inside the case folder should be present to start a simulation. These folders are named *0*, *constant* and *system*. The *0* folder contains the initial data of the variables that will be solved for. The files required in this folder depend on the chosen solver and turbulence model, in case the solver *rhoSimpleFoam* is used the required variables are the thermal diffusivity (alphat), turbulent kinetic energy (k), turbulent viscosity/eddy viscosity (nut), turbulence specific dissipation rate (omega), pressure (p), temperature (T) and velocity (U).. The constant folder contains the files *thermophysicalProperties*, *turbulenceProperties*. These files contain data on the models to be used, for example for the gas law, viscosity relation and the turbulence model to be used. Additionally the constant folder contains sub folders which contain the mesh data, that will be generated later. However, one folder named *triSurface* must be present

if a mesh around an object is to be created. The files is a .stl file, that can be created in any CAD software, in this case the .stl files was created by using the airfoil coordinates in SolidWorks, and extruding them in both directions and writing them to a .stl ASCII readable file. Finally, the system folder, it contains the *controlDict* file which is a file that specifies certain simulation parameters, start/stop time, etc. *fvSchemes* contains the information on what type of numerical schemes to use to solve the equations, and *fvSolution* contains variables related to convergence criteria and relaxation factors. Additionally, the system folder contains files that are used by certain additional functions. In this case *snappyHexMeshDict*, *blockMeshDict*, *createPatchDict* and *extrudeMeshDict* were used. All these files are used during the meshing process, that will be explained the next section.

In reality, it may be difficult to start a case from scratch, however it is very common to start a case by using existing cases in the tutorials that are available in the OpenFOAM installation folders. The starting tutorial case for this report was the `WM_PROJECT_DIR/tutorials/incompressible/pimpleDyMFoam/wingMotion/wingMotion2D_simpleFoam` and `WM_PROJECT_DIR/tutorials/incompressible/pimpleDyMFoam/wingMotion/wingMotion_snappyHexMesh` tutorials combined. By adding the required missing files, the case was set up to be suitable for the simulations that were performed earlier.

C.2 Making a mesh

The meshing procedure in OpenFOAM is done using *snappyHexMeshDict*, *blockMeshDict*, *createPatchDict* and *extrudeMeshDict*. These are files located in the *system* folder. Some aspects of the meshing will be explained here. First a .stl file of the wing must be created, for this first the airfoil coordinates are imported into SolidWorks and scaled by 0.9 m, then it is extruded in both direction by 0.5 meters. An .stl file is exported and saved into the *constant/triSurface* folder.

By running the commands: *blockMesh* , *snappyHexMesh -overwrite*, *createPatch* and *extrudeMesh -overwrite* in this order inside the main case folder, the mesh will be generated and put in another folder *polymesh* inside *constant*.

C.2.1 *blockMesh*

This function creates a mesh of hexahedra, in the domain to be discretized. For the next step it is advised that the faces of the hexahedra that will be made around the airfoil are squares, so the length in both x and y direction of each square is approximately the same. Another criterion is that the this initial mesh has intersections with the airfoil, hence the hexahedra have a maximum size, in the plane which is that

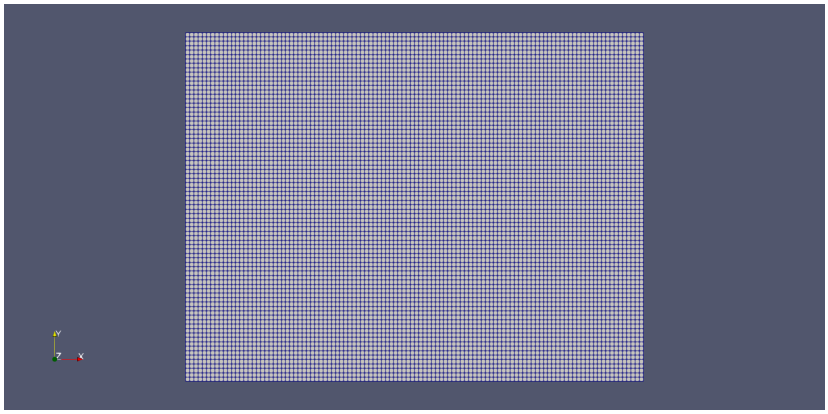


Figure C.1: Initial blockmesh

of the airfoil. However to avoid any other problems it is better to have more than intersection, a summary of the important parameters used in this dictionary file is given in table C.1.

$x_{min}, x_{max}, y_{min}, y_{max}$	$-6c, 15c, -8c, 8c$
number of hexahedra in x direction	110
number of hexahedra in y direction	74
Δx	$\frac{21 \cdot 0.9}{110} \approx 0.172m$
Δy	$\frac{16 \cdot 0.9}{74} \approx 0.195m$

Table C.1: Some parameters set in *blockMeshDict*

The dimension in z direction does not matter as long as it is inside the geometry that was created in SolidWorks, so the hexahedra have dimension $z = -0.1$ and $z = 0.1$ which is inside $z = -0.5$ and $z = 0.5$.

Also, the patches for the inlet, outlet and front and back of the domain are initialized here. The inlet is chosen as the top, left and bottom of the domain, the right is the outlet and the front and back plane are symmetry planes. The file can be found in Listing 6 in Appendix D. The resulting initial mesh can be seen in figure C.1

C.2.2 *snappyHexMesh*

In this step the mesh around the airfoil will be created with the output of *blockMesh* as starting point. The function consists of three different procedures that are iterated, as required. The first procedure is to split the cells at intersection with the airfoil as well as additionally specified regions for example the wake region. The next step is to remove cells that are inside the domain to be excluded, the airfoil. Finally, the cells are snapped into their place such that the cells around the airfoil form the contour of the airfoil as desired. In principle this would be enough to run a simulation, however

an additional final step can be added, which adds layers near the geometry. This is especially important to take into account the effects of the boundary layer. For this the cell heights can be specified, as absolute or relative quantities. The extra option *-overwrite* ensures only to write the final shape of the mesh into the mesh folder. In reality it may be difficult to find working settings and again the tutorial case that was mentioned, will serve as a starting point. A summary of a few values of the variables is shown in table C.2.

refinement box x	-1 to 15
refinement box y	-0.5 to 0.5
nCellsBetweenLevels	10
wing refinement levels	6
refinement box levels	2
nsurfacelayers	5
finallayerthickness	0.7
expansionratio	1.3

Table C.2: Some parameters set in *snappyHexMeshDict*

The refinement box is chosen around the airfoil with given limits in table C.2. It is chosen such that it is all around the airfoil, but also in the wake partly, and slightly upstream. The *nCellsbetweenlevels* allows to give smooth transitions from one level to the other level. The wing refinement level are set to 6, for higher values the grid would not converge, and another thing is that the meshing process also takes much longer for every increased level. The box has 2 refinement level, which could have been better, but these values seemed to work and give relatively good results. On the airfoil 5 levels will be added, of which the relative thickness is set to 0.7, which resulted in an initial layer thickness suitable for the wall functions, corresponding to a $y^+ = 100$.

All in all, it can be said that this way of meshing is not very easily controlled, because many parameters have to be set and by trial and error it is attempted to get a working grid, because for very small grid sizes also it was difficult to obtain convergence, even with coarse grid solutions as initial conditions.

Some important parts of this dictionary files are shown in Listing 7 in Appendix D.

The first step of refinement regions and the splitting of cells with the intersection of the airfoil can be seen in figure C.2

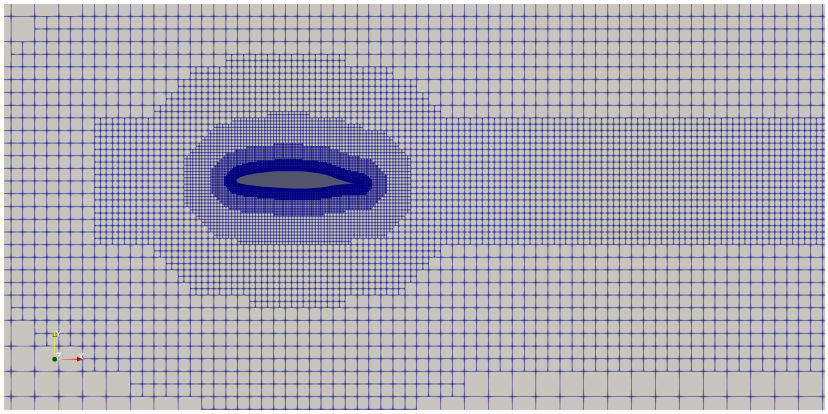


Figure C.2: Cell refinements and splitting at the geometry

A closeup of near the airfoil can be seen in figure C.3

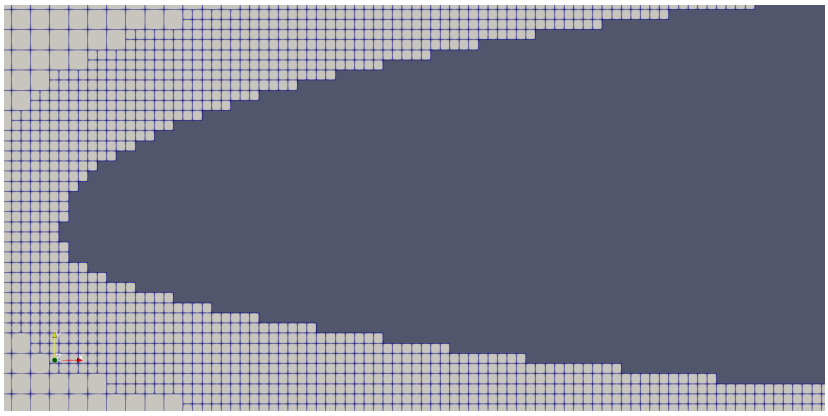


Figure C.3: Closeup after region refinement and cell splitting

Then the cells near the airfoil are snapped such that the contour of the airfoil is smooth instead of the steps as was seen in the previous step. The result is shown in figure C.4

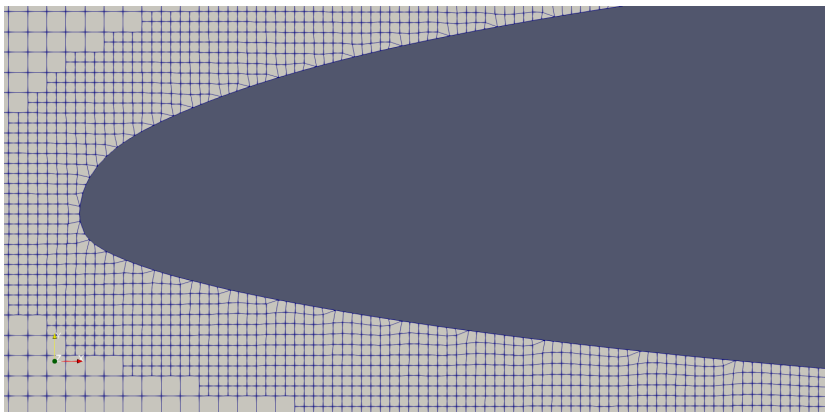


Figure C.4: Snapping of cells near the airfoil

Finally, surface layers are inserted near the airfoil, the result is shown in figure C.5

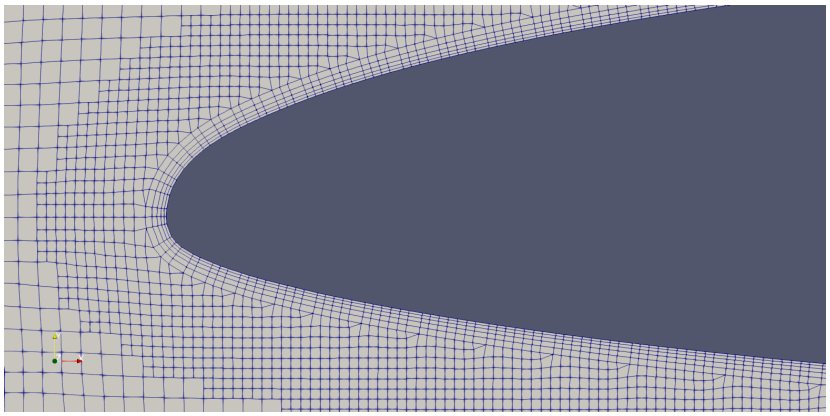


Figure C.5: Insertion of surface layers near the airfoil.

C.2.3 *extrudeMesh* and *createPatch*

The final step in the mesh making process, to obtain a 2d mesh, is to take either front or back face of the mesh created in the previous step and extrude it linearly in one direction such that a 2D mesh is obtained. The reason this has to be one is because *snappyHexMesh* is a 3D mesher, and since we are evaluating a 2D case, this step is required such that in the z direction normal to the plane of in which the airfoil is, there is just one single layer of cell. Because in the *snappyHexMesh* procedure, the mesher also made smaller cells in the z direction. The files corresponding to the function speak for themselves and hence no further explanation is given about the naming convention.

In the process however, the two symmetry planes in the front and back are made so-called empty planes, which is the type suitable for 2d simulations. The files can be found in Listing 8 and 9, respectively for the extrusion and create patch function in Appendix D.

C.3 *rhoSimpleFoam*

Now that the mesh is created, the boundary conditions and other simulations parameters must be set. For the simulations, *rhoSimpleFoam* was found to be the most similar to the simulation in FLUENT. *rhoSimpleFoam* solves the RANS equations using the SIMPLE algorithm for compressible flows. In OpenFOAM only one of the turbulence models was used, the reason being mostly time limitations. For the subsequent simulation we used the $k - \omega$ turbulence model.

The required variables and its corresponding files where explained in Section C.1, the corresponding boundary conditions for these variables were chosen as follows:

boundary	alphat	k	nut	omega	p	T	U
inlet	inletOutlet	inletOutlet	calculated	inletOutlet	inletOutlet	inletOutlet	inletOutlet
outlet	zeroGradient	zeroGradient	calculated	zeroGradient	zeroGradient	zeroGradient	zeroGradient
wing	compressible::alphatWallFunction	kqRWallFunction	nutkWallFunction	omegaWallFunction	zeroGradient	zeroGradient	fixedValue
front and back	empty	empty	calculated	emtpty	empty	empty	empty

Table C.3: boundary conditions for the solver *rhoSimpleFoam*

The exact corresponding values can be seen in the Listing 10 to 16 in Appendix D and they were calculated using another online tool[1], with turbulence intensity set to 1%, the length scale to the chord length, the velocity to 66.9 m/s and the eddy viscosity ratio to 10. For alphat a very small initial field value was assumed. The type of boundary conditions are suitable for inletOutlet, however very often it is common to have zeroGradient for the other boundary conditions which is in this case the outlet. Again the tutorial case served as a starting point for these files, however the variables alphat and T had to be added, because of the use of *rhoSimpleFoam*.

Then the settings in the *constant* folder for *turbulenceProperties* (Listing 17 in Appendix D) is set set to the $k-\omega$ model and *thermophyiscalProperties* (Listing 18 in Appendix D) has entries similar to the ones used in FLUENT, such as the ideal gas law and Sutherland's viscosity law. One may note that many other variables are specified, which is a very common thing among OpenFOAM simulations. Because all the different models and classes built upon each other, explicit declaration of a certain control parameter is required although not being used.

Finally, the controlDict (Listing 19 in Appendix D) was set to run 5000 iterations, later it will appear that this was sufficient, also note that some functions were added in order to monitor the lift and rag coefficients to know whether or not the flow is converged. The fvSchemes (Listing 20 in Appendix D) and fvSolution (Listing 21 in Appendix D) only were modified slightly to take into account the fields for alphat and T, and no further modifications was made to those files inside the tutorial.

C.4 Impingement calculations

Now that the simulation for the flow field is obtained, the next step is to perform particle impingement simulations, the starting points are the results of the final flow iteration, this can be found in the folder *5000*. The flow field remains constant, as the solver name suggest *uncoupledKineMaticParcelFoam*. This solver can be

[1] https://www.cfd-online.com/Tools/turbulence.php

described simply as follows. The used specifies where and how much particles to release in the flow field. Then based upon the forces that are active(which are also specified), the corresponding forces on the particles are evaluated at each time integration step. (which is not equal to the time step specified in the *controlDict*, but it is some fraction of it). Then by simply solving the equation of motion 2.1, the particles' velocities and positions and other parameters are calculated. Assuming the particles to stick, and no further complicated dynamics happening, the particle impingements are calculated, with a characteristic location on the airfoil measured as the curvilinear abscissa β as explained in Section 2.4.1.

In the next section all the parts related to setting up and running the simulation will be explained in further detail. In principle the files that were used in the flow calculation can remain, however slight modification must be made. And the modified *controlDict* can be found in Appendix D. First of all, the *5000* folder should be renamed to *0* and two files in the constant must be added. These are the *kinematicCloudPositions* and *kinematicCloudProperties*. The positions are the positions where the injectors are located and in the *kinematicCloudProperties* file, all particle related parameters are set. Since this is a very general dictionary file, many unrelated parameters have to be set as well, but the most important ones are summarized below.

particleForces	sphereDrag, gravity
integrationScheme	euler/analytical
injectionModel	manualInjection
standardWallInteractionCoeffs	stick/rebound/escape

Table C.4: Some parameters that can be changed in *kinematicCloudProperties*

The particleForces that are present are the sphere drag force and gravity force (also the buoyancy is included in this force). These forces are also present in MultiICE. The integration scheme can be chosen as either Euler or the analytic solution. The difference of using both integration methods and choosing different time steps has been done as well and the results are shown in the next chapter. Finally, *injectionModel* is a manual single injection of particles at the specified locations inside the *kinematicParcelPositions* file. The corresponding file can be found in Listing 23 and 24 in Appendix D.

Appendix D

OpenFOAM files

In this chapter the files used in the different simulation for OpenFOAM are included, in the order as explained in the report in Appendix C. It was not possible to include the full part of the files. Nevertheless, the most important parameters, that were either changed or are very important to the file being considered have been included. For the full files, refer to the source code of OpenFOAM.

To make the folder structure more clear, two illustrations of the flow computation and the particle impingement simulation have been made. These are shown in figure D.1 and in figure D.2.

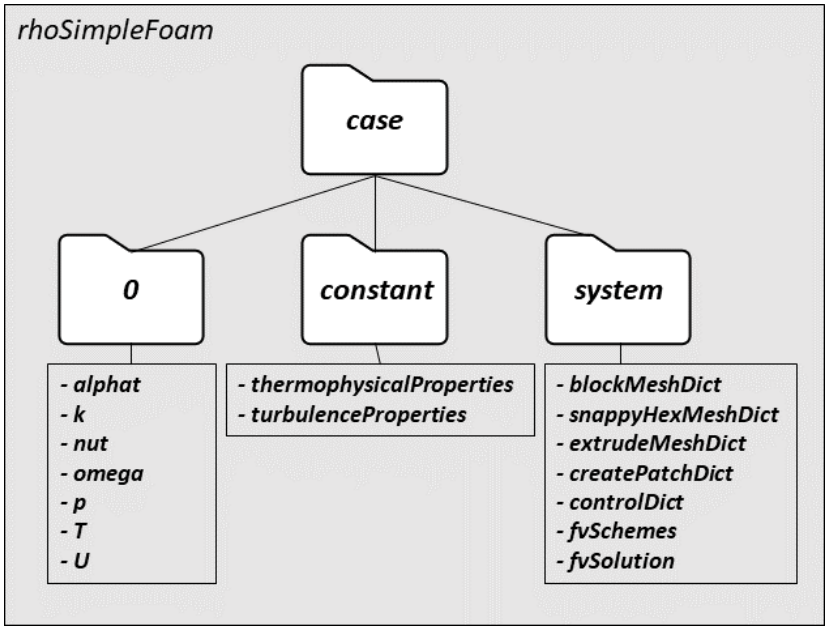


Figure D.1: Case structure of rhoSimpleFoam

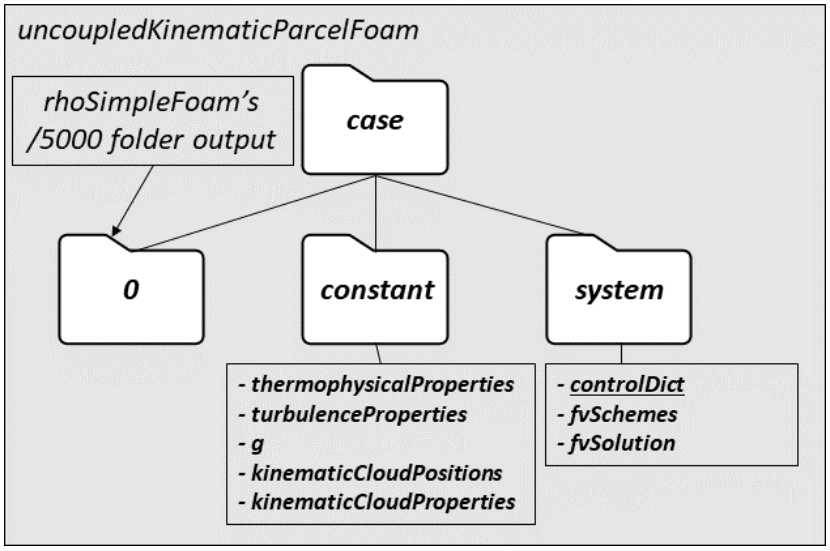


Figure D.2: Case structure of uncoupledKinematicParcelFoam

The *controlDict* file in *uncoupledKinematicParcelFoam* is a different file than Listing 19, that is why it is underlined in figure D.2. The dictionary file can be found in Listing 25 in Appendix D.

```
...
convertToMeters 1;
vertices
(// -6c 15c -8c 8c
    (-5.4 -7.2 -0.1)
    ( 13.5  -7.2 -0.1)
    ( 13.5    7.2 -0.1)
    (-5.4  7.2 -0.1)
    (-5.4 -7.2  0.1)
    ( 13.5   -7.2  0.1)
    ( 13.5   7.2  0.1)
    (-5.4  7.2  0.1)
);
blocks
(
    hex (0 1 2 3 4 5 6 7) (110 79 1) simpleGrading (1 1 1)
edges
(
);
boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 4 7 3)
            (3 7 6 2)
            (1 5 4 0)
        );
    }
    outlet
    {
        type patch;
        faces
        (
            (2 6 5 1)
        );
    }
    symFront
    {
        type symmetryPlane;
        faces
        (
            (4 5 6 7)
        );
    }
    symBack
    {
        type symmetryPlane;
        faces
        (
            (0 3 2 1)
        );
    }
);
mergePatchPairs
(
);
...
```

Listing 6: blockMeshDict

```
...
    refinementBox
    {
        type    searchableBox;
        min     (-1 -0.5 -1);
        max     ( 15  0.5  1);
    }
};
...
    refinementSurfaces
    {
        wing
        {
            // Surface-wise min and max refinement level
            level (6 6);
        }
    }
...
    refinementRegions
    {
        refinementBox
        {
            mode inside;
            levels ((1e15 2));
        }
    }
...
addLayersControls
{
    // Are the thickness parameters below relative to the undistorted
    // size of the refined cell outside layer (true) or absolute sizes (false).
    relativeSizes true;
    // Per final patch (so not geometry!) the layer information
    layers
    {
        wing
        {
            nSurfaceLayers 5;
        }
    }
    expansionRatio 1.3;
    finalLayerThickness 0.7;
    minThickness 0.25;
...
```

Listing 7: parts of snappyHexMeshDict

```
...
// What to extrude:
//      patch   : from patch of another case ('sourceCase')
//      mesh    : as above but with original case included
//      surface : from externally read surface
constructFrom patch;
sourceCase "../wingMotion2D_rhoSimpleFoam";
sourcePatches (symFront);
// If construct from patch: patch to use for back (can be same as sourcePatch)
exposedPatchName symBack;
// Flip surface normals before usage. Valid only for extrude from surface or
// patch.
flipNormals false;
//- Linear extrusion in point-normal direction
extrudeModel        linearNormal;
nLayers             1;
expansionRatio      1.0;
linearNormalCoeffs
{
    thickness       1;//0.05
}
// Do front and back need to be merged? Usually only makes sense for 360
// degree wedges.
mergeFaces false;   //true;
// Merge small edges. Fraction of bounding box.
mergeTol 0;
...
```

Listing 8: extrudeMeshDict

```
...
pointSync false;
patches
(
    {
        // Name of new patch
        name front;
        // Type of new patch
        patchInfo
        {
            type empty;
        }

        // How to construct: either from 'patches' or 'set'
        constructFrom patches;
        // If constructFrom = patches : names of patches. Wildcards allowed.
        patches (symFront);
    }
    {
        // Name of new patch
        name back;
        // Type of new patch
        patchInfo
        {
            type empty;
        }
        // How to construct: either from 'patches' or 'set'
        constructFrom patches;
        // If constructFrom = patches : names of patches. Wildcards allowed.
        patches (symBack);
    }
);
...
```

Listing 9: createPatchDict

```
...
dimensions      [1 -1 -1 0 0 0 0];

internalField   uniform 1e-7;

boundaryField
{
    inlet
    {
        type            inletOutlet;
        inletValue        $internalField;
        value               $internalField;
    }

    outlet
    {
        type            zeroGradient;
        //value           $internalField;
    }

    wing
    {
        type            compressible::alphatWallFunction;
        Prt               0.85;
        value               uniform 0;
    }

front
{
    type empty;
}

back
{
    type empty;
}
}
...
```

Listing 10: alphat

```
...
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0.67;

boundaryField
{
    //#include "include/fixedInlet"
    inlet
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }

    outlet
    {
        type            zeroGradient;//inletOutlet;
        //inletValue      $internalField;
        //value           $internalField;
    }

    wing
    {
        type            kqRWallFunction;
        value           $internalField;//1e-10
    }

front
{
    type empty;
}

back
{
    type empty;
}
}
...
```

Listing 11: k

```
...
dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0.0073;

boundaryField
{
    wing
    {
        type            nutkWallFunction;
        value           uniform 1e-20;//0
    }

    "(front|back||inlet|outlet)"
    {
        type            calculated;
        value           uniform 0;
    }
}
...
```

Listing 12: nut

```
...
dimensions      [0 0 -1 0 0 0];

internalField   uniform 4450;

boundaryField
{

    inlet
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }

    outlet
    {
        type            zeroGradient;//inletOutlet;
        // inletValue      $internalField;
        //value           $internalField;
    }

    wing
    {
        type            omegaWallFunction;
        value           $internalField;
    }
front
{
    type empty;
}

back
{
    type empty;
}
}
...
```

Listing 13: omega

```
...
dimensions      [1 -1 -2 0 0 0 0];

internalField   uniform 101325;

boundaryField
{
    inlet
    {
        type            inletOutlet;//zeroGradient;
        inletValue        uniform $pressure;
        value               uniform $pressure;
    }

    outlet
    {
        type            zeroGradient;//fixedValue;
        //value           $internalField;
    }

    wing
    {
        type            zeroGradient;
    }
front
{
    type empty;
}

back
{
    type empty;
}
}
...
```

Listing 14: p

```
...
dimensions      [0 0 0 1 0 0 0];

internalField   uniform 258;

boundaryField
{
    inlet
    {
        type            inletOutlet;
        inletValue        $internalField;
        value               $internalField;
    }

    outlet
    {
        type            zeroGradient;
        //value           $internalField;
    }

    wing
    {
        type            zeroGradient;
    }
front
{
    type empty;
}

back
{
    type empty;
}
}
...
```

Listing 15: T

```
...
dimensions      [0 1 -1 0 0 0];

internalField   uniform (66.9 0.350 0);

boundaryField
{
    inlet
    {
        type            inletOutlet;
        inletValue      $internalField;
        value           $internalField;
    }

    outlet
    {
        type            zeroGradient;
    }

    wing
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }
front
{
    type empty;
}

back
{
    type empty;
}
}
...
```

Listing 16: U

```
...
simulationType RAS;

RAS
{
    RASModel        kOmega;

    turbulence      on;

    printCoeffs     on;
}
...
```

Listing 17: turbulenceProperties

```
...
thermoType
{
    type            hePsiThermo;
    mixture         pureMixture;
    transport       sutherland;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleInternalEnergy;
}

mixture
{
    specie
    {
        molWeight   28.9;
    }
    thermodynamics
    {
        Cp          1007;
        Hf          0;
    }
    transport
    {
        As          1.4792e-06;
        Ts          116;
    }
}
...
```

Listing 18: thermophysicalProperties

```
...
application     rhoSimpleFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         5000;
deltaT          1;
writeControl    runTime;
writeInterval   100;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
functions
{
    forceCoeffs
    {
    type                forceCoeffs;
    functionObjectLibs  ( "libforces.so" );
    writeControl        timeStep;
    writeInterval       1;

    patches             ( wing );
    pName               p;
    UName               U;
    rhoName             rhoInf;
    log                 true;

    liftDir             (-0.00524 0.99999  0);
    dragDir             (0.99999 0.00524 0);
    CofR                (0.25 0 0);
    pitchAxis           (0 0 1);

    magUInf             66.9;
    rhoInf              1.36;
    lRef                0.9;
    Aref                1;
    }

    p_tools
    {
        type                pressure
        functionObjectLibs  ("libfieldFunctionObjects.so");//("libutilityFunctionObjects.so");
        enabled             yes;
        writeControl        timeStep;
        writeInterval       500;
        pRef                101325;
        pInf                101325;
        rhoInf              1.36;
        UInf                (66.9 0.350 0);
        calcTotal           no;
        calcCoeff           yes;
    }

}
...
```

Listing 19: controlDict

```
...
ddtSchemes
{
    default steadyState;
}

gradSchemes
{
    default         Gauss linear;
    grad(p)         Gauss linear;
    grad(U)         Gauss linear;
}

divSchemes
{
    default         none;
    div(phi,U)      bounded Gauss linearUpwind grad(U);
    div(phi,k)      bounded Gauss upwind;
    div(phi,omega)  bounded Gauss upwind;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
    div(phi,Ekp)    bounded Gauss upwind;
    div(phi,e)            bounded Gauss upwind;
}

laplacianSchemes
{
    default         Gauss linear corrected;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}
...
```

Listing 20: fvSchemes

```
...
solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-7;
        relTol          0.1;
        smoother        GaussSeidel;
    }
    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
    k
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
    omega
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
    e
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
}
SIMPLE
{
    nNonOrthogonalCorrectors 0;
}
relaxationFactors
{
    fields
    {
        p               0.3;
    }
    equations
    {
        "(U|k|omega)"   0.7;
        "(U|k|omega)Final" 1.0;
    }
}
cache
{
    grad(U);
}
...
```

Listing 21: fvSolution

```
...
dimensions      [0 1 -2 0 0 0];
value           ( 0.05136 -9.810 0 );
...
```

Listing 22: gravitational acceleration file, g

```
...
(       -2      -0.063      0.6     )
(       -2      -0.065      0.6     )
(       -2      -0.067      0.6     )
(       -2      -0.069      0.6     )
(       -2      -0.071      0.6     )
(       -2      -0.073      0.6     )
(       -2      -0.075      0.6     )
(       -2      -0.077      0.6     )
(       -2      -0.079      0.6     )
(       -2      -0.081      0.6     )
(       -2      -0.083      0.6     )
(       -2      -0.085      0.6     )
(       -2      -0.087      0.6     )
(       -2      -0.089      0.6     )
(       -2      -0.091      0.6     )
...
```

Listing 23: kinematicCloudPositions

```
...
solution
{
    active          true;
    coupled         false;  //true;
    transient       yes;
    cellValueSourceCorrection off;
    //maxCo           0.3;
...
    integrationSchemes
    {
        U               analytical; //euler
    }
}
...
subModels
{
    particleForces
    {
        sphereDrag;
        gravity;

    }
...
   injectionModels
    {
        model1
        {
            type            manualInjection;
            massTotal       1000;  //2.03e-06;
            SOI             0;
            positionsFile   "kinematicCloudPositions";
            U0              (66.899 0.350 0);
            parcelBasisType fixed;
            nParticle       1;

            sizeDistribution
            {
                type         fixedValue;
                fixedValueDistribution
                {
                    value          20e-6;
                }
            }
        }
    }

    dispersionModel gradientDispersionRAS;  //none;
    patchInteractionModel standardWallInteraction;
    heatTransferModel none;
    compositionModel singlePhaseMixture;
    phaseChangeModel none;
    stochasticCollisionModel none;
    collisionModel none;
    surfaceFilmModel none;
    radiation       off;
    standardWallInteractionCoeffs
    {
        type            stick;  //rebound,escape;
    }
   ...
}
...
```

Listing 24: kinematicCloudProperties

```
...
application     uncoupledKinematicParcelFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         0.045;
deltaT          0.0000003;
writeControl    timeStep;
writeInterval   10000;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
...
```

Listing 25: controlDict as used in uncoupledKinematicParcelFoam

Appendix E

Company information and personal reflection

E.1 Company information

The Italian Aerospace Research Centre is located in Capua, in the province of Campania in Italy. It was founded in 1984, with the aim of promoting research and technological development in the field of space and aeronautics. The number of workers is 370, which mostly work in research and development.

E.2 Personal reflection

I took part in an internship at CIRA as part of my Master curriculum of Mechanical Engineering at the department of Thermal and Fluid Engineering at the University of Twente.

While I was working, I worked as a research intern on icing of aircraft wings. The group, that I took part in, is led by Mr. Mingione. He leads the Fluid Mechanics department at CIRA.

For me it was a very pleasant experience to work at CIRA, because it gave me opportunity to apply my knowledge, that I have gained so far, in a company environment. Also during my work, I gained a lot of new competences.

I learned to work with different programs, which are commonly used in the industry, for performing fluid flow simulations. I believe this will give me an advantage, when I will start to look for my next career opportunity after I graduate. Another advantage is that, if I will be using any of the programs used during my internship for my Master thesis, it might come in handy, that I have already used the simulation programs.

Another important competence I gained, was planning ahead and organizing

files. During the work, I realized, that especially in CFD simulations, a lot of data is processed, and to keep track it is very important to have everything organized neatly.

Aside from the technical competences, I gained a soft skill, to be in touch with colleagues, during coffee breaks and the lunch breaks. Of course, this is a very important skill, when you work among a lot of people. The social interaction, is very valuable, because in my opinion it gives some relief during the work and frustrations encountered, occasionally. Another thing I learned is that, there are highly specialized people, that can give good advice, for example, when making a mesh.

As I went to Italy, I also got the chance to learn more about Italian culture and experience it directly. During the breaks, I had a lot of opportunity to discuss this with colleagues, and it was for myself very valuable to see differences and common things with Italian culture. Some colleagues, also gave me some history background about places and cities that I visited during my stay in Italy, which were really interesting.

To conclude, I was very happy I got this opportunity and therefore I am grateful to Mr. Hoeijmakers and Mr. Mingione for making this possible. In my opinion I learned a lot of new things in relatively short time period, which I highly appreciate. It also gave me good insight on my future career and about what I would like to do after I graduate.