BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF AEROSPACE ENGINEERING

LETECKÝ ÚSTAV

OPEN SOURCE AND COMMERCIAL CFD CODES COMPARISON ON 2D AIRFOIL ANALYSIS

POROVNÁNÍ VÝPOČETNÍCH MOŽNOSTÍ OPEN SOURCE A KOMERČNÍHO NÁSTROJE PRO CFD ANALÝZU PROFILŮ

BACHELOR'S THESIS

BAKALÁŘSKÁ PRÁCE

AUTHOR — **Tomáš Řezníček**

AUTOR PRÁCE

SUPERVISOR — **Ing. Robert Popela, Ph.D.**

VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF MECHANICAL ENGINEERING

Assignment Bachelor's Thesis

Institut:	Institute of Aerospace Engineering
Student:	**Tomáš Řezníček**
Degree programm:	Engineering
Branch:	Fundamentals of Mechanical Engineering
Supervisor:	**Ing. Robert Popela, Ph.D.**
Academic year:	2022/23

As provided for by the Act No. 111/98 Coll. on higher education institutions and the BUT Study and Examination Regulations, the director of the Institute hereby assigns the following topic of Bachelor's Thesis:

Open source and commercial CFD codes comparison on 2D airfoil analysis

Brief Description:

So called open source CFD codes are interesting alternative to commercial CFD bundles. Nevertheless, the quality of results is in CFD quite dependent upon solver selection and engineer inputs. Comparison of both approaches on same benchmarking task should give better idea about applicability and advantages of each alternative.

Bachelor's Thesis goals:

CFD analysis of 2D airfoil characteristics in open source and commercial CFD code (OpenFoam and Ansys Fluent foreseen). Comparison of results to tunnel tests and conclusion about applicability and advantages of both codes.

Recommended bibliography:

OPENFOAM User Guide, https://www.openfoam.com/documentation/user-guide

Ansys Fluent User Guide.

Deadline for submission Bachelor's Thesis is given by the Schedule of the Academic year2022/23

In Brno,

L. S.

doc. Ing. Jaroslav Juračka, Ph.D. Director of the Institute	doc. Ing. Jiří Hlinka, Ph.D. FME dean

Abstrakt
Cílem této práce je zjistit schopnosti open-source a komerčních CFD balíků a použít tuto znalost k porovnání jejich využití při analýze aerodynamických profilů ve dvourozměrném prostoru. CFD kody vybrané pro tento účel byly open-source software OpenFOAM a komerční software ANSYS Fluent. Veškeré simulace byly doprovázeny a porovnávány s měřením z aerodynamického tunelu.

Summary
The purpose of this study is to explore the capabilities of an open-source and commercial CFD solutions, and use that knowledge to compare their use in analyzing airfoils in two-dimensional space. The solutions selected for this purpose were the open-source software OpenFOAM and a commercial product ANSYS Fluent. All simulations are supported and compared to wind tunnel measurements.

Klíčová slova
CFD, OpenFOAM, ANSYS, Fluent, Výpočetní sít, Koeficient vztlaku, Koeficient odporu, Koeficient klopného momentu, Větrný tunel, Profil, LS-0413, GA(W)-1

Keywords
CFD, Computational Fluid Dynamics, OpenFOAM, ANSYS, Fluent, Mesh, Lift coefficient, Drag coefficient, Pitching moment coefficient, Wind tunnel, Airfoil, LS-0413, GA(W)-1

ŘEZNÍČEK, T.*Porovnání výpočetních možností open source a komerčního nástroje pro CFD analýzu profilů*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2023. 35 s. Vedoucí Ing. Robert Popela, Ph.D.

Statement of authenticity
I, Tomáš Řezníček, declare that I prepared this bachelor's thesis independently and disclosed all sources and literature.

Tomáš Řezníček
2023-05-26

At this point, I would like to thank my thesis supervisor Ing. Robert Popela, PhD for his guidance, advice, comments, patience and valid criticisms while writing and researching this thesis.

Tomáš Řezníček

Contents

Introduction

Fluid dynamics is an area often encountered in engineering. Understanding how a fluid interacts with an object, structure, or a system can be a major part of engineering design. One such area, where the prediction of fluid flows and behavior is of paramount importance is the aerospace industry. Since wings play a pivotal role in the behavior of aircraft, the ability to accurately predict their behavior across a wide range of conditions is critical for achieving desired flight characteristics and improving overall efficiency.

In an effort to predict these phenomena, various methods have been employed to predict and model flow fields around objects. One such method is Computational Fluid Dynamics (CFD), which is used to mathematically predict fluid flows, interactions, heat transfer, forces exerted by fluids on objects and other relevant properties of interest.

A number of different CFD solutions have been developed, each with its unique strengths and weaknesses. However, most commercial CFD software requires a significant investment due to licensing costs, and so cheaper alternatives are often sought. In response to this demand, a number of open-source CFD packages have been developed. But while commercial codes often benefit from ongoing investment into their development, driven by their commercial nature, open-source solutions often rely on the contributions of volunteers for their development. As such, concern regarding their robustness and capabilities are often brought into question.

This thesis aims to evaluate the capabilities of such open-source software when analyzing the flows around airfoils in two-dimensional space and compare them to those of their commercial counterparts. By doing so, an insight into the applicability of both of these options in the context of airfoil analysis and wing design can be gained. The results of this study could inform the selection of an appropriate CFD solution for similar applications, and contribute to improving the understanding and utilization of these tools in the aerospace industry.

1 CFD

Computational Fluid Dynamics software is a type of computer program that uses numerical methods and algorithms to solve problems concerning fluid flow, its behavior and heat transfer. To do this, modern CFD solutions employ the finite volume method in combination with solving equations governing fluid flow based on Navier-Stokes equations.[16] [**empty citation**]

CFD solutions can be divided into three parts: pre-processor, solver and post-processor. The pre-processor is used to define the computational domain and create a mesh, define boundary conditions and configure a turbulence modeling approach. The solver then applies numerical methods to discretize equations over the mesh and solves them iteratively over each cell individually, yielding information about flow properties in their respective locations. Once a solution has been reached, the post-processor can be used to visualize fields of interest or calculate forces acting upon objects or surfaces.

Since its introduction, CFD has become a popular tool in engineering design and scientific study. This popularity can be attributed to its cost-effectiveness and time-saving capabilities as compared to other methods such as experimental testing. Moreover, CFD offers the advantage of visualizing flow properties that might otherwise be difficult or even impossible to measure or observe experimentally, such as vorticity, total pressure coefficient, velocity profiles, and wall shear stress.

However, CFD encounters certain challenges linked to their inherent complexity. Proper understanding of fluid dynamics principles and the underlying algorithms involved are crucial in order to correctly setup, execute and evaluate every simulation. Because an analytical solution to Navier-Stokes equations has not yet been found, the reliability of CFD simulations poses a significant drawback, and result validation remains an important part of any process where these methods are employed.

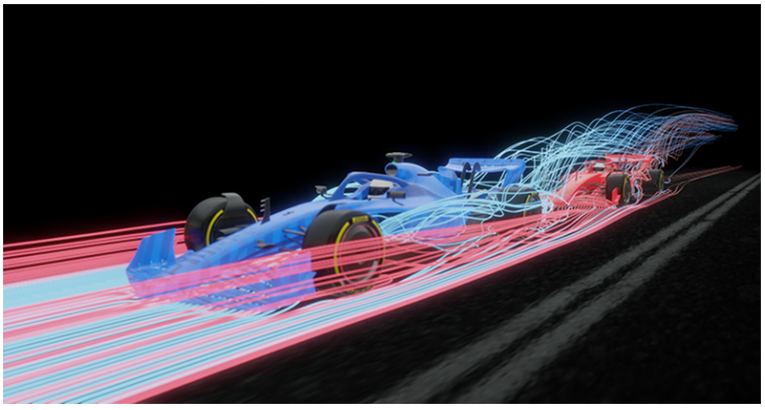

Figure 1.1: Result of a CFD simulation showing streamlines around an F1 car

1.1 History of CFD software

The history of CFD traces back to the mid-20th century when researchers began exploring the application of new developments in finite difference and finite element methods to solve equations governing fluid flow. Notably, F.H. Harlow, J.E. Welch and A.A Amsden at Los Alamos made significant contributions by introducing the explicit transient

algorithm called Marker-And-Cell (MAC). Building on this algorithm, D.B Spalding and his student S. V. Patankar at the Imperial College of London developed the SIMPLE algorithm, which marked a major milestone in CFD. This development was accompanied by the introduction of the k-epsilon turbulence model, which has since become a workhorse of practical engineering flow calculations. [6]

These advancements, coupled with an increase in computing power saw a shift in focus on techniques for solving flows around arbitrarily shaped geometries in the 1980s. CFD began to see use in industrial applications, including in the aerospace, energy and automotive sectors, however it wasn't until the 1990s when commercial software saw widespread adoption. CFD solutions such as Fluent (now ANSYS Fluent) further accelerated the embrace of these technologies across a wide range of industries and marked the shift of CFD into the mainstream. In the early 2000s, the highly popular open-source CFD software OpenFOAM was introduced. Since its inception, it has served as the basis for a number of different OpenFOAM forks, as well as several commercial CFD services. It was also integrated into other software in the form of addons, bringing CFD capabilities to a broader audience, including hobbyists and enthusiasts. The versatility and accessibility of OpenFOAM have contributed to its widespread use and its integration into various domains beyond traditional engineering applications. [6] [16]

1.1.1 ANSYS CFD

ANSYS is a commercially available suite of software tools for performing simulations in a wide range of engineering disciplines. Included among these tools are tools for CFD. The primary module of ANSYS CFD is Fluent. It was independently developed in the 1980s, and was purchased by ANSYS to be incorporated into their software suite in 2006. Today, it is one of the most widely used commercial CFD solutions on the market. The advantage of ANSYS CFD lies in its integration with other ANSYS modules, allowing it to handle multiphysics simulations. [3]

ANSYS is commercial software and requires a license to use. However, a free student version of ANSYS products is available, which offers a limited set of features and capabilities and which is intended for educational purposes.

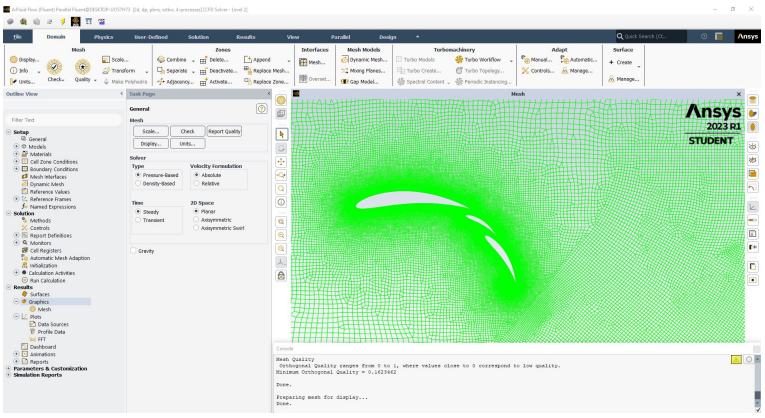

Figure 1.2: ANSYS Fluent UI

1.1.2 Autodesk CFD

Autodesk CFD is a CFD package developed by Autodesk; a provider of a wide range of software packages and solutions designed for engineering, architecture, manufacturing and media industries. Along with a solver, Autodesk CFD includes tools for mesh generation, geometry manipulation and post-processing. One of Autodesk CFD's advantages lies in its integration with other Autodesk software, which allows its users to seamlessly integrate the software into their workflow to form a complete design package.

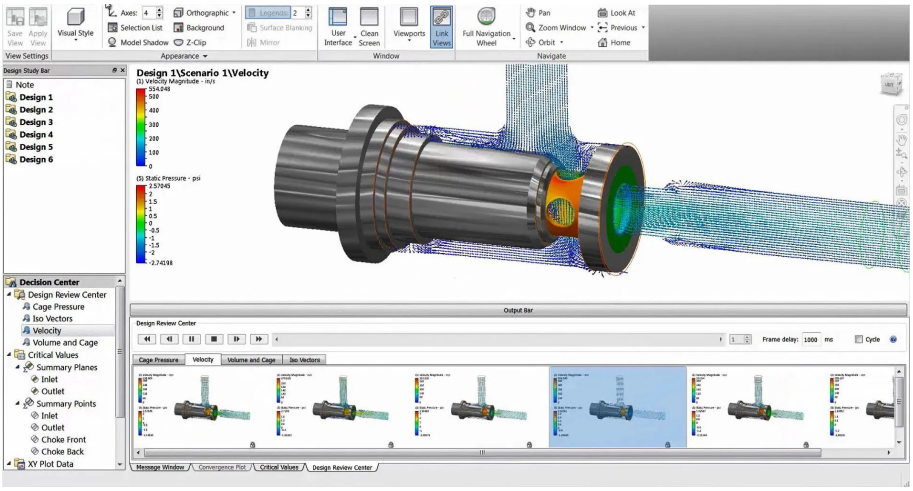

Figure 1.3: Autodesk CFD User Interface [15]

1.1.3 OpenFOAM

OpenFOAM is a popular CFD toolkit licensed under the GNU General Public Licence. Several different versions of OpenFOAM are available, with the most popular versions being released by OpenFOAM.com and OpenFOAM.org. It offers three distinct meshing algorithms as well as a wide variety of tools, utilities and solvers. Also included is a library of tutorial or demo cases that demonstrate the use of OpenFOAM's various functions. Limited post-processing capabilities are also included, although the use of third-party post-processing software such as ParaView is strongly recommended.[10] A notable disadvantage of OpenFOAM lies in its lack of a graphical user interface (GUI), as it is completely operated using a terminal and by manually editing text files. That means settings which are usually automated in other software need to be set and modified manually by the user. The biggest advantage of OpenFOAM lies in its open-source nature, as it allows its users to inspect and modify the program's source code in order to meet their needs. Being freely available, it is also a great option for anyone looking for a more affordable CFD solution.

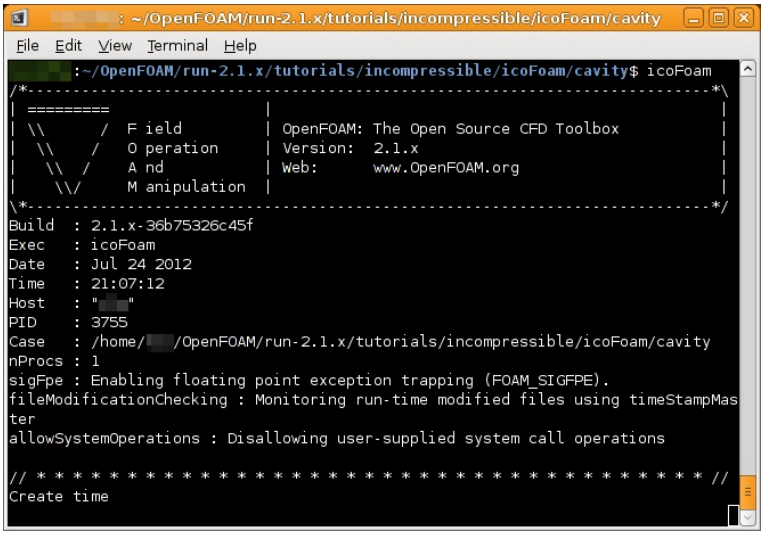

Figure 1.4: Example of OpenFOAM terminal command execution [5]

1.1.4 Simscale

Simscale is a cloud-based simulation platform founded in 2012 by David Heiny, Vincenz Dölle, Johannes Probst, Alex Fischer, and Anatol Dammer. It offers a range of simulation capabilities, including CFD, structural analysis and thermal simulations for solving heat transfer. It is built on modified open-source codes like OpenFOAM and Code_Aster.[14][17] Simscale uses a subscription-based pricing model. Its users can choose from four different financial plans, each with different features, pricing and capabilities, including a free option that provides its users with 3000 core hours of computational time and 500GB of storage for any project that is made publicly available, but private projects are also available with any of the three paid membership plans.[12] Being a cloud-based platform, Sinscale's advantage lies in its ability to be run entirely from a browser and as such can be run on any platform with graphical user interface without the need for local installation or maintenance. Its use of Simscale's cloud infrastructure also removes the need for large investments into the procurement of powerful hardware by its users and reduces simulation time.

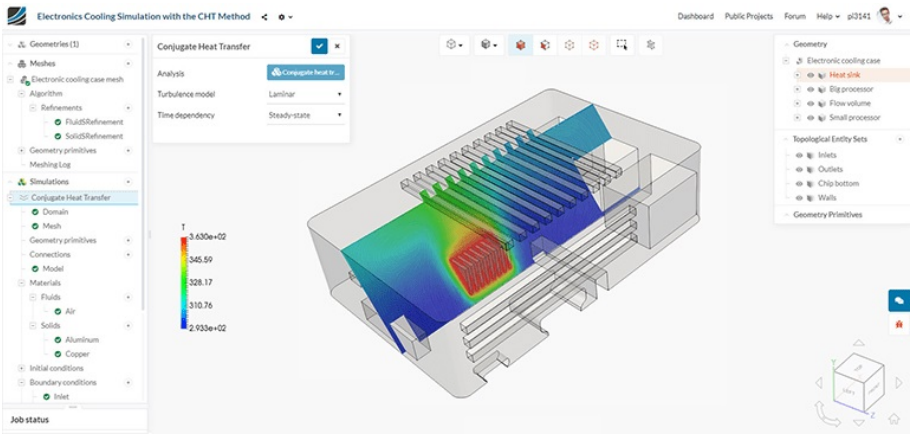

Figure 1.5: Simscale User Interface[13]

1.1.5 Airshaper

Airshaper falls into a special category of CFD software. Unlike traditional CFD solutions, where users set up simulations manually, Airshaper services are fully automated. A user provides the geometry to be simulated and defines a small number of variables such as flow velocity or fluid type. The service then handles the creation of the domain, mesh creation, solver setup and simulation execution. The results of the simulation are then sent to the user electronically, or can be displayed using the service's basic visualization tools. For interested users, the option for results to be exported for further analysis in third-party applications also exists.[7] This service is particularly suited for users whose experience with CFD software is limited, but who nevertheless wish to evaluate the aerodynamic properties of their designs with minimal time and financial investment. As the service is fully automated, however, the control its users have over the simulation is limited. Similar to Simscale, Airshaper is entirely accessed through a browser and all computations are performed on the service's hardware. Users are charged per simulation, with pricing determined based on the simulation type and number of cells used.

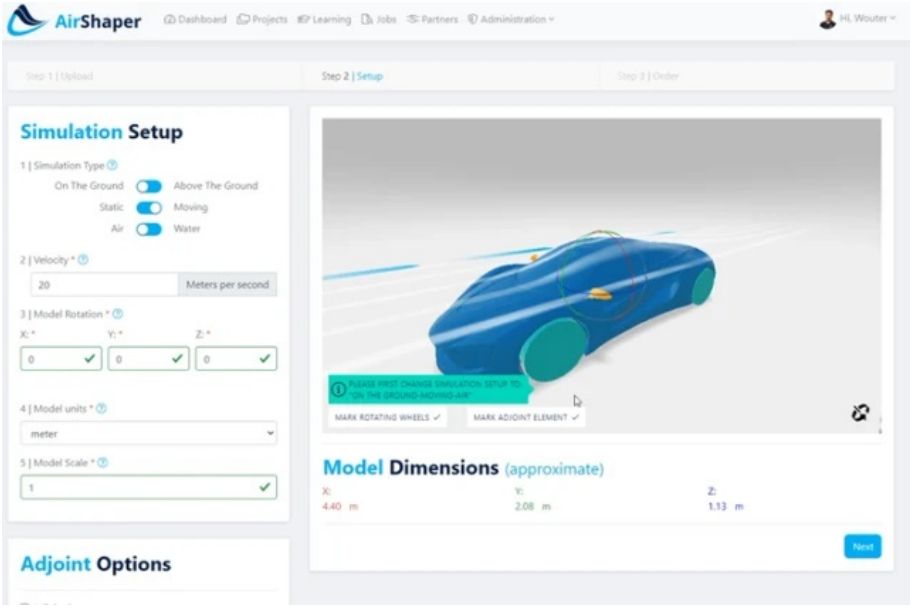

Figure 1.6: Airshaper User Interface [4]

2 Characteristics of airfoils

An airfoil is a two-dimensional shape designed to create lift while minimizing drag by manipulating the flow of a surrounding fluid. They define the cross-sectional shape of a wing or a turbine blade.

2.1 Airfoil nomenclature

In the early years of the 20th century, airfoil designs were personalized. However, in the 1930s, the National Advisory Committee for Aeronautics (NACA) conducted a series of tests using airfoil shapes constructed systematically. During these tests, a nomenclature was used that has since become a standard, and which is used throughout this thesis.[2]

The distance between the leading and trailing edges, connected by a straight line (called the chord line) is simply designated as the chord of the airfoil, commonly denoted by a lowercase letter c. Halfway between the upper and lower surfaces of the airfoil lies the mean camber line. It is a curve, the maximum distance of which is measured perpendicular to the mean camber line itself. Airfoils whose mean camber line is coincident with the chord line (that is, their camber is equal to zero) are referred to as symmetrical airfoils, while airfoils with camber with a value higher than zero are asymmetrical airfoils. Airfoil thickness corresponds to the maximum distance between the upper and lower airfoil surfaces as measured perpendicular to the chord line, and is usually denoted in % of the chord. Angle of attack of an airfoil (often referred to simply as the angle of the airfoil), is the relative angle between the airfoil chord line and its direction of travel. [2]

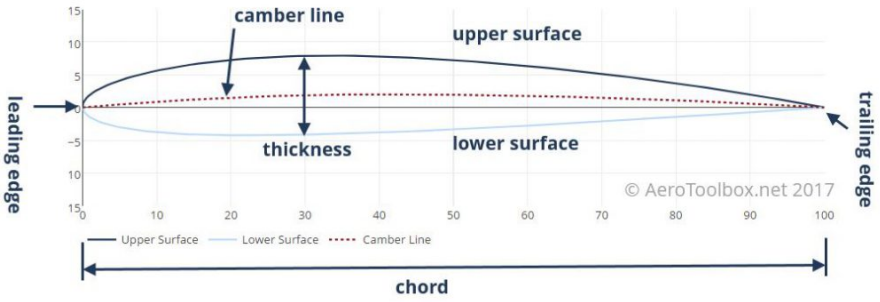

Figure 2.1: Airfoil nomenclature [1]

2.2 Lift

Lift is a force exerted on an airfoil in a direction normal to the direction of travel. Airfoils create lift by creating a pressure difference between their upper and lower surfaces, which results in a net force being exerted on the airfoil. In order to calculate the lift acting on an airfoil, CFD solutions integrate pressure over its surface.

3 Methodology

To evaluate the capabilities of commercial and open-source CFD software, this thesis focuses on comparing two different CFD solutions - ANSYS Fluent and OpenFOAM. This choice was made to select a representative from both commercial and open-source licensing options. Both programs were selected for their popularity in the CFD field and within their respective financial models. Each software was evaluated based on two aspects - meshing and solution. These aspects were selected, as they are both critical to the software's ability to accurately resolve fluid behavior, and together determine the capabilities of a CFD software. The post-processing capabilities of each program were not evaluated as their differences primarily involve subjective metrics, making them difficult to quantify and evaluate in an objective manner.

3.1 Approach

To compare each software, this thesis replicates an experimental study of an airfoil measurement in a wind tunnel. The airfoil selected for this purpose was the LS-0413 airfoil (previously designated as GA(W)-1), as it has been extensively analyzed both experimentally and numerically. It is a 17% thick asymmetrical airfoil, which was used on aircraft such as the Piper PA-38 Tomahawk.[2]

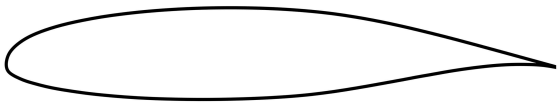

Figure 3.1: LS-0413 airfoil

3.2 Experimental measurement

In this thesis, the study of the LS-0413 airfoil performed by Robert J. McGhee in his work "Low-speed aerodynamic characteristics of a 17-percent-thick airfoil section designed for general aviation applications" was chosen to be recreated, as many measurements of the airfoil behavior were performed under a wide range of conditions. These conditions and the methodology used in this study are described in detail, allowing for a close recreation in CFD.

3.3 Case setup

The wind tunnel test section used in the aforementioned study had a height of 2.286m. The airfoil had a chord length of 0.5842 meters, and was attached on both ends to circular endplates, which rotated the airfoil about an axis located at 0.25c. The airfoil characteristics were measured at various air speed velocities and Reynolds numbers, which were altered by changing the stagnation pressure of the wind tunnel. In This study, the measurement performed at a Mach number of 0.15 and a Reynolds number of 1.9Ee6 was used as a reference to compare analytical solutions to.

4 Meshing

Both OpenFOAM and ANSYS Fluent employ the finite volume method, which necessitates the creation of a mesh. A mesh is a set of a finite number of individual elements, known as cells that partition the computational domain, allowing the fluid flow to be numerically calculated by solving equations over each cell individually. A mesh plays a crucial role in the accuracy and stability of a simulation, and as such the ability of a CFD software to accurately predict fluid flow will be closely linked to its ability to construct high-quality meshes. As such, the meshing capabilities of both OpenFOAM and ANSYS CFD packages were evaluated in this thesis.

4.1 Mesh and cell types

There are two approaches to creating a mesh to be used for CFD - structured and unstructured. A structured mesh is a type of mesh that is organized in an organized and structured way. The size, shape and direction (rotation) of cells is defined by the user, usually by defining a set of lines and curves that the cells follow. This approach usually results in a more accurate solution of the simulation, as the quality of cells can be higher, the cell sizing can be controlled more precisely and the cell rotation can be aligned with the surrounding flow. These types of meshes can be more difficult and time-consuming to create, however, especially as the complexity of the geometry increases. In such cases, an unstructured mesh may be preferable. An unstructured mesh is a type of mesh that consists of cells of various shapes and which has no obvious structure. The mesh is usually defined or controlled by specifying cell type and refinement inside a given area (sometimes referred to as "bodies of influence" or "refinement zones"), on or in proximity of a specified surface, edge or point. The benefit of this approach is that the user does not need to carefully construct the mesh around the geometry "by hand", but instead specify cell sizings in relevant areas and let the program fill the remaining space automatically. The disadvantage of this approach is that the cell quality is usually lower as the program attempts to create a mesh that satisfies the criteria set by the user while also precisely following the geometry. It can also be more computationally intensive, as the majority of the work is shifted from the user to the program.

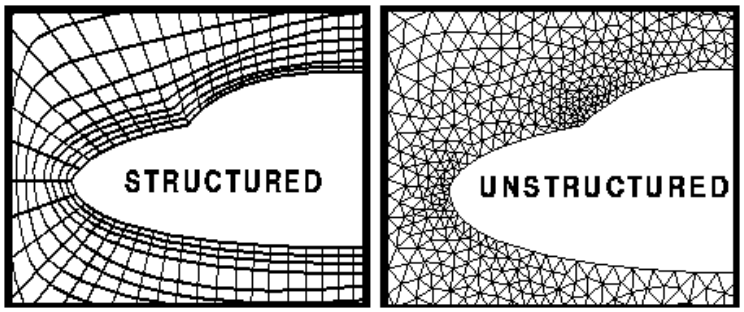

Figure 4.1: Comparison between structured and unstructured mesh

As the mesh quality achieved through the use of a structured grid is mostly dependent on user input, this thesis focuses only on evaluating the capabilities of meshing algorithms when creating unstructured meshes.

4.2 Mesh refinement

Mesh refinement (often also referred to as sizing) refers to the size of cells within the computational domain. As equations are solved over every cell individually, each cell carries information about the flow in its location within the domain. As such, as the number of cells within the domain increases, the amount of information available to describe the flow increases proportionally. That means a sufficient refinement of the mesh in key areas is necessary to accurately resolve the flow and its structures in these areas. As equations have to be solved over each cell individually however, the computational cost increases with the number of cells. As such, a tradeoff must be made between accuracy and computational complexity. The standard approach is to use a finer mesh refinement in areas of interest where an accurate prediction of fluid behavior is critical for accuracy of the solution, while a coarser mesh refinement is used in areas where an accurate prediction of the fluid flow has lower impact on the overall accuracy of the solution. In order to accurately capture the flow within the domain, four zones, each with progressively finer cell refinement, were created. The sizes and placement of these zones were chosen to increase mesh density close to the airfoil surface and its wake, and were kept unchanged throughout all simulations in order to ensure consistency between individual runs.

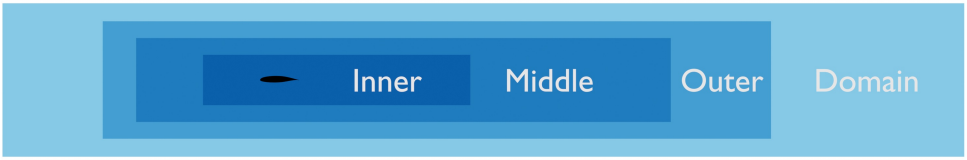

Figure 4.2: Refinement zone schematic

To determine the cell sizing to be used in these areas, a mesh independence study was performed - a simulation was run for several different cell sizing configurations and the change in result was observed for each configuration. The results obtained from this study are visualized in fig. 4.3. The airfoil surface refinement was chosen individually for each meshing algorithm, so as to allow each to use a cell size that allows for the most accurate result to be obtained.

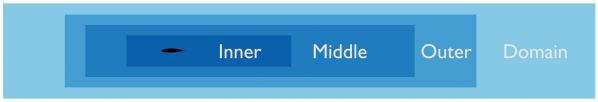

Figure 4.3: Effect of mesh refinement on simulation result

Cell sizings inside each refinement zone are listed in table 4.1.

Table 4.1: Region cell sizings

Region	*Domain*	*Outer*	*Middle*	*Inner*
Element size [m]	0.06	0.03	0.015	0.0075

4.3 Mesh quality

Among the most important metrics of a mesh is mesh quality. There are several criteria that together determine the quality of a mesh, such as non-orthogonality, skewness or smoothness. Each quality metric can have a different effect on a simulation, from increasing numerical diffusion to worsening solution stability. The exact definition of these metrics can vary across different CFD codes, so OpenFOAM definitions were used.

4.3.1 Non-orthogonality

Face-based non-orthogonality is the angle between the face normal vector and the vector connecting cell centroids. The issue non-orthogonality poses stems from the difficulty in evaluating the dot product of the normal vector and the velocity gradient. In most CFD solutions, non-orthogonal correctors can be employed to mitigate the negative effects of non-orthogonality, at the cost of increased solution time.

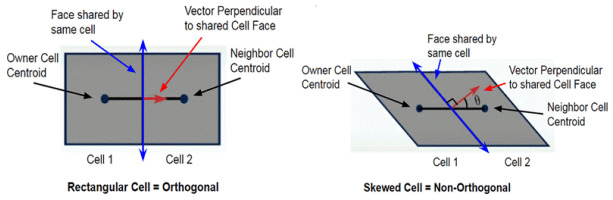

Figure 4.4: Non-orthogonality

4.3.2 Skewness

Skewness refers to the deviation of a vector connecting two neighboring cells from the midpoint of a connecting face. A high skewness can have a negative effect on the interpolation of cell-centered qualities, and add numerical diffusion.

4.3.3 Aspect ratio

Aspect ratio refers to the ratio between the largest and smallest dimension of a cell. In steady-state cases, high aspect ratio cells can affect coefficients in a pressure correction equation, and make it more computationally difficult to solve. In transient cases, high-aspect ratio cells in bad positions within the domain can significantly increase the Courant number, again increasing computational complexity. High aspect ratios can also cause instability in the solver if not aligned with the surrounding flow.

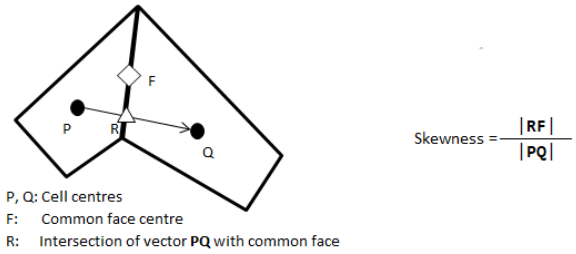

Figure 4.5: Skewness

Figure 4.6: Aspect ratio

4.3.4 Smoothness

Smoothness or growth rate describes the ratio between the sizes of neighboring cells. A sharp change in cell sizing can have a negative effect on the accuracy and stability of a solution, and as such a gradual change in cell size or aspect ratio should be maintained where possible.

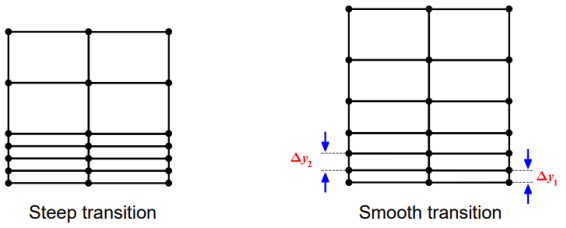

Figure 4.7: Smoothness

4.4 Meshing in OpenFOAM

OpenFOAM comes pre-packaged with three different meshing algorithms - blockMesh, snappyHexMesh and cfMesh. As both snappyHexMesh and cfMesh are similar in popularity, both were evaluated in this thesis. blockMesh is a meshing algorithm used for creation of structured meshes, and as such this thesis does not focus on evaluating this algorithm.

4.4.1 snappyHexMesh

snappyHexMesh is a meshing algorithm built into OpenFOAM used to create unstructured cartesian meshes. It requires an already existing mesh to build on and modify, so the blockMesh utility is often used to create the domain and define the base cell sizing, but other meshing software and utilities can be used to create a base mesh as well. As snappyHexMesh can only be used to create three-dimensional meshes, the extrudeMesh

OpenFOAM utility is used to extrude cells in a specified direction, making the number of cells in that direction equal to one.

snappyHexMesh works by first subdividing (castellating) an existing mesh in specified areas. These can be given by user-specified refinement regions or by their proximity to a surface. In the next step, the pointInMesh entry in the snappyHexMeshDict file is checked for coordinates to determine which cells should be kept, while cells that are separated from its position by geometry are removed. Remaining cells adjacent to a surface are then snapped to the geometry, attempting to follow its shape as closely as possible. Once snapped, prism layers are grown from specified surfaces. A number of smoothing operations are performed between these steps to ensure sufficient quality of the mesh.

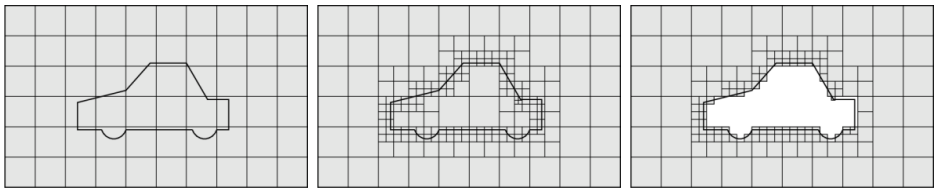

Figure 4.8: snappyHexMesh meshing procedure (from left to right): Base mesh, mesh castellation, cell removal

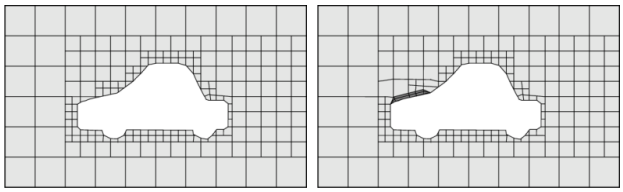

Figure 4.9: snappyHexMesh meshing procedure (cont.): Mesh feature snapping, prism layer addition

When using snappyHexMesh, the nCellsBetweenLevels entry, which sets the minimum number of cells to be used between each refinement level, was set to a value of 7 in order to achieve a smooth transition between each refinement level. Because the overall prism layer thickness is closely linked to the surface cell size, a constant cell sizing was used along the entire airfoil surface to avoid sharp changes in the prism layer thickness. The number of prism layers was set to 19 layers, as that is the maximum number of layers snappyHexMesh was able to create reliably while meshing this case without a significant degradation of the mesh quality. The number of mesh smoothing operations was optimized to achieve the best quality mesh possible, while minimizing the meshing time required. A close-up view of the mesh at the leading edge of the airfoil can be seen on fig. 4.10.

4.4.2 cfMesh

cfMesh is an open-source library for mesh generation released by Creative Fields under the GPA license that comes pre-packaged with many OpenFOAM distributions. It is capable of creating tetrahedral, cartesian and polyhedral cell types. An advantage of cfMesh over snappyHexMesh lies in its ability to create two-dimensional meshes, removing

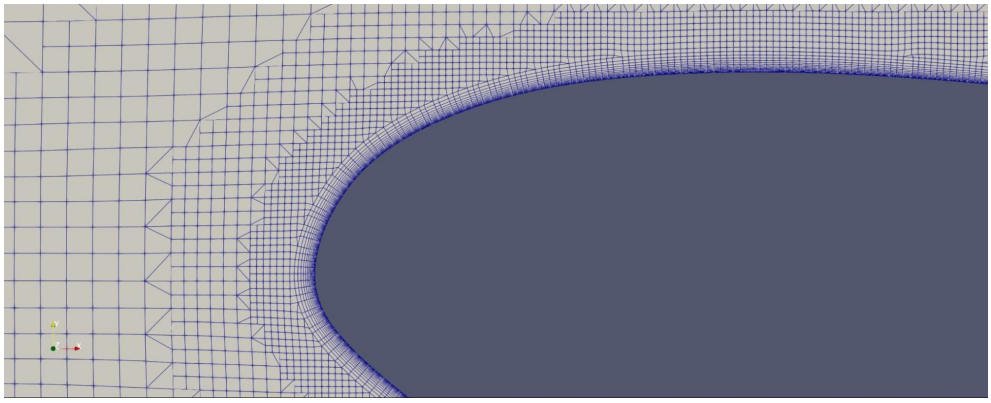

Figure 4.10: snappyHexMesh mesh detail

the need for other tools to be used and reducing meshing time considerably. Unlike snappyHexMesh, cfMesh does not require an already existing mesh to build on, so stl files are used to define the domain.

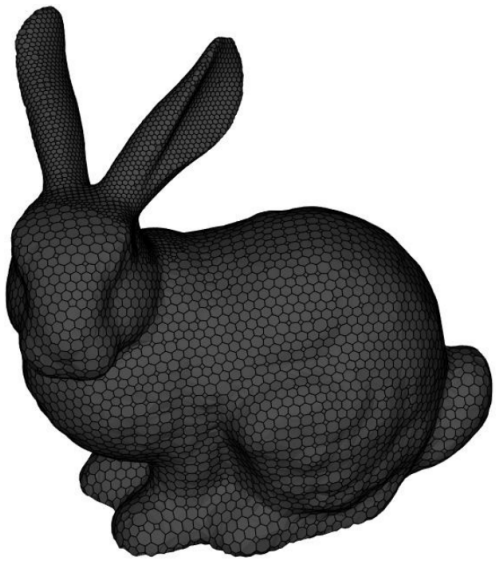

Figure 4.11: An example of a polyhedral mesh created using cfMesh

cfMesh offers only a very limited number of meshing controls to its user. While this approach makes cfMesh very easy to use, the drawback is that cfMesh offers its user the least amount of control over the final mesh out of all meshing approaches explored in this thesis. Another important aspect of cfMesh lies in its approach to creating prism layers. While most meshing algorithms create layers by growing new cells from the geometry surface, cfMesh instead creates prism layers by subdividing cells closest to the surface by the number of layers requested by the user. While this approach can be more robust in creating a large number of prism layers and avoids any issues caused by moving existing cells to make space for new prism layers to be grown, the implication of this approach is that a smooth transition between the prism layers and surrounding mesh cannot be achieved.

While using cfMesh to create the mesh, the airfoil surface cell sizing was deliberately kept larger than in other meshing algorithms in order to maximize the overall prism layer

thickness. The overall thickness of the surface cell sizing was set to extend to a distance of five centimeters from the airfoil surface to further improve the accuracy of flow prediction in this area. Similar to snappyHexMesh, cfMesh was not successful in creating more than 20 prism layers, with mesh quality dropping considerably when 20 layers were used, so the number of prism layers was set to 19. The number of mesh smoothing operations was increased significantly, but no noticeable difference in mesh quality or meshing time was observed. A close-up view of the mesh at the leading edge of the airfoil can be seen on fig. 4.12.

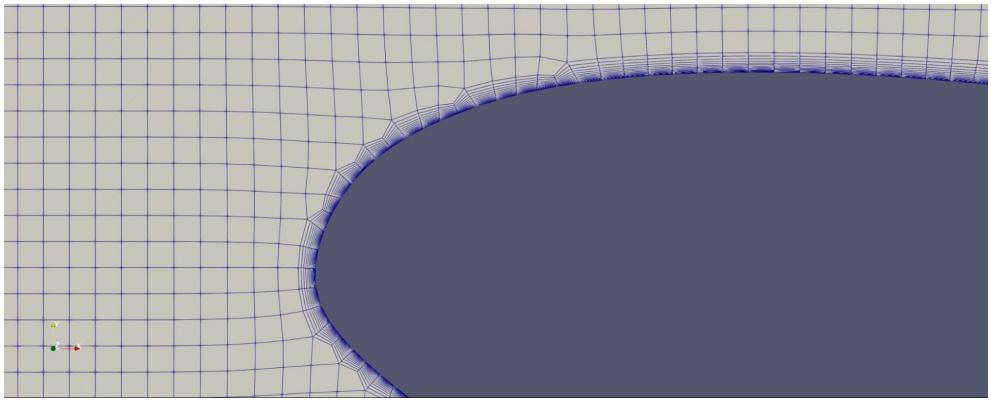

Figure 4.12: cfMesh mesh detail

4.5 Meshing in ANSYS CFD

The Fluent module of ANSYS has two distinct modes of operation - it can be used as a solver, but it can also be used to create three-dimensional meshes. Unlike OpenFOAM, ANSYS Fluent does not feature a utility that would allow for conversion of a three-dimensional mesh into a two-dimensional version of itself, and as such the meshing capabilities of ANSYS Fluent were not explored in this thesis. As such, ANSYS Mechanical module is used to create two-dimensional meshes to be imported into and used in ANSYS Fluent, so its meshing capabilities were evaluated instead. ANSYS Mechanical offers the user a wide range of meshing tools. It can be used to create either structured or unstructured meshes, and in two-dimensional space, it can create meshes consisting of either triangular or quadrilateral cells. A test case was performed to determine whether a triangular or square cell mesh should be used. This test showed only a negligible difference in resultant forces acting upon the airfoil, but a significant difference in the meshing speed where a significantly faster meshing speed was achieved when a triangular cell shape was used. As such, the triangular cell shape was used in all following simulations.

When meshing in ANSYS Mechanical, a much larger number of prism layers was used than in either meshing algorithm available in OpenFOAM. This higher number of prism layers allowed for a smoother transition between the prism layers and the surrounding mesh, but also served to further reduce the y+ values over the entire airfoil surface. The surface sizing was given a “bias”, which created a progressive change in cell sizing along the airfoil surface and which allowed for a finer mesh refinement in critical areas - the leading and trailing edges, without a significant increase in cell count. A close-up view of the mesh at the leading edge of the airfoil can be seen on fig. 4.13.

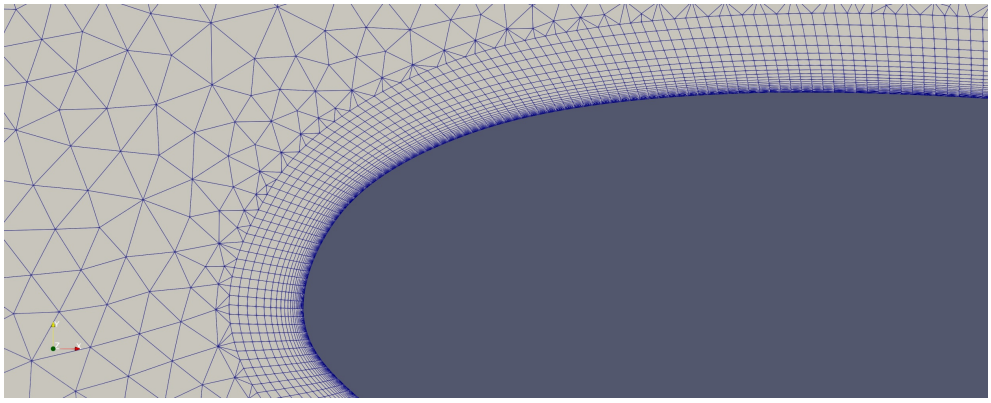

Figure 4.13: ANSYS Meshing mesh detail

4.6 Meshing summary

The averages of meshing time, number of cells, maximum non-orthogonal quality, average non-orthogonal quality, maximum skewness, maximum y+ and average y+ for each meshing solution are listed in tab. 6.1. As we can see, cfMesh meshes have the lowest number of cells. This is caused by the larger surface cell sizing used. Meshes created in ANSYS Mechanical on the other hand have the largest number of cells, as the number of prism layers is higher and the triangular cells have a smaller area as compared to the rectangular cells used by both snappyHexMesh and cfMesh.

Table 4.2: Mesh quality

	OF - sHM	*OF - cfMesh*	*ANSYS Mechanical*
Meshing time [s]	123.53	3.52	28.36
Cells	139365	110852	231925
Non-orthogonality (max.)	53.22	63.34	52.67
Non-orthogonality (avg.)	3.32	2.53	10.76
Skewness	2.75	2.88	2.92
y+ (max.)	3.88	3.59	<1
y+ (avg.)	1.32	1.19	<1

5 Solution

Once a mesh is created, the simulation can be setup. In this step, the

5.1 Turbulence modeling

Turbulence models are mathematical models used in CFD to describe the chaotic behavior of fluids. Because it would be too computationally expensive to resolve turbulent flow features at all scales, turbulence models were developed to approximate flows at a lower computational cost. Due to turbulence being one of the most challenging phenomena to model in fluid dynamics, several different approaches were developed. These approaches differ in computational cost and accuracy, where a tradeoff between these two factors must be made. As the computational cost increases dramatically with each step-up in approach, the least computationally expensive approach, the steady-state Reynolds-Averaged Navier-Stokes (RANS) approach, was chosen to be used in this thesis. While this approach can be the least accurate, due to the simplicity of the geometry and the case involved, its accuracy was deemed sufficient for the purposes of this thesis.

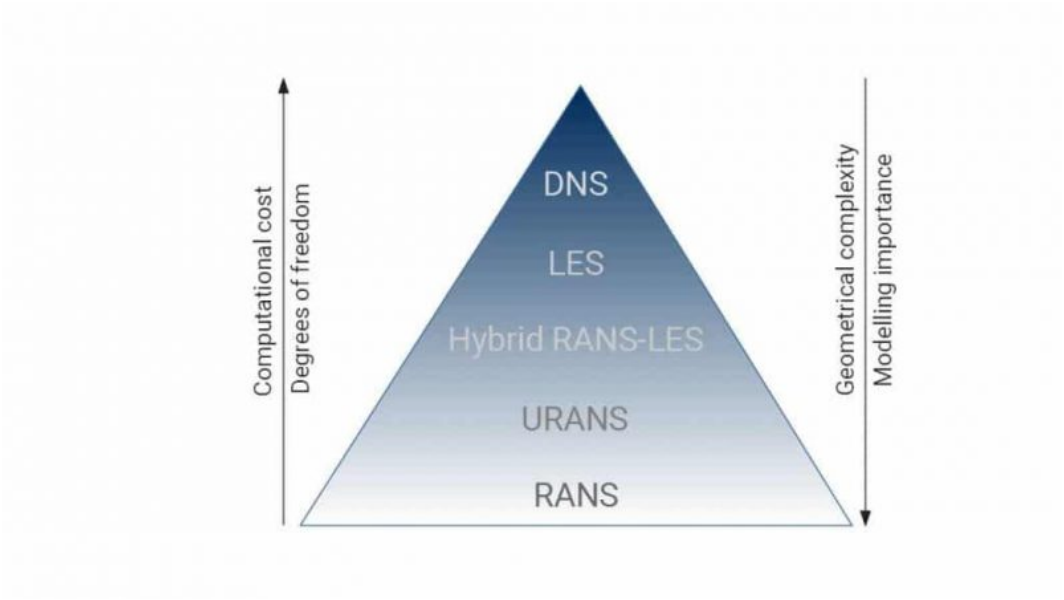

Figure 5.1: An example of a polyhedral mesh created using cfMesh

5.1.1 $k - \epsilon$ turbulence model

The standard $k - \epsilon$ turbulence model is a two-equation turbulence model that was introduced by B.E Lauder and D.B. Spalding. Being a two-equation turbulence model, it solves two transport equations that account for Reynolds Stresses using the Eddy Viscosity approach. The first transport variable k represents turbulent kinetic energy, whereas the second transport variable ϵ represents the turbulent dissipation rate. The standard $k - \epsilon$ turbulence model has been one of the most widely used turbulence models since its inception. A notable disadvantage of this model is its poor performance in problems involving adverse pressure gradients and large separations in complex flows.

A number of different variations of the model have been developed in an effort to address its shortcomings, such as the Realizable $k-\epsilon$ and RNG $k-\epsilon$ models. Unlike the standard version of this model, the Realizable $k-\epsilon$ model contains a new formulation for turbulent viscosity and a new transport equation for the dissipation rate. As a result, its performance in resolving strong adverse pressure gradients, separation and recirculation is improved over the standard model.[8]

5.1.2 $k-\omega$ turbulence model

The standard $k-\omega$ turbulence model is a two-equation turbulence model. Similar to the $k-\epsilon$ model, the variable k represents turbulent kinetic energy with the second transport variable ω representing the rate of dissipation per unit turbulent kinetic energy.

A very popular variation of the model is the $k-\omega$ SST model. In order to avoid the free-stream stability problem of the standard model, the SST variant combines the behavior of the $k-\epsilon$ and $k-\omega$ models in free-stream situations. Its advantages over the standard model lie in better prediction of flow separation and improved behavior in adverse pressure gradients. [9]

5.1.3 Spalart-Allmaras

The Spallart-Allmaras turbulence model is a one-equation turbulence model, developed to address the shortcomings of the standard $k-\epsilon$ turbulence model. Unlike the two-equation $k-\epsilon$ model, Spalart-Allmaras solves a single transport equation for the kinematic turbulent viscosity, to describe viscous eddy current flow. As Spalart-Allmaras only solves one transport equation per iteration, fewer calculations need to be performed per iteration and solution time can be decreased. It is particularly suited for solving flows around airfoils and turbine blades, but doesn't perform as well as other models in describing the decay of turbulent flows. Since its introduction, the Spalart-Allmaras model has seen wide adoption, especially in the aerospace, automotive and industrial sectors.[18]

In this thesis, the Realizable $k-\epsilon$, $k-\omega$ SST and Spalart-Allmaras models were chosen as the turbulence models to be used. This decision was made based on the popularity of these models as well as their suitability for use in this case.

5.2 Near-wall treatment

Near-wall treatment refers to the approach to turbulence modeling close to a wall within a boundary layer, where velocity gradients are highest. As boundary layers can have a major impact on the overall behavior of a flow around an object, an accurate prediction of their behavior is of critical importance. The approaches to this problem lie in either resolving the boundary layer, or in modeling of its effects on the surrounding flow through the use of wall functions. These approaches are sometimes referred to as Low Reynolds Number (LRN) and High Reynolds Number (HRN) respectively. Which of these approaches is used depends largely on the turbulence model used, as well as the mesh refinement within the boundary layer. A visual representation of a boundary layer-resolving and boundary layer modeling approach can be seen on fig.5.2.

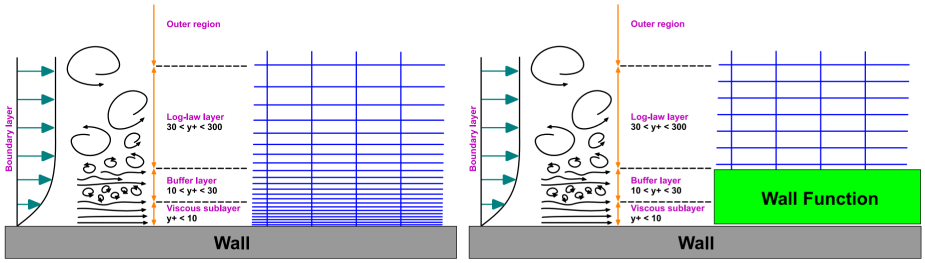

Figure 5.2: Visualization of boundary layer-resolving and boundary layer-modeling approaches [11]

To determine which approach should be used, a y+ value is calculated over an object's surface. y+ is a dimensionless parameter that describes the position of a cell center within the boundary layer profile. Its value is calculated by

$$y^+ = \frac{\rho U y}{\mu} \tag{5.1}$$

where ρ is the fluid density, U is the flow velocity at the cell centroid, y is the distance to the centroid and μ is the fluid dynamic viscosity.

As the approach to boundary layer modeling is different for every turbulence model, the appropriate y+ values to be used differ across turbulence models and CFD software. In ANSYS Fluent, appropriate y+ values for each turbulence model listed in ANSYS User Manual were used. As a large number of conflicting information was found across various sources in regards to the appropriate y+ values to be used for each turbulence model in OpenFOAM, the approximate ranges of $0 < \text{y+} < 5$ for a LRN approach and $30 < \text{y+} < 300$ for HRN approach were assumed for all[1] turbulence models.

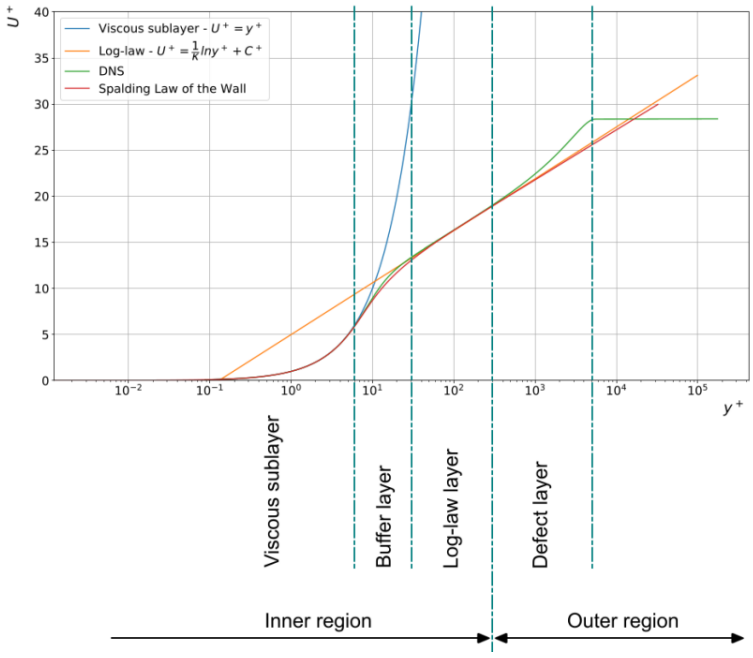

Figure 5.3: Turbulent boundary layer regions

[1]HRN approach does not apply to the Spalart-Allmaras model

5.3 Boundary conditions

Boundary conditions are applied to domain patches and geometry surfaces, and are used to describe how a fluid should interact with each surface.

As the target velocity is 0.15 Mach, the inlet velocity for all simulations was set to 51.45 m/s.

In order to change the Reynolds number, a change of the wind tunnel stagnation pressure was used in the reference study. To replicate this change, the fluid density was changed accordingly in all simulations. In ANSYS Fluent, the fluid density was set to $1.1315\frac{kg}{m^3}$, and the kinematic viscosity in OpenFOAM was set to PLACEHOLDER. Resultant Reynolds numbers in Fluent and OpenFOAM are calculated in equations 5.2 and 5.3 respectivelly.

$$Re = \frac{U_\infty c}{\nu} = \frac{U_\infty c \rho}{\mu} = \frac{51.45\,m \cdot s^{-1} \cdot 0.5842\,m \cdot 1.1315\,kg \cdot m^{-3}}{1.17894 \cdot 10^{-5}\,kg \cdot \quad m^{-1} \cdot \quad s^{-1}} = 1900614.6 \tag{5.2}$$

$$Re = \frac{U_\infty c}{\nu} = \frac{51.45\,m \cdot s^{-1} \cdot 0.5842\,m}{1.58 \cdot 10^{-5}\,m^2 \cdot \quad s^{-1}} = 1902347.5 \tag{5.3}$$

The slip boundary condition in OpenFOAM and the Specified Shear boundary condition in ANSYS Fluent were used on the upper and lower walls of the domain. This was done in order to avoid the need to create a large number of prism layers at these surfaces, reducing the overall cell count considerably. To assess the impact of this approach, a simulation was conducted in OpenFOAM at 14-degree airfoil angle. The difference between the two approaches amounted to 0.67% change in lift coefficient Cl of the airfoil. This error is relatively small at this angle, and was therefore considered acceptable in light of the benefits this approach offered.

Table 5.1: OpenFOAM boundary conditions

	inlet	*outlet*	*airfoil*	*walls*
p	zeroGradient	fixedValue	zeroGradient	slip
U	fixedValue	inletOutlet	fixedValue	slip
nut	zeroGradient	zeroGradient	nutLowReWallFunction	slip
k[2]	fixedValue	zeroGradient	kLowReWallFunction	slip
omega[3]	fixedValue	zeroGradient	fixedValue	slip
epsilon[4]	fixedValue	zeroGradient	fixedValue	slip
nuTilda[5]	fixedValue	zeroGradient	fixedValue	slip

5.4 General settings

Although air is a compressible medium, in free flows its compressibility does not become significant until velocities of approximately 0.7 Mach. As the peak velocity magni-

tude within the domain is not larger than 0.5 Mach at any airfoil angle, an incompressible solver was used for all simulations.

The selected discretization schemes were 2nd-order accurate.

6 Results

6.1 Mesh

The averages of meshing time, number of cells, maximum non-orthogonal quality, average non-orthogonal quality, maximum skewness, maximum y+ and average y+ for each meshing solution are listed in tab. 6.1. As we can see, cfMesh meshes have the lowest number of cells. This is caused by the larger surface cell sizing used. Meshes created in ANSYS Mechanical on the other hand have the largest number of cells, as the number of prism layers is higher and the triangular cells have a smaller area as compared to the rectangular cells used by both snappyHexMesh and cfMesh.

Table 6.1: Mesh quality

	OF - sHM	*OF - cfMesh*	*ANSYS Mechanical*
Meshing time [s]	123.53	3.52	28.36
Cells	139365	110852	231925
Non-orthogonality (max.)	53.22	63.34	52.67
Non-orthogonality (avg.)	3.32	2.53	10.76
Skewness	2.75	2.88	2.92
y+ (max.)	3.88	3.59	<1
y+ (avg.)	1.32	1.19	<1

6.2 Cl curve

The Cl curve displays the coefficient of lift of the airfoil for a given angle. The change is linear at angles of attack near zero, but declines as the airfoil approaches stall angle. At angles lower than critical angle, each program was able to predict forces action upon the airfoil with reasonable accuracy, however accuracy dropped considerably past the stalling point. The most accurate combination of meshing algorithm and turbulence model was cfMesh with the Spalart-Allmaras turbulence model.

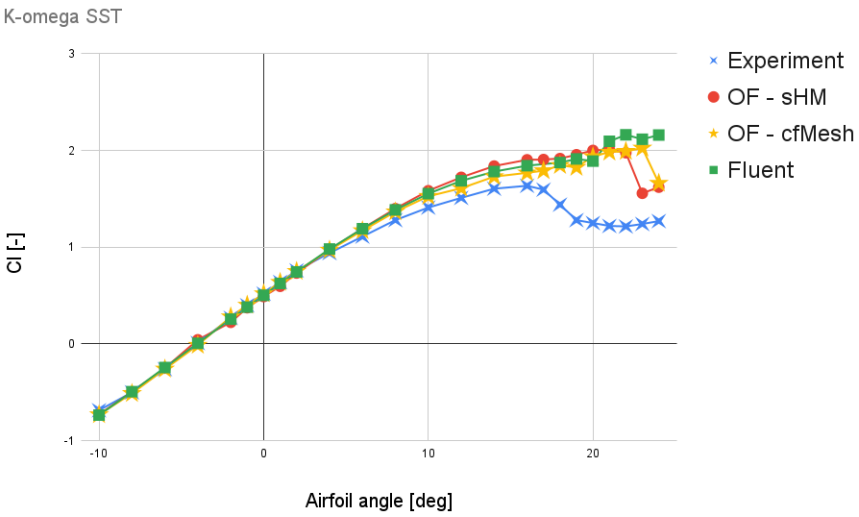

Figure 6.1: Airfoil lift curve for the k-omega turbulence model, Re=1.9e6

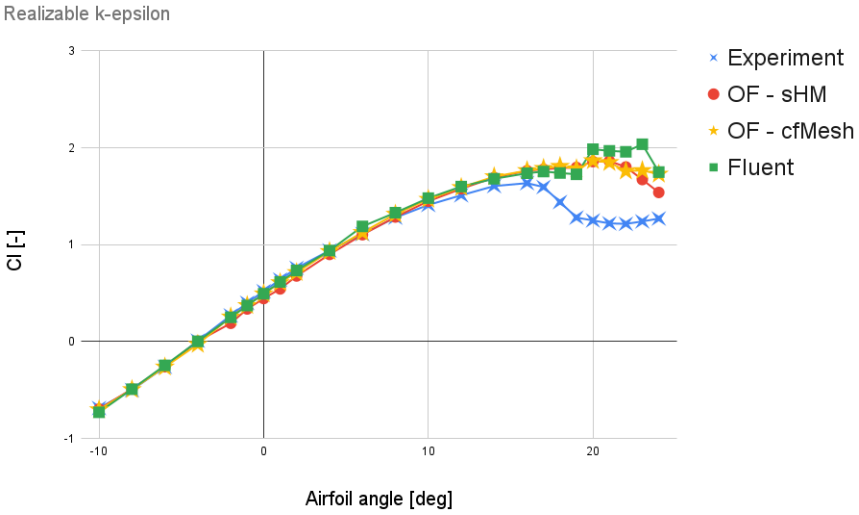

Figure 6.2: Airfoil lift curve for the k-epsilon turbulence model, Re=1.9e6

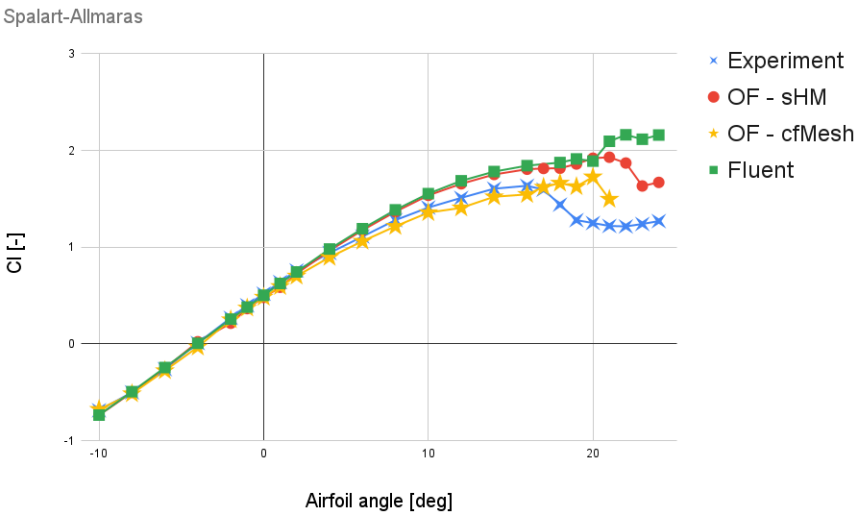

Figure 6.3: Airfoil lift curve for the Spalart-Allmaras turbulence model, Re=1.9e6

6.3 Cm curve

The Cm curve shows how pitching moment changes with airfoil angle.

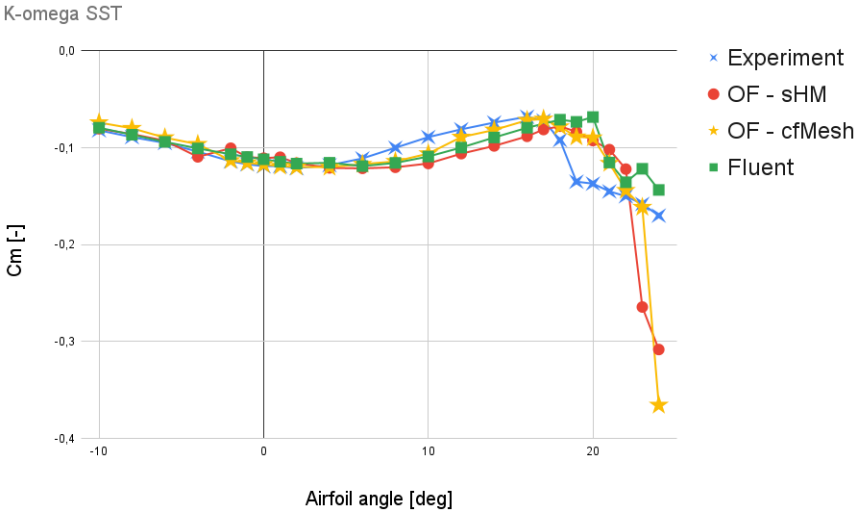

Figure 6.4: Airfoil pitching moment for the k-omega turbulence model, Re=1.9e6

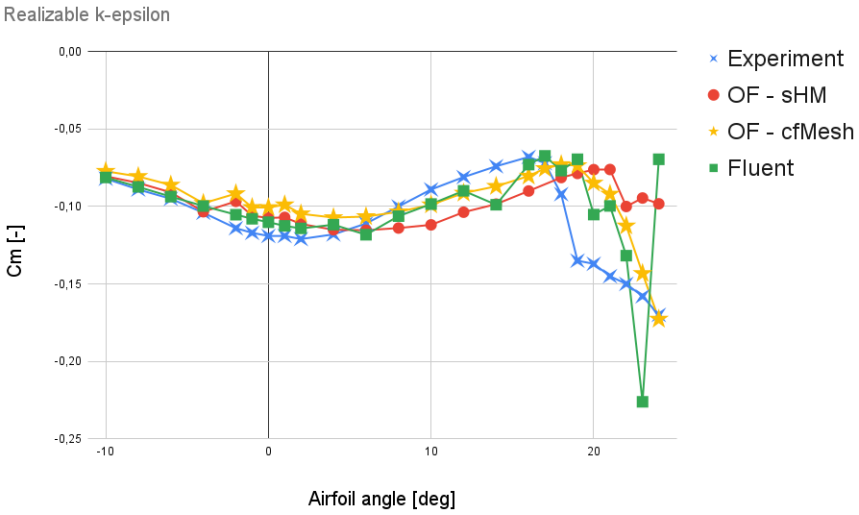

Figure 6.5: Airfoil pitching moment for the k-epsilon turbulence model, Re=1.9e6

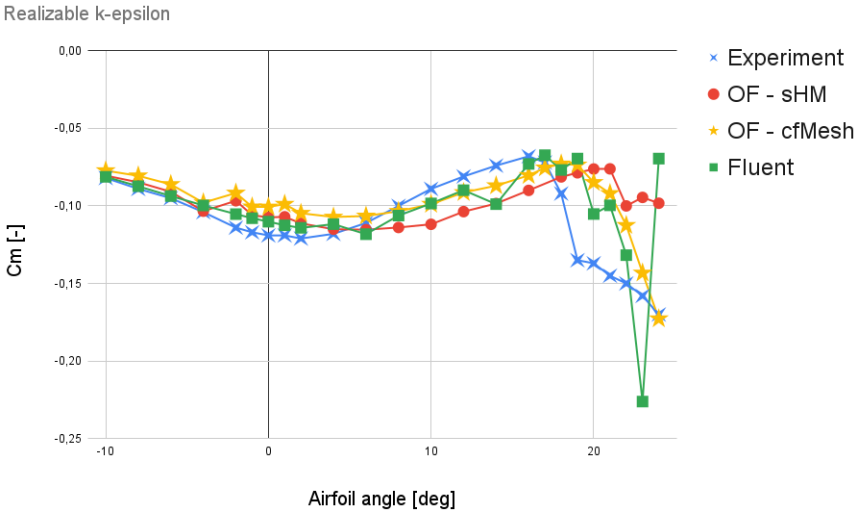

Figure 6.6: Airfoil pitching moment for the Spalart-Allmaras turbulence model, Re=1.9e6

6.4 Polar curve

The polar curve shows change in coefficient of drag with change in coefficient of lift.

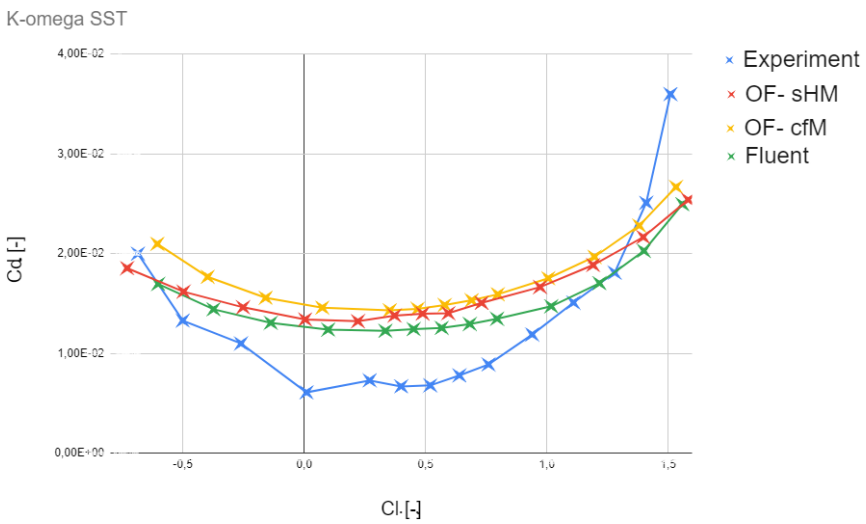

Figure 6.7: Airfoil polar curve for the k-omega turbulence model, Re=1.9e6

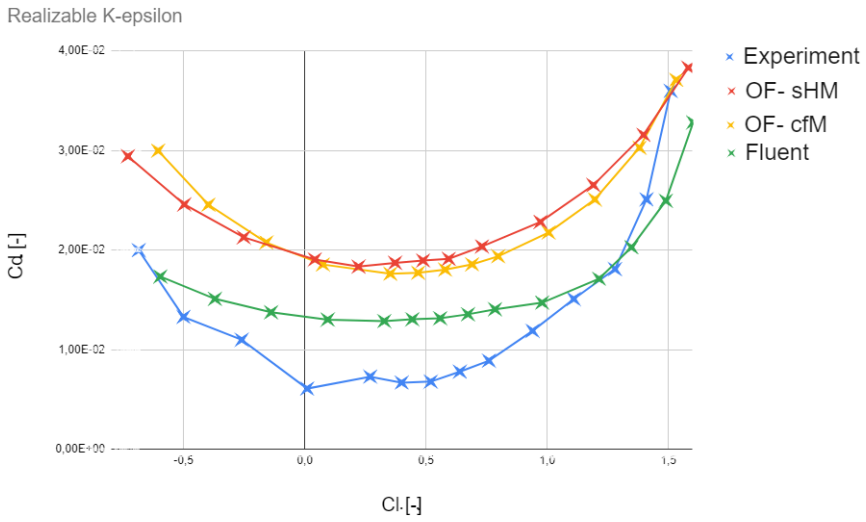

Figure 6.8: Airfoil polar curve for the k-epsilon turbulence model, Re=1.9e6

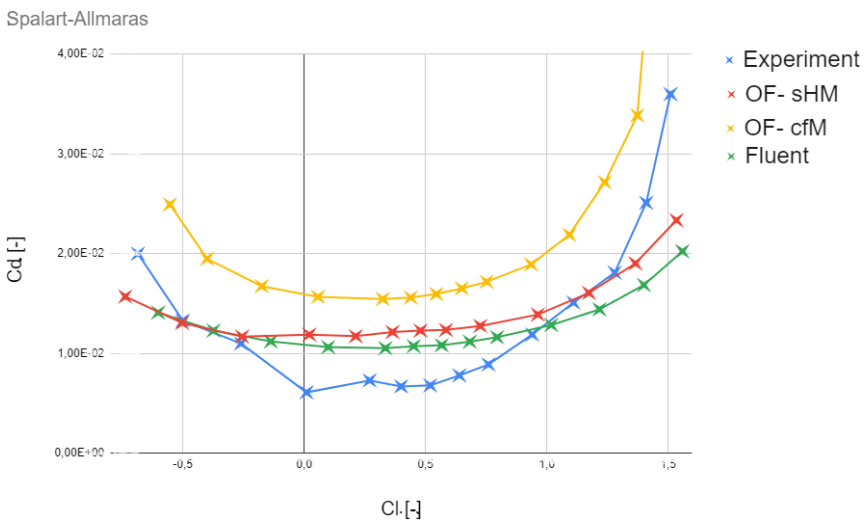

Figure 6.9: Airfoil polar curve for the Spalart-Allmaras turbulence model, Re=1.9e6

Conclusion

The focus of this thesis was a comparison of two different CFD solutions, one with commercial and one with open-source licence. It aimed to provide an overview into the capabilities and limitations of each software, and compare them with results obtained from an experimental study. The first chapter focusing on establishing what CFD is, its history and provided an overview of a number of available codes with different licensing. In the second chapter, airfoil nomenclature for this thesis was established, followed by a quick look into how forces acting upon them are determined in CFD. This was followed by a third chapter, which explained the format this thesis would follow. After an introduction into meshing in CFD, meshes were constructed using two different meshing algorithms in OpenFOAM - snappyHexMesh and cfMesh, followed by a mesh creation in ANSYS Mechanical. This was done to evaluate one of the most important parts of CFD analysis - mesh creation, as mesh has a significant impact on the result of the simulation. Once meshes were constructed, the solver setup was the next step. The simulations were performed using three different turbulence models, in order to determine how each would respond to different meshes.

Bibliography

[1] *Airfoil Nomenclature*. 2022. URL: https://aerotoolbox.com/airfoil-nomenclature/.

[2] John D. Anderson. *Fundamentals of Aerodynamics*. 6th ed. New York, NY: McGraw-Hill Education, 2017. ISBN: 978-1-259-25134-4.

[3] *Ansys Fluent: A History of Innovations in CFD*. Canonsburg, PA, 2006. URL: https://www.ansys.com/blog/ansys-fluent-history-of-innovations.

[4] *Dead Simple Flow Simulation Gets New Optimization Features. One of the simplest flow simulation workflows has added an adjoint solver to perform optimizations.* Mississauga, Canada, 2021. URL: https://www.engineering.com/story/dead-simple-flow-simulation-gets-new-optimization-features.

[5] *File:Screenshot OpenFOAM-2.1.x gnome-terminal.png*. San Francisco (CA), 2012. URL: https://commons.wikimedia.org/wiki/File:Screenshot_OpenFOAM-2.1.x_gnome-terminal.png.

[6] *Historical Perspectives of CFD and Some Applications*. Thiruvananthapuram, India. URL: https://www.iist.ac.in/sites/default/files/people/cfd_history.html.

[7] *How to run an Aerodynamic Simulation*. San Bruno, California, 2020. URL: https://www.youtube.com/watch?v=O_DNDxCtI_I.

[8] *K-Epsilon*. Munich, 2021. URL: https://www.simscale.com/docs/simulation-setup/global-settings/k-epsilon/.

[9] *K-Omega and K-Omega SST*. Munich, 2021. URL: https://www.simscale.com/docs/simulation-setup/global-settings/k-omega-sst/.

[10] *OpenFOAM*. Bracknell (England), c2023. URL: https://www.openfoam.com/.

[11] *OpenFOAM Introductory Training*. 2021. URL: https://doi.org/10.6084/m9.figshare.16783657.

[12] *SimScale Pricing & Plans*. Munich, c2023. URL: https://www.simscale.com/product/pricing/.

[13] *SimScale Releases Major User Interface Update for a Better Simulation Experience in the Cloud*. Munich, 2018. URL: https://www.simscale.com/press/user-interface-update-cloud/.

[14] *Structural Mechanics Simulation*. Munich, c2023. URL: https://www.simscale.com/product/structural-mechanics/.

[15] *What is Autodesk CFD?* San Francisco, c2023. URL: https://www.autodesk.com/products/cfd/overview.

[16] *What is CFD | Computational Fluid Dynamics?* Munich, 2023. URL: https://www.simscale.com/docs/simwiki/cfd-computational-fluid-dynamics/what-is-cfd-computational-fluid-dynamics/.

[17] *What is SimScale CFD?* Munich, c2023. URL: https://www.simscale.com/product/cfd/.

[18] *What Is the Spalart-Allmaras Turbulence Model?* San Jose, c2023. URL: https://resources.system-analysis.cadence.com/blog/msa2021-what-is-the-spalart-allmaras-turbulence-model.

7 List of symbols

C_l	Coefficient of lift	[-]
C_d	Coefficient of drag	[-]
C_m	Coefficient of pitching moment	[-]
Re	Reynolds number	[-]
c	Airfoil chord	[-]
U	Airspeed velocity	[m/s]
y	Distance	[m]
U_∞	Free-stream airspeed velocity	[m/s]
μ	Dynamic viscosity	[kg.m^{-1}.s^{-1}]
ρ	Kinematic viscosity	[m^2.s^-1]

List of figures

List of tables