ISRN LUTMDN/TMHP-15/5352-SE

ISSN 0282-1990

LUNDS UNIVERSITET

Triblade Wind Turbine

CFD Simulation

Farah El-Salem

Thesis for the degree of Master of Science in Engineering Physics
Division of Fluid Dynamics
Department of Energy Sciences
Faculty of Engineering | Lund University

Triblade Wind Turbine

CFD Simulation

Farah El-Salem

June 2015, Lund

This degree project for the degree of Master of Science in Engineering Physics has been conducted at the Division of Fluid Mechanics, Department of Energy Sciences, Faculty of Engineering, Lund University.

Supervisors at the Division of Fuid Mechanics were Dr Robert-Zoltan Szasz and PhD Ali-Al Sam.
Industrial supervisor at Winfoor AB was Rikard Berthilsson.
Examiner at Lund University was Dr Lei Wang.

The project was carried out in cooperation with Winfoor AB.

Thesis for the Degree of Master of Science in Engineering

ISRN LUTMDN/TMHP-15/5352-SE

ISSN 0282-1990

Efficient Energy Systems

Department of Energy Sciences

Faculty of Engineering, Lund University

Box 118, 221 00 Lund

Sweden

www.energy.lth.se

Abstract

In wind turbine design, the primary objective is to maximize the aerodynamic efficiency extracted from the wind. The number of blades, blade length, tower height and blade shape play an important role in wind turbine design. Winfoor AB has in co-operation with Lund University developed a new technology for wind turbine blades. This new design is intended to be compatible with standard wind turbines with benefits like low material cost, less fatigue and deformation.

Turbulent flow is a phenomenon encountered in many engineering applications such as vehicle, aircraft and wind turbines. By using computational fluid dynamics (CFD), the flow behavior can be predicted and important coefficients like lift and drag can be computed. Two-and three dimensional simulations are made on triblade wind turbines with the open source software OpenFOAM. Incompressible steady state RANS simulations are carried out for both two-and three dimensional cases.

First 2D simulations are carried out because they are faster. The main focus is to predict the lift and drag coefficients. It also includes testing the sensitivity to turbulence models and mesh resolution. The results from the simulations that are made in this thesis are then compared to data from other simulations.

In the three dimensional part, different cases are investigated. For each case, the pressure, viscous, lift and drag forces are computed on each blade. The effect of the angle of attack is investigated. The meshing is done in OpenFOAM where one mesh includes the blades with the beams. Another mesh is made where some geometry are excluded, i.e. only the three blades are included without the beams. In the last case, the possibility to optimize the structure, by altering the pitch angle of one the blades is investigated.

Sammanfattning

Målet med vindkraftverks design är att utvinna så mycket energi från vinden som möjligt. Antalet blad, bladlängd, tornhöjd och formen på bladen har en stor påverkan på hur effektivt ett vindkraftverk är. Winfoor AB har i samarbete med Lunds universitet utvecklat en ny teknik för vindturbinblad. Deras nya design är avsedd att vara kompatibel med vanliga vindkraftverk med förmåner som låg materialkostnad, mindre utmattning och deformation.

Turbulent flöde är ett fenomen som förekommer i många tekniska tillämpningar som fordon- och flygindustrin, även vindkraftverk. Genom numerisk strömningsteknik kan flödet analyseras och viktiga parametrar som lyft- och motståndskoefficienter beräknas. Både två- och tredimensionella simuleringar görs på triblade rotor blad med OpenFOAM.

Först genomförs 2D simuleringar eftersom de är mycket snabbare. Målet är att förutsäga lyft- och motståndskoefficienterna. Känsligheten mot turbulens modeller och nätet studeras också. Resultaten från simuleringarna som görs i denna avhandling jämförs sedan med data från andra simuleringar.

I den tredimensionella delen undersöks olika fall. För varje fall beräknas tryck, viskösa, lyft och motståndskrafterna på varje blad. Effekten av anfallsvinkeln undersöks genoma att testa olika vinklar. Nätet görs i OpenFOAM, där ett nät inkluderar både bladen och stagarna. I ett annat nät så utesluts en del av geometrin, endast de tre bladen är inkluderade utan stagarna. I det sista fallet undersöktes möjligheten att optimera konstruktionen genom att rotera ena bladet.

Acknowledgements

First of all, I want to express my gratitude to my supervisor at Winfoor AB, Richard Berthilsson, for giving me the opportunity to work on such an interesting and challenging subject.

Next, I want to thank my supervisor Ali-Al Sam for his big effort in helping me with my first steps in using OpenFOAM. Thank you for all your help, support and never ending patience you had with me. I also want to express my appreciation and gratitude to my supervisor Robert-Zoltan Szasz for his help with OpenFOAM and explaining aspects of CFD. Thank you for all your time and long discussion you had with me, even though my questions were trivial sometimes. Thanks to all of you in the group for being open minded,kind and making this thesis interesting and fun.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Wind power is the renewable energy source that is increasing most in the world. It has a small impact on biodiversity and is thus an important step towards reducing carbon emissions. In this thesis, CFD is applied on aerodynamic development of wind turbine rotor blade.

1.1 Background

Today, one can find wind turbines along coasts, fields and mountains. The design of the rotor blades has a big impact on the amount of wind captured, so it is important to develop an optimal design.

Rotor blades account today for nearly 25% of the total wind power cost. As longer rotor blades are being produced today, an increase of both cost and material occurs. This means that more material is needed for manufacturing of rotor blades. The stresses and strain that the rotor blades are exposed to lead to fatigue, deformation and eventually cracks.

Winfoor AB is a research-intensive company that runs research and development projects. The company has previously collaborated with the Faculty of Engineering (LTH) in various projects, but the focus is more on energy. Therefore Winfoor AB has developed new technology for rotor blades for wind turbines.

Today Winfoor AB has developed stronger, longer and lighter blades. This means more than 80% reduction in weight which results in lower cost. Fatigue and deformations is decreased and can be run at higher wind speeds than traditional blades. Instead of having one rotor blade, three parallel blades are linked together.

1.2 Purpose

The purpose of this project is to investigate the lift and drag force of the rotor blade by making two- and three dimensional analysis with OpenFOAM. Turbulent flow analysis will be done under different wind conditions, both low and high velocities are included. The analysis also includes studying the effect of small variation of the angle of attack.

Chapter 2

Wind Turbines

In this chapter, an introduction to wind turbines and their design is presented.

2.1 Modern Wind Turbine

A wind turbine is a mechanical device that converts the winds energy into electrical energy by using a generator. It consists of three main parts: tower, rotor blades and nacelle, see Fig.2.1. The tower is cylindrical and made of steel with a height about 75 m. The blades are often made of fiberglass-reinforced polyester or wood-epoxy. The length of the blades vary due to different designs, but the diameter can exceed 160 m. In the nacelle, typically there is gearbox, generator and other electrical control systems. When the blades start rotating, they turn a shaft inside the nacelle which goes into a gearbox. This increases the rotational speed and the generator converts the rotational mechanical energy into electrical energy [8].

There are different types of wind turbines, all have their advantages and disadvantages. The dominating type in the industry today is the Horizontal Axis Wind Turbine (HAWT), which is shown in Fig.2.1. It means that the rotating axis of the wind turbine is horizontal or parallel to the ground. Another type is the Vertical Axis Wind Turbine (VAWT) which means that the rotating axis stands vertically or perpendicular to the ground, see Fig.2.2. HAWT type produce more electricity than VAWT so it's more used in big wind applications. The disadvantage of HAWT is that it doesn't produce well in turbulent winds, which is one of the reasons why the tower is high since the wind near the ground is more turbulent than higher up. The wind speed is faster and more consistent at high altitudes. Ground level wind is interrupted and slowed by obstacles like trees and buildings. The VAWT type are used in smaller projects and in residential applications. The VAWT types are powered by wind coming from all 360 degrees unlike the HAWT type that has to be turned in the wind direction. For this reason, VAWT are good for conditions where the wind is not consistent [8].

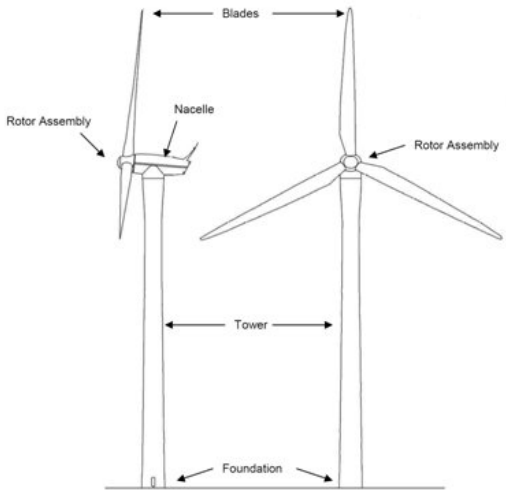


Figure 2.1: *Anatomy ofa wind turbine, HAWT type [22].*

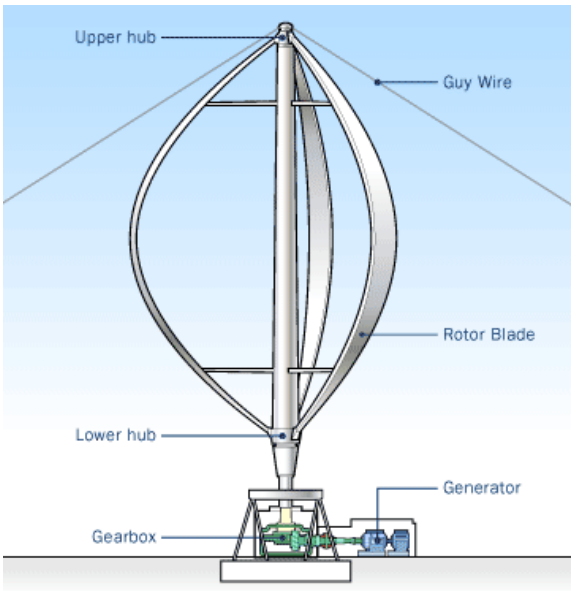


Figure 2.2: *Anatomy of a wind turbine, VAWT type [22].*

2.2 Wind Turbine Design

Besides considering wind conditions, there are other factors that play an important role in designing a wind turbine. The goal is to obtain maximum power extracted from the wind and this should be met by satisfying mechanical strength and economical aspects. The important factors are presented as follows:

Number of blades

Increasing the number of blades gives more efficiency. Each blade disturbs the air flow for the following blade, so one wants to use as few blades as possible to maximize the efficiency. Three blades are usually enough. To balance the turbine, at least two blades are used, but as mentioned previously, three blades work good and are therefore used in the industry today. Increasing the number of blades will also increase the economical cost. Wind turbine blades are manufactured with thicker root to withstand huge bending moment caused by axial wind loading, see Fig.2.3 [22].

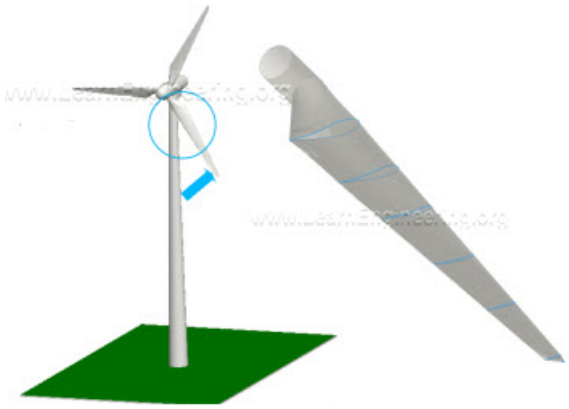


Figure 2.3: *Blade with thicker root to withstand bending moment induced [22].*

Blade length

Longer blades increase the aerodynamic efficiency, but it also has its limitations. One should not have too long blades because the deflection of the blade tip due to axial wind force will increase. This causes a collision between the blade and the tower, see Fig.2.4. Another limitations is that the noise produced by the turbine will be larger with longer blades. The noise is a function of the tip velocity which increases with longer blades [22].

Figure 2.4: *Blade bending due to loading [22].*

Tower height

Another important factor is the determination of a proper tower height. The power output is a strong function of the wind speed, so a small change in wind speed has a huge effect on the power extraction. Therefore it is desirable to have as high towers as possible, but difficulties in road transportation, economical cost and mechanical design limit the maximum tower height [22].

Blade shape

The last factor that has a huge impact on the aerodynamic performance is the blade shape and orientation of its cross section. As the blade moves, it will experience a different flow velocity than the actual velocity, this velocity is called the relative velocity. The flow velocity is uniform along the length of the blade, but it increases linearly along the blade. The angle and magnitude of the relative velocity will vary along the blade, see Fig.2.5. The highest velocity is on the tip and lowest on the root. Therefore the blade is rotated along its own axis so it can achieve the right orientation with varying wind condition. There is an optimum angle of attack that gives the highest lift force of the blade [22]. A close look at the aerodynamics of wind turbine blades is presented in the next chapter.

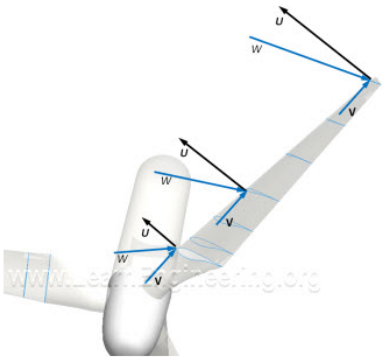


Figure 2.5: *The velocity is different at different sections (airfoils) along the blade [22].*

Chapter 3

Aerodynamics Of Wind Turbines

In this chapter, the theory for aerodynamics of wind turbines is presented.

3.1 Lift and Drag

The aerodynamics of wings are dependent on the topology of the wing, i.e. the shape of the wing determines how efficient it is in capturing the wind and extracting maximum power. Therefore the aerodynamics for one wing will be unique due to different shapes, but the fundamental concepts of this theory apply to all kind of wing shapes. The working principle of a rotor blade is the same as for an airplane wing [4]. Consider the wing shown in Fig.3.1.

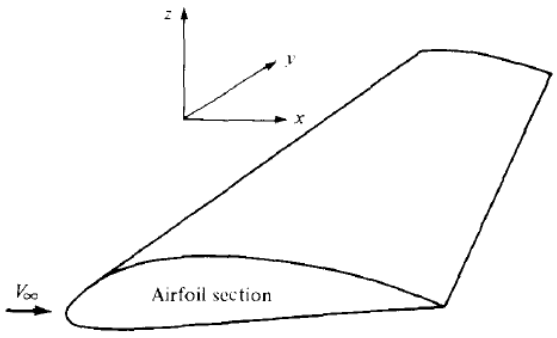


Figure 3.1: *Definition of an airfoil [4].*

The wing (rotor blade) extends in the y-direction and an airfoil is defined as the section that is cut by a plane parallel to the xz-plane. An airfoil is a structure with a specific geometry that generates mechanical forces due to the relative motion of the airfoil and the surrounding fluid (air). These airfoils are used to develop mechanical power. Each airfoil experiences different flow and maximum rotor power due to different rotational speed, height and width. Therefore it is convenient to study the aerodynamic concepts related to airfoils. The terms that describe an airfoil are presented in Fig.3.2 [4].

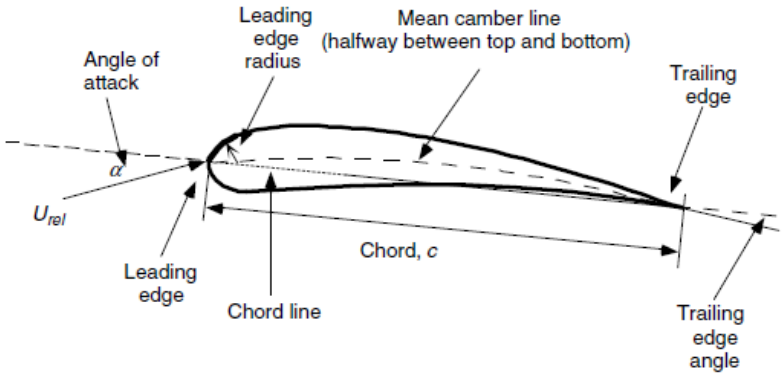


Figure 3.2: *Airfoil nomenclature [4].*

When the air flows over the airfoil, it produces a distribution of forces at the surface. The flow above the airfoil (the convex side) exhibits higher velocity than the flow below the airfoil (the concave side). This results in lower pressure at the convex side and higher pressure at the concave side of the airfoil. Viscous friction between the air and the airfoil surface will also slow down the air next to the surface. The pressure and shear stress vectors are presented in Fig.3.3 [4].

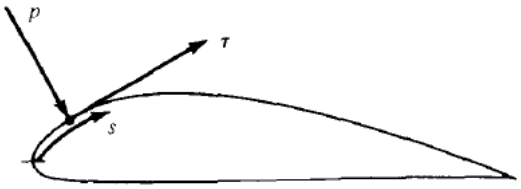


Figure 3.3: *pressure and shear stress per unit area along the surface s [4].*

The dimensions of the pressure p and shear stress τ are force per unit area. The pressure is acting normal to the surface and the shear stress parallel to the surface. By integrating p and τ along the surface s, a resultant force R and a pitching momentum M is obtained. This resultant is further split into a lift force L perpendicular to the oncoming flow and a drag force D parallel to the oncoming flow, see Fig.3.4. The lift force generates lift and is a consequence of the pressure difference between the upper and lower side of the airfoil. The drag force generates retardation and is a consequence of both viscous forces at the surface and pressure difference between the upper and lower side of the airfoil. The pitching moment act upon a surface perpendicular to the airfoil cross section [4].

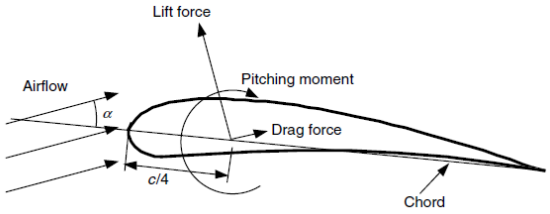


Figure 3.4: *Forces acting on the airfoil [4].*

3.2 Non-Dimensional Coefficients

The characteristics of flow can be described by non-dimensional parameters. The most important parameter is the Reynolds number which is defined by:

$$Re = \frac{Ul}{\nu} = \frac{\rho UL}{\mu} = \frac{Inertial force}{Viscous force} \tag{3.1}$$

where ρ is the density, μ the dynamic viscosity,ν the kinematic viscosity and l the length scale (in this case the chord length c). Other dimensionless parameters are the force, moment and pressure coefficients, which are functions of the Reynolds number. These coefficients can be defined for 2 or 3 dimensional objects and are usually determined in wind tunnel tests. Three dimensional airfoils have a finite span, which makes the force and moment coefficient affected by the flow around the end of the airfoil (end effects). For the two dimensional airfoils, infinite span is assumed which means that there is no flow around the end of the airfoil (no end effects). The coefficients for three dimensional airfoils are usually denoted by an upper case subscript and two dimensional by lower case subscript [4]. The two dimensional force coefficients are defined as:

$$C_l = \frac{2L}{\rho U_\infty^2 A} \tag{3.2}$$

$$C_d = \frac{2D}{\rho U_\infty^2 A} \tag{3.3}$$

where U_∞ the free stream velocity and A the projected area which the forces L and D act upon and c the chord length. Other important coefficients are the pressure C_p and moment C_m coefficients but are not covered in this thesis.

3.3 Lift to drag ratio

The lift to drag ratio is defined as the lift force divided by the drag force as follows:

$$G = \frac{L}{D} = \frac{C_l}{C_d} \tag{3.4}$$

The first term is for the total forces and the the second term for the coefficients (the forces per unit length). One wants to achieve a higher lift and lower drag, because it's better economically and improves the aerodynamic performance. The lift and drag coefficients increase with increasing angle of attack α. Usually the lift coefficient is much higher than the drag coefficient, but after a certain limit, stall limit, the drag coefficient increases dramatically, see Fig.3.5. At a specific angle of attack, the wing generates high lift, which give a high lift to drag ratio [2].

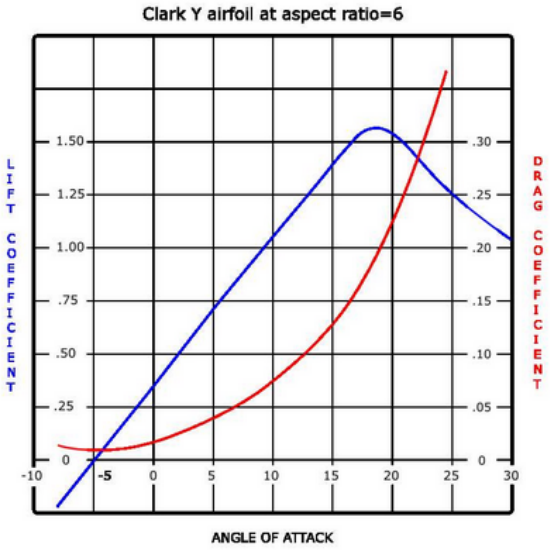


Figure 3.5: *Lift to drag ratio for Clark Y airfoil [9].*

Chapter 4

Turbulence Modeling

In this chapter, the theory and modeling of a turbulent flow is presented.

4.1 Turbulent Flow

Many engineering applications encountered today are turbulent. Flow around cars, airplanes, buildings and in engines are usually turbulent. There isn't an exact definition of turbulence, but it is characterized by its irregularity and chaotic behaviour. A turbulent flow consists of a spectrum of different scales. These scales consist of so called eddies of different sizes, where each eddy has a velocity and length scale. An eddy is a characteristic of a fluid that describe the swirling motion of the flow. Eddies exist for a limited time in space and are subsequently destroyed into smaller eddies, see Fig.4.1. The largest scales are of the order of the flow geometry (like boundary larger thickness) and are denoted v_0 and l_0. When the eddies interact with each other, a part of the kinetic energy is transferred to the smaller eddies. This transfer of kinetic energy from larger to smaller eddies is called the *cascade process*. At the smallest scales, viscous stresses become large and the kinetic energy is dissipated into thermal energy. This dissipation of the kinetic energy occur in the smallest scales, the so called *Kolmogorov's scale* denoted v_n and l_n [15].

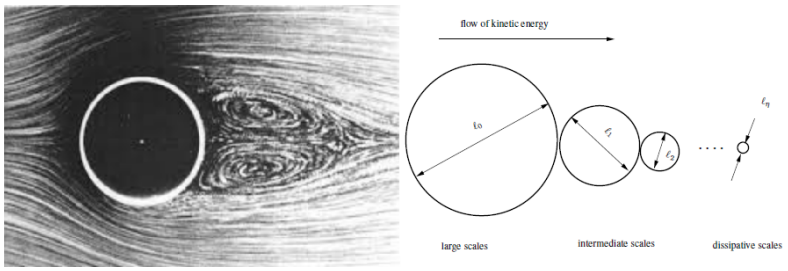


Figure 4.1: *To the left, flow around a cylinder where eddies are formed. To the right, length scales of different eddies in descending order [10].*

In addition to that a turbulent flow is unsteady, dissipative and three dimensional, it has other important properties like:

- Increased diffusivity. The exchange of momentum which delays the separation at bluff bodies is increased. Even the wall friction and heat transfer is increased.

- Large Reynolds numbers. The critical Reynolds number where transition from laminar to turbulent flow occurs is different for different flows. In pipes this value is about 2300 and in boundary layers about 500 000.
- Continuum, the smallest scale is much larger than the molecular scale so the flow can be treated as a continuum.

Before one can model a turbulent flow, it is important to distinguish between small and large scale turbulence. The large scales control the mixing and transport of the flow and are strongly influenced by the geometry of the flow. The small scales are influenced by how fast they receive energy from the large scale and the viscosity. The smaller scales are independent of the flow geometry unlike the large scales. Therefore it is important to make sure that both the small and large scales can be predicted (resolved by the governing equations) [15].

4.2 Governing Equations

The governing equations for a fluid are the Navier-Stokes equations and they include the continuity, momentum and energy equation as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0 \tag{4.1}$$

$$\left(\frac{\partial \rho u_i}{\partial t} + \frac{\partial \rho u_j u_i}{\partial x_j}\right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}\right) + \rho f_i \tag{4.2}$$

$$\left(\frac{\partial \rho E}{\partial t} + \frac{\partial \rho u_j E}{\partial x_j}\right) = -\frac{\partial p u_j}{\partial x_j} + \frac{\partial u_i \tau_{ji}}{\partial x_j} + \frac{\partial}{\partial x_j}\left(k\frac{\partial T}{\partial x_j}\right) + S_E \tag{4.3}$$

By using the constitutive relation that relates the pressure to density, one obtains a closed system of equations. This relation used is the state equation for an ideal gas as follows:

$$p = \rho R T \tag{4.4}$$

The equations above can further be simplified if one assumes that fluid is incompressible. Flows with the Mach number below 0.3 are considered to be incompressible [15]. The governing equations are then reduced to:

$$\rho\frac{\partial u_i}{\partial x_i} = 0 \tag{4.5}$$

$$\rho\frac{\partial u_i}{\partial t} + \rho\frac{\partial}{\partial x_j}\left(u_j u_i\right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\mu s_{ij}\right) \tag{4.6}$$

s_{ij} is the strain-rate tensor and is given by:

$$s_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{4.7}$$

Since there is no link between the energy, momentum and continuity equation, the energy equation doesn't need to be solved if isothermal conditions are assumed. As mentioned before, the turbulent flow is unsteady and the velocity and pressure are therefore highly oscillating with time. Many mathematical methods have been developed to describe and model a turbulent flow, like RANS method which is described in the next section. Other methods are LES and DNS, but they are not covered in this thesis.

4.3 Reynolds Averaged Navier Stokes (RANS)

One way to model turbulence is to use Reynolds decomposition. The flow quantities are decomposed into a mean part and fluctuating part according to:

$$u_i = \bar{u}_i + u_i' \tag{4.8}$$

where the mean part is denoted by a bar and fluctuating part by prim. By replacing the instantaneous variables in the governing equations with the decomposed variables one ends up with the Reynolds Averaged Navier Stokes (RANS) equations for an incompressible flow:

$$\rho\frac{\partial \bar{u}_i}{\partial x_i} = 0 \tag{4.9}$$

$$\rho\frac{\partial}{\partial x_j}\left(\bar{u}_i\bar{u}_j\right) = -\frac{\partial \bar{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left(2\mu\bar{s}_{ij} - \rho\overline{u_i'u_j'}\right) \tag{4.10}$$

The mean stress-rate tensor is defined as:

$$\bar{s}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) \tag{4.11}$$

The term $-\overline{u_i'u_j'}$ that pops up in RANS equations is unknown and is called the Reynolds stress tensor. Due to this unknown term, a closure problem occurs, where there are ten unknown terms and four equations. The four equations include the time averaged continuity equation and the three momentum equations. To close this system of equations, the Reynolds stresses need to be modeled [15]. The Reynolds stresses can be determined by the Boussinesq hypothesis by including the concept of eddy viscosity:

$$-\overline{u_i'u_j'} = 2\nu_T\bar{s}_{ij} - \frac{2}{3}k\delta_{ij} \tag{4.12}$$

where ν_T is the eddy viscosity and $k = \frac{1}{2}\overline{u_i'u_j'}$ the turbulent kinetic energy. Many turbulence models have been developed to model Reynolds stresses [15]. Models of these types are known as eddy viscosity models and are divided into four main groups:

- Algebraic/zero equation models
- One equation models
- Two equation models
- Reynolds stress models (RSM)

One of the models mentioned above that is not based on the Boussinesq hypothesis are the RSM model. In the RSM models, the eddy viscosity is not included and the Reynolds stresses are solved separately. Algebraic models use the Boussinesq approximation to calculate the stresses. The SpalartAllmaras model is a one equation model and solves one transport equation for the turbulent viscosity. The most used models are the two equation models $k-\omega$ and $k-\epsilon$. They solve two tranpsort equations, i.e. for the kinetic energy (k), dissipation (ϵ) and specific dissipation (ω) [15].

4.4 Boundary Layers

A boundary layer is the layer of fluid that originates in the vicinity of a surface due to the viscous forces distorting the flow. This boundary layer could be either laminar or turbulent, see Fig.4.2. The flow hits the airfoil at the leading edge and a stagnation point is formed. From this point, the laminar layer starts to develop. The pressure gradient is negative and the flow is accelerating. When the maximum velocity is reached (minimum pressure) , the transition point is reached. Then from this point, the air starts to decelerate and a turbulence layer is formed. The air is decelerating and eddies start to appear [19].

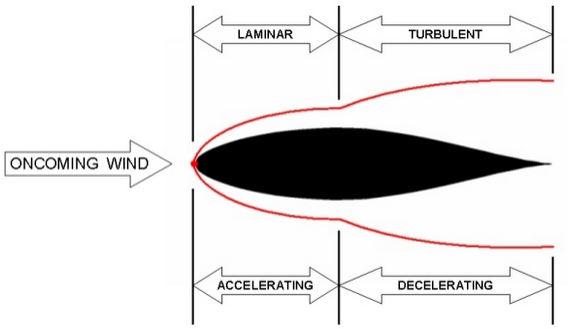


Figure 4.2: *Boundary layer around an airfoil [19].*

After the transition point and when the flow is fully turbulent, a thin viscous sublayer becomes apparent (see Fig.4.3). Further away, the so called logarithmic region follows. Unlike the viscous sublayer which is dominated by viscous diffusion, the logarithmic region is dominated by turbulent diffusion. Between these two layers, there is a buffer layer where viscous shear stresses are replaced by turbulent shear stresses [19].

The dimensionless distance of the wall is given by:

$$y^+ = \frac{u^* y}{\nu} \tag{4.13}$$

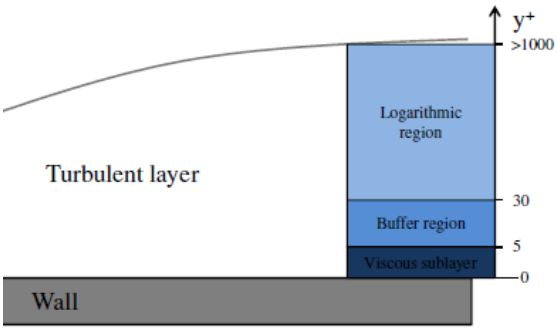


Figure 4.3: *Boundary layer in the turbulent domain divided by different sub-layers [19].*

where the friction velocity is given by $u^* = \sqrt{\frac{\tau_w}{\rho}}$, distance to the wall by y and the kinematic viscosity by ν.

4.4.1 Wall Functions

The boundary layer plays a significant role in determining the drag of an airfoil. The turbulent layer produces higher friction drag than the laminar layer. Therefore it is of great importance to model and resolve the boundary layer so one can obtain a correct solution. In order to resolve the viscous sublayer, one can make a finer mesh near the wall, but this will be time consuming as it increases the computational cost. Another way is to use wall functions provided that the mesh at the wall is coarse enough. In this case the viscous sublayer is not resolved and the wall functions is applied. The wall functions describe how the flow behaves in the logarithmic sublayer and are developed by semi empirical methods. The use of wall functions require that y^+ is higher than 30 (the first grid point is located in the logarithmic region). In case y^+ is low, then there is no need to add wall functions to resolve an already resolved region, i.e. the mesh is fine at the wall [5].

Chapter 5

Computational Fluid Dynamics

In this chapter, the theory of computational fluid dynamics is presented.

5.1 CFD

Computational Fluid Dynamics (CFD) is a computer based simulation of a fluid. It is a powerful technique with a wide range of applications in aerospace and vehicle industry, hydrodynamics of ships, weather prediction, blood flows in veins and many more.

CFD uses numerical methods and algorithms to solve and analyze the flow problem. There are mainly three important steps in computational modeling of a flow problem:

1. Problem definition
2. Mathematical model and discretization
3. Computer simulation

The physical problem is idealized and the quantities (velocity, pressure, temperature) that will be measured are identified. The idealized problem is then presented with a mathematical model including boundary and initial conditions. The fundamental basis in CFD problems are the Navier-Stokes equations, see (4.1)-(4.3). These equations are difficult to solve both analytically and numerically. Therefore assumptions are introduced to reduce the complexity and computational cost. Next step is discretisation of the region of interest (computational domain). There are many different discretisation methods, finite difference, finite element and finite volume methods. The finite volume method is often used in CFD simulations while the finite element method is used for solid mechanics. When the Navier-Stokes equations are discretized, a system of algebraic equations are obtained. These are finally solved with a numerical solver which gives an approximate solution. These numerical solvers are usually based on iterative methods. The numerical solvers introduce some errors and the goal is to minimize the errors so the approximate solution is comparable with the exact solution [17]. An acceptable solution in CFD must satisfy three main conditions:

1. Consistency
2. Stability
3. Convergence

A numerical scheme is consistent if the discrete numerical equation tends to the exact differential equation as the mesh size (represented by Δx and Δt) tends to zero. The numerical errors have the tendency to grow uncontrolled, therefore it is sensible to require that the solution is stable. The last requirement is the convergence, which means that the difference between the numerical and exact solution vanishes as the mesh size decreases [17].

5.2 Finite Volume Method

The Finite Volume Method (FVM) is the most used discretization technique in CFD and it's based on the control volume formulation of fluid dynamics. The domain is divided into a number of control volumes (cells, elements) and the variables of interest (velocity, pressure e.t.c) are located at the center of the volume (there are methods where the variables are stored in the nodes), see Fig.5.1. The grid defines the boundaries of the control volumes while the computational node lies at the center of the control volume. Depending on whether it is a two-or three dimensional case, the cells can be of different types. For a two dimensional case, most common are the triangle and quadrilateral, which correspond to tetrahedron and hexahedron in a three dimensional case [17].

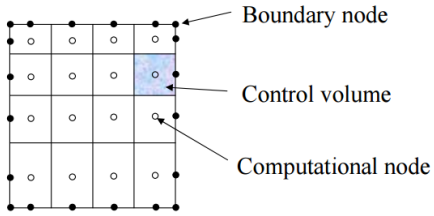


Figure 5.1: *An illustration of the computational domain (grid) [17].*

Then the differential form of the governing equations is integrated over each cell. One need the fluxes on the cell faces and in order to calculate these fluxes one needs to interpolate the variables to the cell faces.

5.3 Numerical Solver

When the algebraic equations are obtained, an appropriate solver needs to be chosen. The problem with the Navier-Stokes equations in (4.1)-(4.2) is that the source term $-\frac{\partial p}{\partial x_i}$ involves the pressure p. The equations system is undetermined, there are 3 equations, 4 unknowns and 1 constraint equation (depends if it is a 2D or 3D problem). To be able to solve the incompressible Navier-Stokes equations in (4.5)-(4.7), one needs to couple the velocity and pressure. Two commonly used solvers

are PIMPLE and SIMPLE. PISO stands for Pressure Implicit with Splitting of Operator and SIMPLE for Semi-Implicit Method for Pressure Linked Equations. PISO is used for transient problems and SIMPLE for steady state problems. Both methods iterate until all fields are correct [16]. Since only steady state problems are dealt with, the SIMPLE algorithm is explained briefly:

1. Guess an initial pressure field p^*.
2. Solve the momentum equation in Navier-Stokes to get the velocity field u^* from that pressure field.
3. Find a correction $p^{'}$ to the pressure field according to $p^{'}$=p^{**}-p^*.
4. Correct the velocity to obey the continuity.
5. At each step p^{**} and p^* during the iterative process, the algorithm can become unstable. Therefore under-relaxation can be used according to $p^{n+1} = \alpha p^{**} + (1 - \alpha)p^n$. α is a parameter and n the iteration step.

This is done until the desired solution is reached, see Fig.5.2.

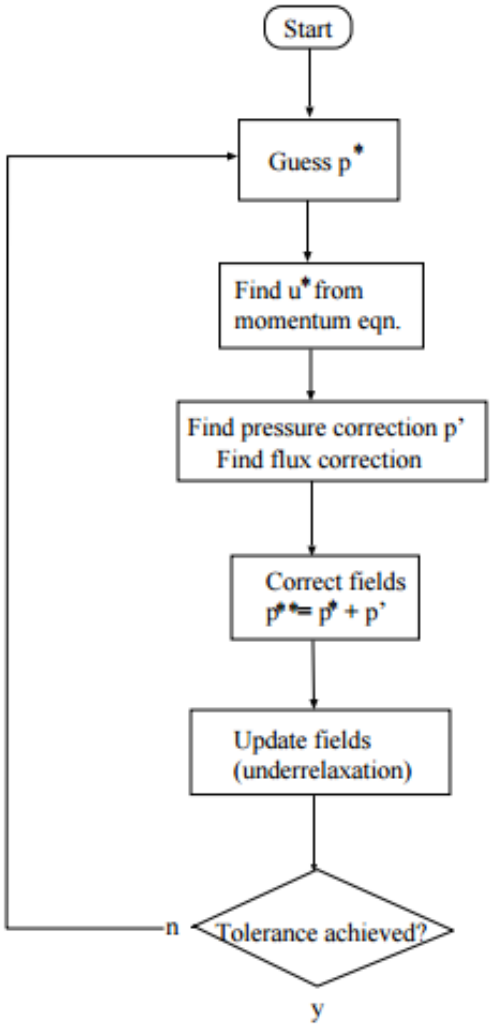


Figure 5.2: *The steps in SIMPLE algorithm [16].*

Chapter 6

OpenFOAM

In this chapter, the software OpenFOAM used in the thesis is presented. This is only an introduction to give a background, which makes it easier to understand the simulation set up in OpenFOAM. Only the solvers, schemes and turbulence models used in this thesis are explained briefly. Otherwise, there are various solvers, schemes and turbulence models to choose from.

6.1 OpenFOAM

OpenFOAM stands for Open Source Field Operation And Manipulation and it's a C++ library for complex simulations produced by the commercial company OpenCFD. It's an open source software package, which allows free use and modification of a CFD code. It's widely used by industrial and academic environments [20]. Besides that OpenFOAM is an open source software and free, its disadvantage is that it takes time to learn how to use it. Unlike other commercial CFD platforms, there is only limited documentation available for OpenFOAM. One basically starts learning from a tutorial, then a copy is made but modifications are made due to different geometries, initial conditions e.t.c.

6.2 File Structure

A simulation involves three major steps, pre-processing, solving and post-processing. In the first step, the mesh and simulation is set up, where turbulence model, physical properties, initial and boundary conditions are chosen. In the next step, the solver runs the simulation and in the final step the results are saved. The results are visualized with Paraview. The structure of a case is shown in Fig.6.1. The main directory (case) is divided into at least three sub-directories. These are the time (0-directory), constant and system directory. In the system directory, finite volume schemes and solutions are controlled. For example, one could choose a PISO or SIMPLE algorithm in the fvSolution file and a first order upwind scheme in the fvSchemes file. Time step and simulation time is given in the controlDict file. In the constant directory, the geometry and mesh are controlled. Properties like the kinematic viscosity and the turbulence model are given in this directory too. In the time directories, the initial and boundary conditions for the physical quantities are given [11].

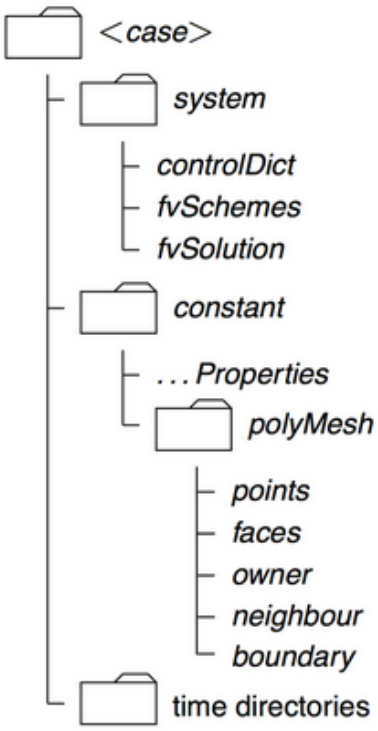


Figure 6.1: *File structure of a case in OpenFOAM [11].*

6.3 Finite Volume Schemes

OpenFOAM uses the finite volume method. The numerical schemes are different depending on the particular application one is dealing with [12]. The various numerical schemes used in this are described as follows:

- **Gradient Scheme**: This scheme models the pressure and velocity gradients, ∇p and ∇U. These are modeled with a Gauss linear upwind scheme.
- **Divergence Scheme**: Depending on the turbulence model that is used, each divergence term and its corresponding interpolation scheme has to be presented. A first order upwind scheme was used for both turbulence models.
- **Laplacian Scheme**: The Laplacian terms are modelled with an interpolation scheme and a normal surface gradient. The interpolation scheme is a linear Gauss scheme. The surface normal gradient is evaluated at the cell face.

6.4 Finite Volume Solvers

There are different numerical solvers for different applications/cases in OpenFOAM. The solvers are based on iterative methods. For instance, the solver pisoFoam solves transient incompressible turbulent flows and icoFoam transient incompressible laminar flows. Both these solvers use the PISO algorithm. In this thesis, steady state incompressible RANS problems are solved, so only the simpleFoam solver is used [18].

6.5 Turbulence Models

There are many turbulence models available in OpenFOAM, but only the ones that are used are presented.

Spalart-Allmaras

The Spalart-Allmaras model is a one equation model with the kinematic eddy viscosity as its transport variable. This eddy viscosity,$\tilde{\nu}$ is assumed to be the same as the molecular viscosity except in the near wall region, where viscous effects are dominating. The transport equation for $\tilde{\nu}$ is:

$$\frac{\partial(\rho\tilde{\nu})}{\partial t} + \frac{\partial(\rho u_i\tilde{\nu})}{\partial x_i} = G_\nu + \frac{1}{\sigma_{\tilde{\nu}}}\Big[\frac{\partial}{\partial x_j}\Big\{(\mu+\rho\tilde{\nu})\frac{\partial\tilde{\nu}}{\partial x_j}\Big\} + C_{b2}\rho\Big(\frac{\partial\tilde{\nu}}{\partial x_j}\Big)^2\Big] - Y_\nu + S_{\tilde{\nu}} \quad (6.1)$$

where G_ν is the production of the turbulent viscosity and Y_ν is the destruction of the turbulent viscosity. The kinematic viscosity is given by ν and constants by $\sigma_{\tilde{\nu}}$ and C_{b2}. The term $S_{\tilde{\nu}}$ denotes the source which is defined by the user depending on the specific application [21].

kkl-omega

Most turbulence models in CFD simulations assume that the flow is fully turbulent while in reality, a transition from laminar to turbulent flow occurs. Therefore the kkl-ω is brought to predict such a transitional flow. It's a three equation model with the laminar kinetic energy (k_L), turbulent kinetic energy (k_T) and specific dissipation ω as its transport variables [6]. The equations for these three variables are as follows:

$$\frac{Dk_T}{Dt} = P_{k_T} + R_{BP} + R_{NAT} - \omega k_T - D_T + \frac{\partial}{\partial x_j}\Big[\Big(\nu + \frac{\alpha_T}{\sigma_k}\Big)\frac{\partial k_T}{\partial x_j}\Big] \quad (6.2)$$

$$\frac{Dk_L}{Dt} = P_{k_L} - R_{BP} - R_{NAT} - D_L + \frac{\partial}{\partial x_j}\Big[\nu\frac{\partial k_L}{\partial x_j}\Big] \quad (6.3)$$

$$\frac{D\omega}{Dt} = C_{\omega1}\frac{\omega}{k_T}P_{k_T} + \Big(\frac{C_{\omega R}}{f_w} - 1\Big)\frac{\omega}{k_T}(R_{BP}+R_{NAT}) - C_{\omega2}\omega^2 + C_{\omega3}f_\omega\alpha_T f_w^2\frac{\sqrt{k_T}}{d^3} + \frac{\partial}{\partial x_j}\Big[\Big(\nu+\frac{\alpha_T}{\sigma_\omega}\Big)\frac{\partial\omega}{\partial x_j}\Big] \quad (6.4)$$

The various parameters in the equations above represents productions, dissipation, transport mechanism and constants. The parameters can be find in [6].

6.6 Meshing

OpenFOAM supports unstructured meshes of cells of any shapes. One have complete freedom on the cell shape, which give great flexibility for the generation of meshes around complex shapes. OpenFOAM has meshing software for both simple and complex geometries. The blockMesh utility is a multiblock mesh generator which generates meshes of hexahedra from a text configuration file. It is used for simple

geometries like cylinders and wedges. When it comes to more complex geometries, the snappyHexMesh utility is a great tool. A description of this mesh is given in chapter 7. Another mesh tool is the extrudeMesh utility, which generates a mesh by extruding cells from a surface of an existing mesh or from a surface mesh. This utility is used in the tutorial wingMotion in pimpleDyMFoam directory. First a snappy mesh is made (three dimensional) then the new two dimensional mesh is extruded from the snappy mesh [13].

In the *constant/polyMesh* directory, the mesh is described, see Fig.6.1. In this directory, the faces and points of the cells are given. Also the boundaries of the computational domain are described. A point is a location in 3-D space, defined by a vector in units of meters (m). The points in the domain are given by a list in the points file, where each point is referred to by its label. A face on the other hand is an ordered list of points, see Fig.6.2. Each two neighbouring points are connected by an edge. The direction of the face normal vector is defined by the right-hand rule. The faces are compiled into a list in the face file where each face is referred to by its label. The face are of two types, internal and neighbour faces. Internal faces connect only to two cells. Neighbour faces only connect to one cell since they coincide with the boundary of the domain [13].

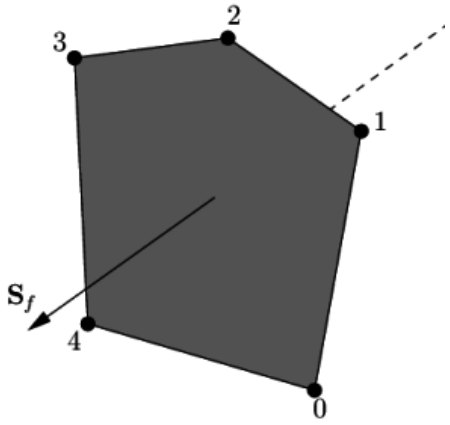


Figure 6.2: *A face of a cell [13].*

6.7 Boundaries

The boundaries in OpenFOAM are not just treated as a geometric entity, but an integral part of the solution and numerics of the boundary conditions. To be able to apply boundary conditions, the boundary is broken into a set of patches, i.e regions. These patches are of different types with different attributes presented in a hierarchy as shown in Fig.6.3. The base types describe geometric boundaries and are given in the constant/polyMesh directory. In this directory there is a file boundary, which contains all the different patches (boundaries) of the geometry. The primitive and derived types are numerical patch types which describe the numerical condition assigned to a field variable (pressure, velocity e.t.c.) on the patch. These boundaries are located in the field files (0-directory) where the boundary and initial conditions are given. Only a few patches that are used in the thesis are described [14].

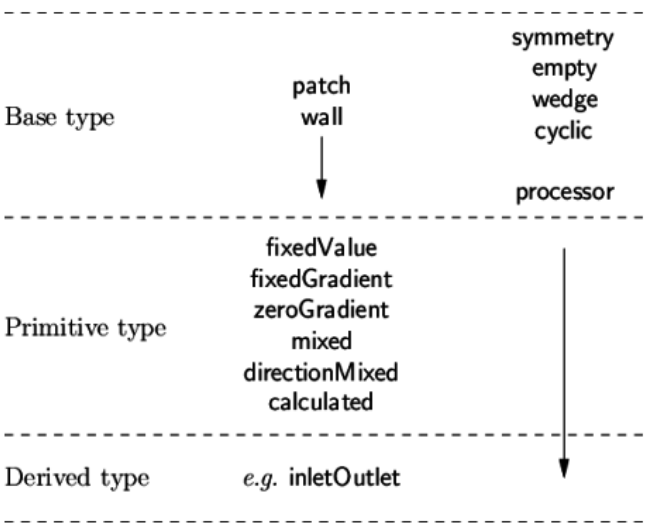


Figure 6.3: *Hierarchy of the boundary condition types in OpenFOAM [14].*

Base Types

- Inlet and Outlet: Basic types where no geometric or topological information is required.
- wall: This patch type is chosen when a certain patch coincides with a wall. This is specially required for turbulence modeling, where the distance to the wall is important.
- empty: This patch type is needed when two- or one dimensional problem is solved. OpenFOAM only generates geometries in three dimensions. Using this patch type, it is possible to create two-dimensional geometries by specifying an empty condition on each patch whose plane is normal to the third dimension. If one wishes to work with a one dimensional problem, en empty condition is applied on the patch whose plane is normal to the second dimension.

Primitive Types

- fixedValue: Value of the variable field (velocity, pressure e.t.c) is specified.
- zeroGradient: Normal gradient of the variable field is zero.
- calculated: The variable field is calculated from other fields.
- freestream: The value of the variable field is freestream, i.e the value far away from the obstacle

Derived Types

- inletOutlet: The value of the velocity and pressure switches between fixedValue and zeroGradient depending the direction of the velocity.

An example from a velocity field file is demonstrated in Fig.6.4. The patch name is inlet (base type) and is of type fixedValue with the value (27.778 0 0), which states that the velocity is in the x-direction.

```
inlet
{
    type            fixedValue;
    value           uniform (27.778 0 0);
}
```

Figure 6.4: *Example Boundary Condition [14].*

Chapter 7

Methodology

In this chapter, the method and simulation set up is presented. First, the model is described and then the problem that is going to be solved is formulated.

7.1 Problem Description

The model is a HAWT type wind turbine consisting of three rotor blades. Each rotor blade include three parallel blades that are linked together with plates and beams as shown in Fig.7.1. The three major blades have a NREL s831 profile, see Fig.7.2. A sample blade has a length of 60 m. The blade is made of epoxy and and weighs 3700 kg.

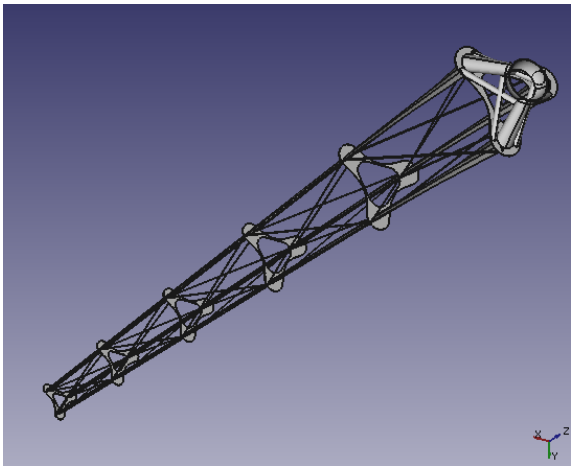


Figure 7.1: *Triblade built as a 3D framework, giving a very light and strong structure that is ideal for wind power. This is a model constructed in CAD.*

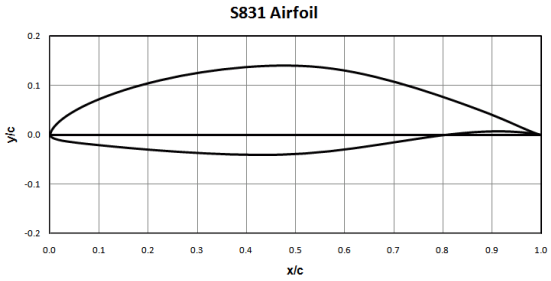


Figure 7.2: *NREL s831 airfoil profile [3].*

Winfoor AB has carried out some simulations in OpenFOAM for the uni- and triblade cases, where the uniblade case is included as a reference case to the triblade case. Uniblade denotes the case with only one airfoil. Many turbulence models have been tested: Spalart-Allmaras, kkl-ω, kω-SST and Launder Sharma. Both low and high Reynolds number simulations were carried out. Unfortunately , convergence issues were encountered for some of the tested models. The cases that converged gave an over estimated value of the drag coefficient. That's because the laminar sublayer near the wing wasn't resolved enough and most of the turbulence models assume that the flow is fully turbulent. The turbulence model that gave the best results was the transition model kkl-ω. When it comes to the triblade case, the simulations showed divergence.

As a result, first the configurations in the report that was given by Karl Hylen were evaluated to asses the sensitivity of the results to the choice of turbulence model [7]. In addition to the turbulence model, the mesh sensitivity was also tested. Computations have been carried out both on the mesh used in Karl's report and newly generated meshes with different levels of more refinement. A more detailed description of the method and mesh is given in the following two sections.

7.2 Approach

The different steps are presented for both the two- and three dimensional cases.

Two Dimensional Case: Uniblade

1. First, the ability to reproduce previous findings was tested by computing the same set up as in the report [7]. Additionally, Reynolds sensitivity was tested by simulating cases with different wind velocities.

2. To see how sensitive the turbulence models are to change in initial conditions, the same calculations are made but with different initial values.

3. A snappyHexMesh is made in OpenFOAM which gives possibility to refinement. The snappy mesh is then compared to the mesh used in the previous computations. Different mesh levels are tested for different velocities, to investigate how the mesh quality effects the results. Then a sensitivity analysis is made for Spalart-Allmaras and kkl-ω models one the same mesh, to investigate the sensitivity to the turbulence models.

Two Dimensional Case: Triblade

1. The same procedure that was used for the uniblade case is carried out for the triblade case. The same models are used and tested for both low and high velocities.

2. To avoid over- or under predicting the drag force, the sensitivity to the wall functions were tested. This is done for both Spalart-Allmaras and kkl-ω.

3. The mesh sensitivity is tested. The same procedure that is carried out for the uniblade is also carried out for the triblade.

Three Dimensional Case

In the three dimensional case, the forces on the blades and the angle of attack are investigated. Spalart-Allmaras turbulence model is tested for high velocities. The mesh is made with the snappyHexMesh utility.

1. First the simulations are carried out with a base mesh, which includes the major blades and the beams. Then the pressure, viscous, lift and drag forces are calculated for each blade. To maintain the computational effort in reasonable levels, only a 10 m section of the tri-blade structure (between 42 and 52 m in radial direction) is modelled.

2. The same calculations are made but with a different mesh where only the three blades are included. This is done to see how including or excluding the beams affect the results.

3. To further save computational time, the blade section was not rotated, instead, the inlet velocity was prescribed to have the desired angle of attack at the middle of the section. The modeled part being located at an outer radius, the changes in the real angle of attack are small (within 1 degree). Nevertheless, to evaluate the sensitivity of the results to changes in the angle of attack (which does happen even in real cases) the base case was compared with two cases where the angle of attack was altered by +/- one degree.

4. The initial computations revealed that the blade located downwind being parallel to the upwind ones has a sub-optimal angle of attack, since the upstream blades deflect the flow. To evaluate if the structure can be optimized to account for this deflection a new case has been set up, where the downstream blade has been pitched to have the correct angle of attack.

7.3 Assumptions and Limitations

To be able to perform this thesis in a reasonable amount of time and reduce the possible sources of errors a few assumptions and limitations had to be made. They are listed below:

- The flow is assumed to be incompressible, steady state simulations. No time dependent cases investigated.
- Two-and three dimensional cases carried out to save computational resources and to make the simulations faster.
- End effects not studied in three dimensions due to limited time of only 20 weeks.

7.4 Meshing

Two different meshes are used in the simulations. The gmsh provided by Winfoor AB and the SnappyHexMesh by OpenFOAM. The gmsh for the triblade consists of the three major blades and the three minor blades. The SnappyHexMesh described here is for the uniblade. An overview is given for the snappy mesh as follows:

SnappyHexMesh

The snappyHexMesh is a mesh generation utility provided by OpenFOAM. It generates 3-dimensional meshes consisting of hexahedra and split-hexahedra from triangulated surface geometry. To run the SnappyHexMesh, the following is required:

- Surface data files in STL- or object format in the $constant/triSurface$ subdirectory of the case.
- A background hexahedral mesh generated by $blockMesh$.
- A configuration file named snappyHexMeshDict located in the *system* subdirectory of the case,all the settings to control the various stages of the mesh.

The objective with the snappyHexMesh is to generate an oct-tree structured mesh surrounding an object described either by a STL-surface or object-file.

The configuration file consists of five main sections:

1. geometry specifications
2. CastellatedMesh controls
3. snapControls
4. addLayersControls
5. meshQualityControls

In the geometry section, the geometry is prescribed as:

```
geometry
{
    wing_5degrees.obj
    {
        type    triSurfaceMesh;
        name    wing;
    }
```

Figure 7.3: *Geometry given in the snappyHexMeshDict file.*

In the first meshing stage castellatedMeshControls, feature, surface and volume mesh refinements are prescribed. The initial block mesh is refined by cell splitting and removal of cells. The second meshing stage is the snapControls where patch faces are projected down onto the surface geometry. The last meshing stage is the addLayersControls where layers of cells are added to a specified set of boundary patches, in this case on the wing. Last but not least, the mesh quality is constantly monitored so no errors occurs in the meshQualityControls. The same is done for the triblade, the only difference is that instead of having one object-file, one uses three. The pictures for the snappy mesh is presented in Fig.7.4, Fig.7.5 and Fig.7.6.

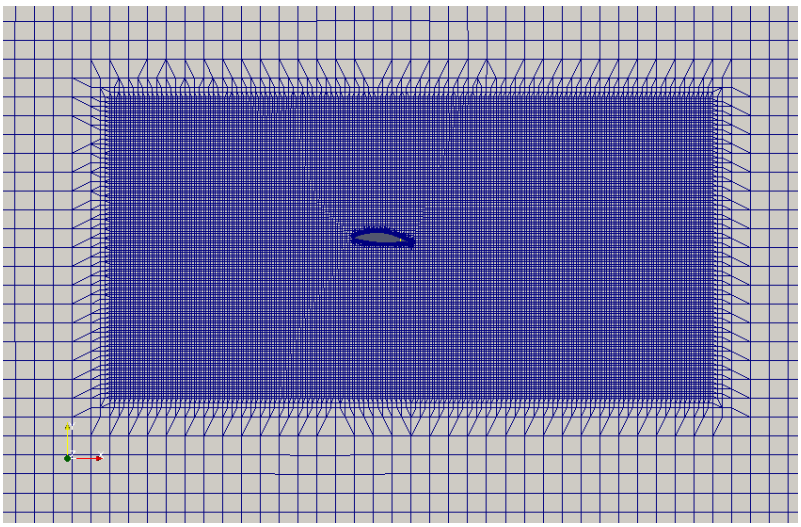


Figure 7.4: *The snappy mesh for one blade, this is just a zoomed region.*

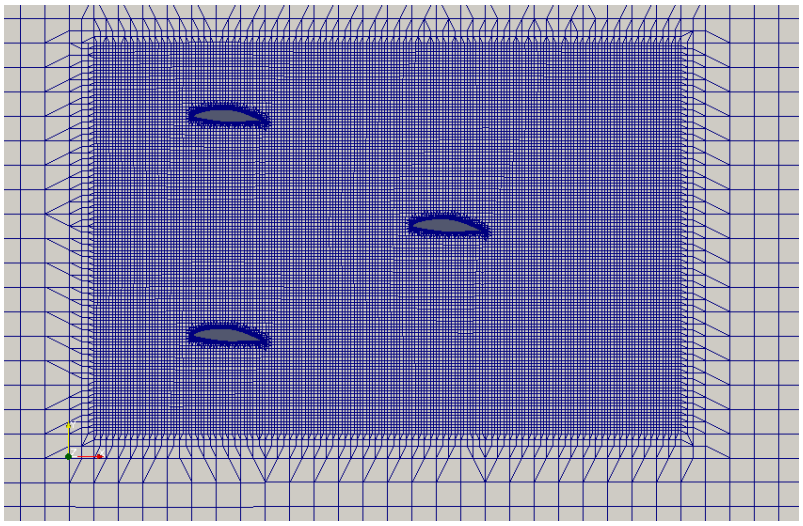


Figure 7.5: *The snappy mesh for three blades, this is just a zoomed region.*

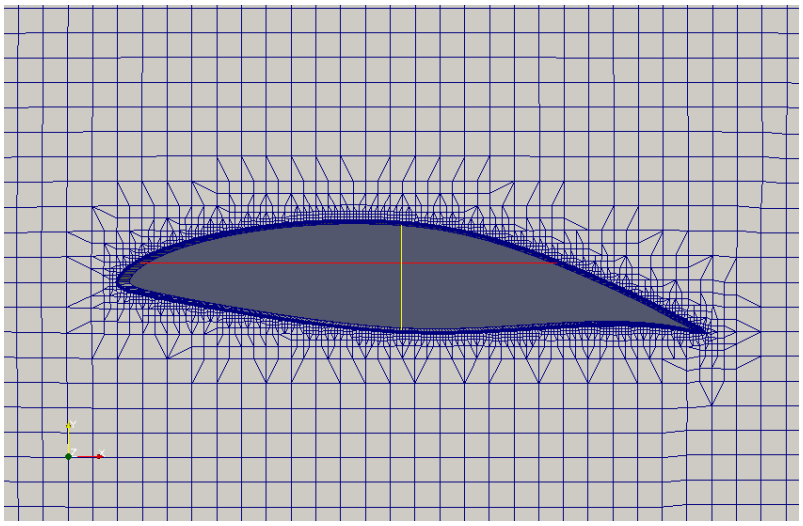


Figure 7.6: *A closer look of the mesh near the airfoil.*

7.5 Simulation Setup

As mentioned before, it is important to have appropriate initial and boundary conditions. Unfortunately, initial conditions are not known in before hand. They are usually guessed or estimated in some way or another. The domain consists of seven boundaries (patches). These are inlet,outlet,up,down,walls,front and back. Picture a cube with six sides (inlet, outlet, up, down, front, back) and the airfoil (walls) placed in it, see Fig.7.7. The simulation set up is presented for Spalart-Allmaras and kkl-ω. Each model has a different set up of boundary conditions.

Spalart-Allmaras

This case is presented for the velocity 90 m/s.

The velocity and pressure are denoted by U and P. The kinematic viscosity are denoted by ν and $\tilde{\nu}$. The pressure field is considered as relative freestream pressure

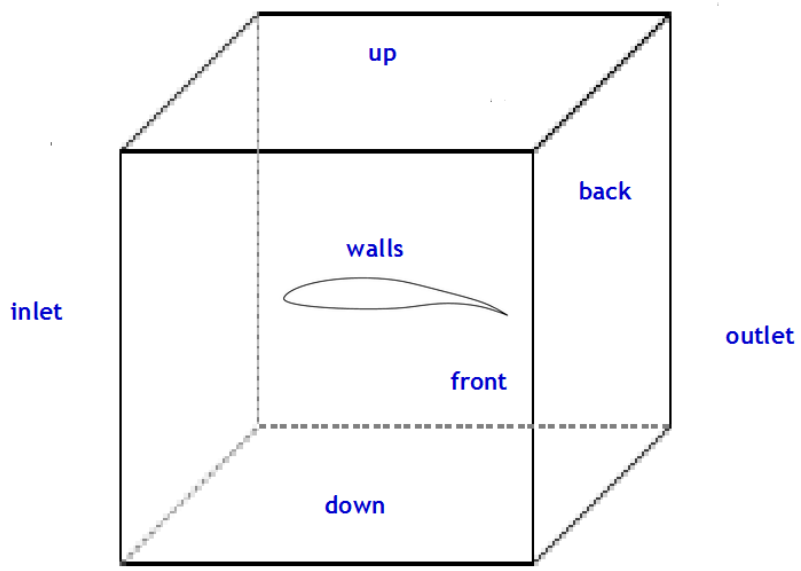


Figure 7.7: *Boundary patches.*

Table 7.1: Initial Conditions for Winfoor case

Boundary	U	P	nut	nuTilda
inlet	90	0	1e-3	1e-3
outlet	90	0	1e-3	1e-3
up	90	0	1e-3	1e-3
down	90	0	1e-3	1e-3
walls	0	zeroGradient	Wall Function	Wall Function
frontAndBack	empty	empty	empty	empty

which is equal to zero. The inlet,outlet,up and down have a freestream condition. For the velocity this freestream value is 90 m/s and for the viscosity 0.14 m^2/s. The freestream condition is the condition far away from the airfoil, the flow is uniform before it hits the airfoil. When the flow reaches the airfoil, one should be careful. No-slip condition is used, which states that for a viscous flow, the velocity is zero at the airfoil. For the pressure, zero gradient is applied, which states that the pressure is constant and doesn't change. The Spalart-Allmaras is a low-Reynolds number model that requires the boundary layer to be properly resolved. Therefore wall functions are applied on the wing. There are many wall functions in OpenFOAM, the one used here is the nutLowReWallFunction. Since it's a two dimensional simulation, frontAndBack have the empty condition. The front and back patches are given together as one patch, frontAndBack.

kkl-omega

This case is presented for the velocity 15 m/s.

Table 7.2: Initial Conditions for Winfoor case

Boundary	U	P	omega	kl	kt	nut
inlet	15	zeroGradient	9.2e-2	0	8.56e-5	calculated
outlet	15	0	9.2e-2	0	8.56e-5	calculated
up	slip	slip	slip	slip	slip	calculated
down	slip	slip	slip	slip	slip	calculated
wing	0	zeroGradient	1e-11	0	1e-11	Wall Function
frontAndBack	empty	empty	empty	empty	empty	empty

In this model, the initial and type of boundary condition is different. The inlet velocity is a fixedValue type meanwhile the outlet has a inletOutlet type. The only exception is the pressure, where the inlet is a zeroGradient type and the outlet a fixedValue type. The inletOutlet is basically a fixedValue boundary condition when a flux into the domain is present and a zeroGradient when the flux is pointing outwards. A slip condition is applied on the up and down patches. The turbulent viscosity has a calculated boundary condition and the wall function used is the nutLowReWallFunction.

Chapter 8

Uniblade Case 2D

In this chapter the results for the uniblade case simulations are presented. First the effect of initial conditions and Reynold's number are presented and then the sensitivity of the mesh.

8.1 Effects of the initial conditions and Reynold's number

Four cases are presented for the Spalart-Allmaras and kkl-ω models. Case 1 and 2 denote cases with different initial values for kkl-ω. Case 3 and 4 denote cases with different initial values for Spalart-Allmaras. All cases have been tested for three increasing velocities.

The convergence is only presented for the pressure. The pictures are only presented for the velocity and pressure. The rest of the convergence plots and pictures for Spalart-Allmaras and kkl-ω can be found in Appendix A.

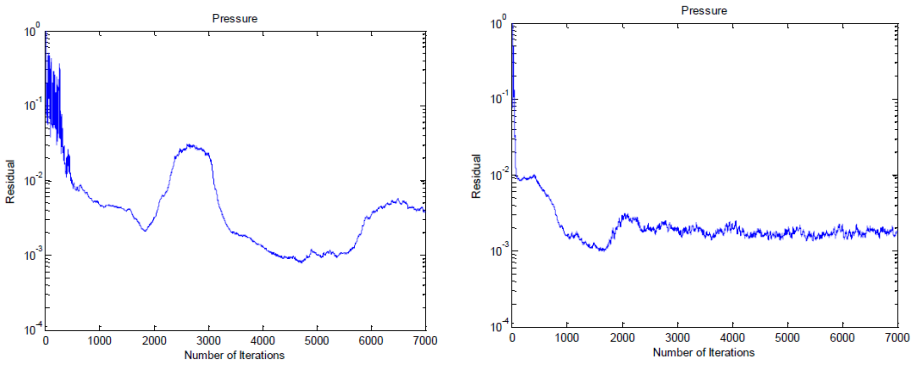
8.1.1 kkl-omega

Table 8.1: Case 1: kkl-ω

$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
15	1 500 000	1.03096	0.00773326	133.3
47	4 700 000	1.06138	0.014661	72.394
90	9 000 000	1.5929	-0.0700465	-22.74

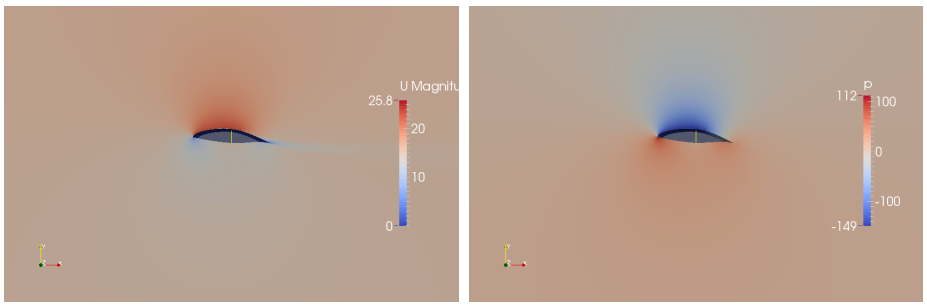
Table 8.2: Case 2: kkl-ω

$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
15	1 500 000	1.30115	0.0103449	125.776
47	4 700 000	1.30662	-0.0374435	-34.89
90	9 000 000	1.35556	0.000758716	1786



(a) *Convergence for case 1* (b) *Convergence for case 2*

Figure 8.1: *Residuals for pressure as function of number of iterations for kkl-ω, 15 m/s. As seen in (a), case 1 didn't converge.*



(a) *The magnitude of the velocity.* (b) *The pressure distribution.*

Figure 8.2: *Pictures for case 2, kkl-ω.*

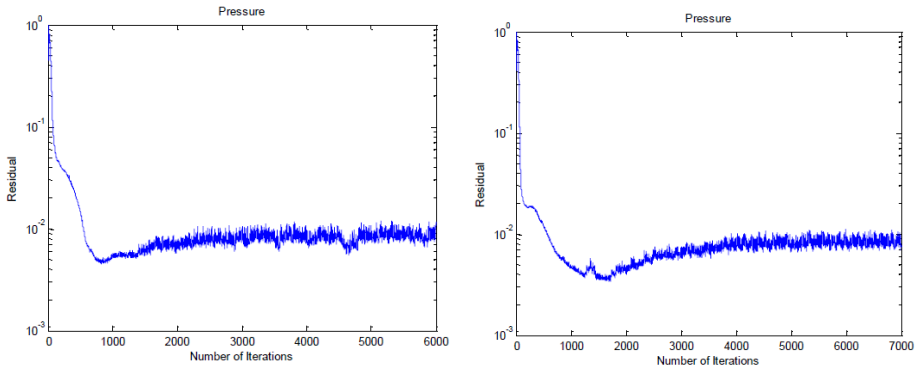
8.1.2 Spalart-Allmaras

Table 8.3: Case 3: Spalart-Allmaras

$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
15	1 500 000	1.09429	0.0234508	46.663
47	4 700 000	1.12043	0.0208857	53.645
90	9 000 000	1.15885	0.0184654	62.757

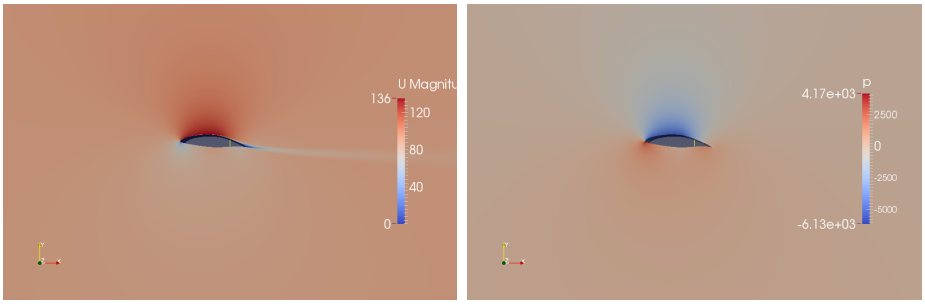
Table 8.4: Case 4: Spalart-Allmaras

$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
15	1 500 000	1.07733	0.0507139	21.24
47	4 700 000	1.13447	0.0361544	31.378
90	9 000 000	1.17537	0.0286844	40.975



(a) *Convergence for case 3* (b) *Convergence for case 4*

Figure 8.3: *Residuals for pressure as function of number of iterations for Spalart-Allmaras, 90 m/s. Both cases converged.*



(a) *The magnitude of the velocity.* (b) *The pressure distribution.*

Figure 8.4: *Pictures for case 3, Spalart-Allmaras.*

8.1.3 Mesh Sensitivity

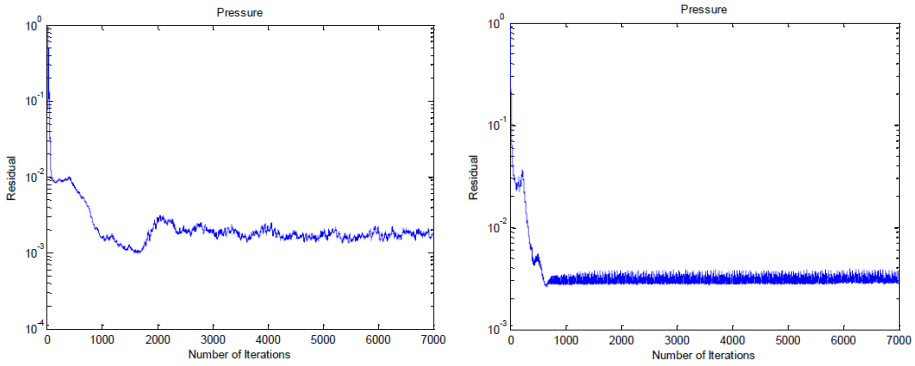
The snappy mesh and gmsh are compared for the two turbulence models. Case 2 (kkl-ω) is carried out with the snappy mesh for the velocity 15 m/s. Case 3 (Spalart-Allmaras) is carried out with the snappy mesh for the velocity 90 m/s. Then the sensitivity analysis for the snappy mesh carried out with Spalart-Allmaras and kkl-ω is presented.

Three mesh levels have been tested (Mesh1, Mesh2 and Mesh3) with both turbulence models. The results are not presented in this section, but are given in Appendix A together with the rest of the convergence plots.

kkl-omega

Table 8.5: Comparison of snappy and gmsh for kkl-ω.

Mesh	*Velocity*$[m/s]$	*Re*	C_l	C_d	*GlideRatio*
gmsh	15	1 500 000	1.30115	0.0103449	125.776
snappy	15	1 500 000	1.28552	0.013756	93.451



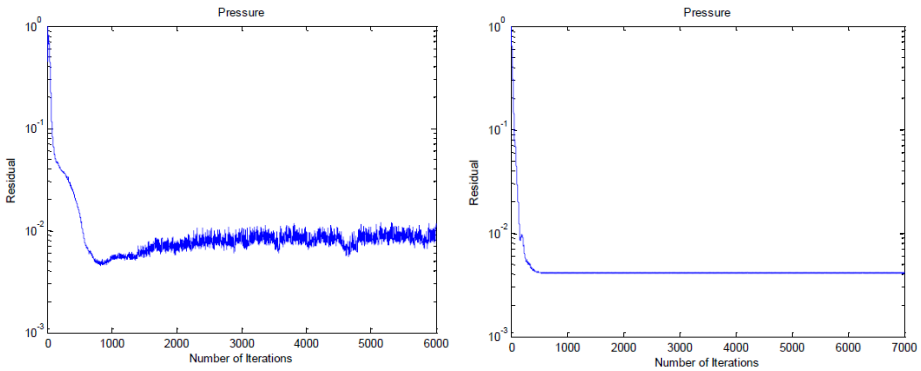
(a) *Convergence for case 2 with gmsh* (b) *Convergence for case 1 with snappy mesh*

Figure 8.5: *Residuals for pressure as function of number of iterations for kkl-ω, 15 m/s.*

Spalart-Allmaras

Table 8.6: Comparison of snappy and gmsh for Spalart-Allmaras.

$Mesh$	$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
gmsh	90	9 000 000	1.15885	0.0184654	62.757
snappy	90	9 000 000	1.33296	0.0193836	68.767



(a) *Convergence for case 3 with gmsh* (b) *Convergence for case 3 with snappy mesh*

Figure 8.6: *Residuals for pressure as function of number of iterations for Spalart-Allmaras, 90 m/s.*

Mesh sensitivity to turbulence models

Table 8.7: Mesh sensitivity for Mesh 1

$Mesh$	$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
kkl-omega	80	6 000 000	1.33782	0.0103169	129.67
Spalart-Allmaras	80	6 000 000	1.32623	0.020591	64.40

Table 8.8: Mesh sensitivity for Mesh 2

$Mesh$	$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
kkl-omega	80	6 000 000	1.34187	0.0117161	114.5
Spalart-Allmaras	80	6 000 000	1.24664	0.0259582	48.02

Chapter 9

Triblade Case 2D

In this chapter, the results for the triblade case simulations are presented. The effects of Reynold's number, sensitivity to wall functions and mesh are presented.

9.1 Effects of Reynold's number

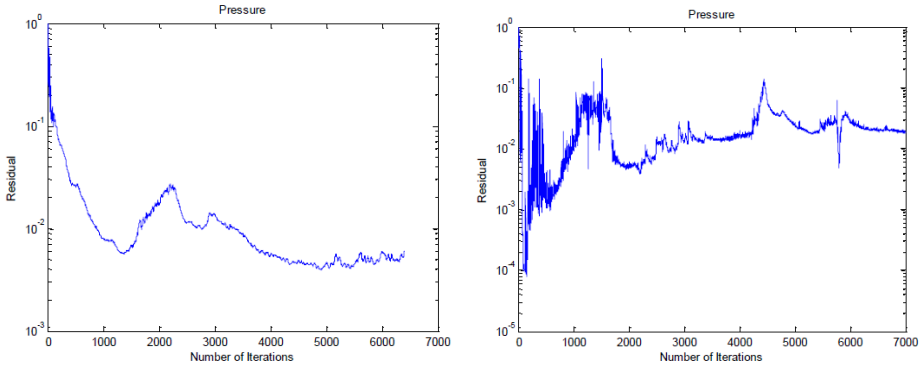
Two cases are presented for the two turbulence models where three different velocities are tested. For kkl-ω, the velocity 90 m/s was tested so both turbulence models simulations were carried out for high and low velocities.

The convergence is only presented for the pressure. The pictures are only presented for the velocity and pressure. The rest of the convergence plots and pictures for Spalart-Allmaras and kkl-ω can be found in Appendix B.

9.1.1 kkl-omega

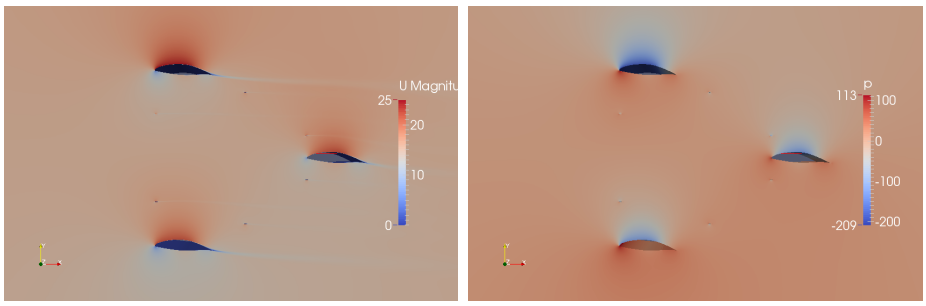
Table 9.1: Case for kkl-ω

$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
5	500 000	1.143	0.028	40.82
10	1 000 000	1.1394	0.023	48.813
15	1 500 000	1.213	0.0156	77.362
90	9 000 000	2.267	-3.832	0.591



(a) *Convergence for the velocity 15 m/s* (b) *Convergence for the velocity 90 m/S*

Figure 9.1: *Residuals for pressure as function of number of iterations for kkl-ω. As seen in the two figures, convergence wasn't reached.*



(a) *The magnitude of the velocity.* (b) *Pressure Distribution.*

Figure 9.2: *Pictures for kkl-ω, 15 m/s.*

9.1.2 Spalart-Allmaras

Table 9.2: Case for Spalart-Allmaras

$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
5	500 000	0.9128	0.088	10.301
10	1 000 000	0.9927	0.068694	14.451
15	1 500 000	1.012	0.0617	16.395

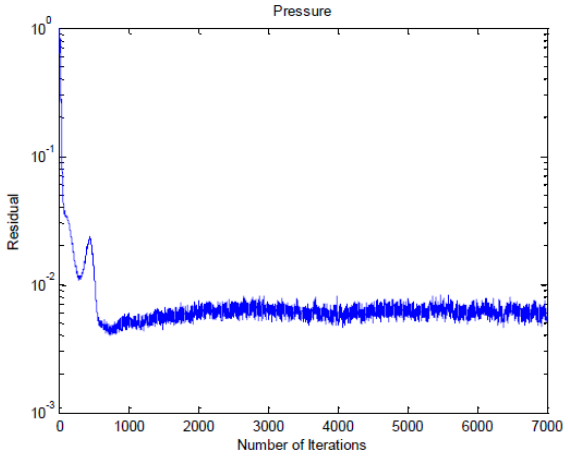
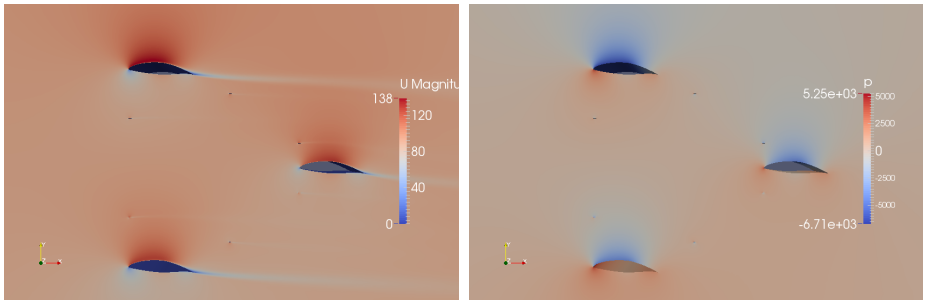


Figure 9.3: *Residuals for pressure as function of number of iterations for Spalart-Allmaras, 90 m/s. It showed better convergence than kkl-ω.*



(a) *The magnitude of the velocity.* (b) *Pressure Distribution.*

Figure 9.4: *Pictures for Spalart-Allmaras, 90 m/s.*

9.2 Sensitivity to wall functions

Table 9.3: Case for kkl-ω, WF denotes Wall function

Spalart	$Velocity[m/s]$	*Re*	C_l	C_d	*GlideRatio*
With WF	15	1 500 000	1.213	0.0156	77.362
No WF	15	1 500 000	1.21	0.016	75.54

Table 9.4: Case for Spalart-Allmaras, WF denotes wall function

Spalart	$Velocity[m/s]$	*Re*	C_l	C_d	*GlideRatio*
With WF	90	9 000 000	1.135	0.03264	34.77
No WF	90	9 000 000	1.1345	0.03269	34.66

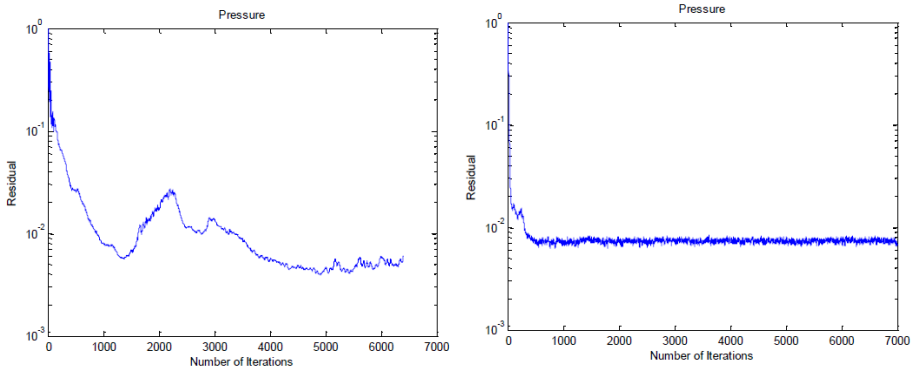
9.3 Mesh Sensitivity

The same procedure that was carried out for the uniblade case was also carried out for the triblade case. The kkl-ω case in Table 9.1 for the velocity 15 m/s and the Spalart-Allmaras case in Table 9.4 for the velocity 90 m/s are compared with the snappy mesh. The simulations included the wall functions. The results for the different mesh levels and convergence plots can be find in Appendix B.

kkl-omega

Table 9.5: Comparison of snappy and gmsh for kkl-ω.

$Mesh$	$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
gmsh	15	1 500 000	1.213	0.0156	77.362
snappy	15	1 500 000	1.22518	0.0170231	71.97



(a) *Convergence for kkl-ω with gmsh.*

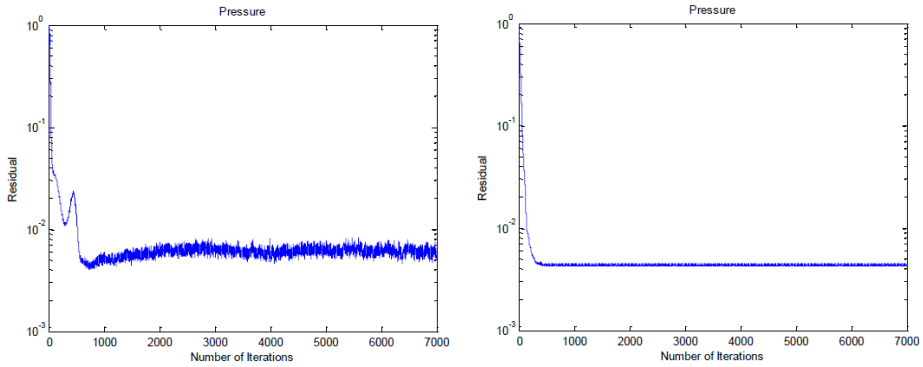
(b) *Convergence for kkl-ω with snappy mesh.*

Figure 9.5: *Residuals for pressure as function of number of iterations for kkl-ω, 15 m/s.*

Spalart-Allmaras

Table 9.6: Comparison of snappy and gmsh for Spalart-Allmaras.

$Mesh$	$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
gmsh	90	9 000 000	1.13502	0.03264	34.77
snappy	90	9 000 000	1.2815	0.02124	60.31



(a) *Convergence for kkl-ω with gmsh.* (b) *Convergence for kkl-ω with snappy mesh.*

Figure 9.6: *Residuals for pressure as function of number of iterations for Spalart-Allmaras, 90 m/s.*

Mesh sensitivity to turbulence models

Table 9.7: Mesh sensitivity for Mesh 1

$Mesh$	$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
kkl-omega	80	6 000 000	1.2741	0.0137	92.9
Spalart-Allmaras	80	6 000 000	1.276	0.0224	57

Table 9.8: Mesh sensitivity for Mesh 3

$Mesh$	$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
kkl-omega	80	6 000 000	1.291	0.01556	82.77
Spalart-Allmaras	80	6 000 000	1.193	0.0282	43.37

Chapter 10

Triblade Case 3D

In this chapter, the results for the three dimensional cases are presented. First the *Base Case* is presented which is the mesh with the blades and beams included. In the *Base Case*, the angle of attack is set to 5 degrees. The *Only Blade Case* denotes the mesh without the beams included. The case *6AOA* and *4AOA* denote the cases for the two different angles of attack which are 6 and 4 degrees. The last case represent the case where one blade is rotated 10 degrees (*C-Rotated Case*).

Only the magnitude of the forces are given in the tables, the components of the forces and other data are given in Appendix C.

10.1 Effect of Geometry

The effect of the geometry is presented for the *Base* and *only Blade Case*. The *Only Blade Case* is then compared to the *Base Case*.

10.1.1 Base Case

Table 10.1: Magnitude of the pressure (P), viscous (V), lift (L), drag (D), tangential (T) force.

Blades	*P*	*V*	*L*	*D*	*T*	*GlideRatio*
Blade A	28 736.3	241.3	28 695.8	1401.9	8030.7	20.5
Blade B	38 304.1	288.33	38 210.1	2333.3	11 147.2	16.4
Blade C	20 466.1	259.6	20 472.5	673.5	4101.5	30.4

10.1.2 Only Blade Case

Table 10.2: Magnitude of the pressure (P), viscous (V), lift (L), drag (D), tangential (T) force.

Blades	*P*	*V*	*L*	*D*	*T*	*GlideRatio*
Blade A	32 464.8	245.8	32 383.1	2156.3	9621.3	15
Blade B	38 842.3	290.1	38 746.6	2370.5	11 308.1	16.3
Blade C	21 389.9	264.5	21 398.4	581.2	4406.4	36.8

Table 10.3: Only Blades Case: Force difference relative to the base case in %

Blades	*P*	*V*	*L*	*D*	*GlideRatio*
Blade A	13	1.9	12.8	53.8	-26.6
Blade B	1.4	0.6	1.4	1.6	-0.2
Blade C	4.5	1.9	4.5	-13.7	21.1

10.2 Angle of attack sensitivity

The results for the *6AOA* and *4AOA* cases are presented and compared to the *Base Case*.

10.2.1 6AOA Case

Table 10.4: Magnitude of the pressure (P), viscous (V), lift (L), drag (D), tangential (T) force.

Blades	*P*	*V*	*L*	*D*	*T*	*GlideRatio*
Blade A	30 585.5	234.4	30 510.9	2008	9042	15.19
Blade B	41 343.7	284.6	41 194.8	3161.7	12 646.6	13.02
Blade C	22 309.2	261.2	22 320.5	461.2	4737.4	48.4

Table 10.5: Only Blades Case: Force difference relative to the base case in %

Blades	*P*	*V*	*L*	*D*	*GlideRatio*
Blade A	6.4	-2.9	6.3	43.2	-25.8
Blade B	7.9	-1.3	7.8	35.5	-20.4
Blade C	9	0.6	9	-31.5	59.2

10.2.2 4AOA Case

Table 10.6: Magnitude of the pressure (P), viscous (V), lift (L), drag (D), tangential (T) force.

Blades	*P*	*V*	*L*	*D*	*T*	*GlideRatio*
Blade A	27 353	246.8	27 331.3	964.2	7288	28.34
Blade B	35 215	289.8	35 160.7	1593.7	9719.4	22.06
Blade C	18 944.7	257.8	18 945.2	826.5	3597.8	22.1

Table 10.7: Only Blades Case: Force difference relative to the base case in %

Blades	*P*	*V*	*L*	*D*	*GlideRatio*
Blade A	-4.8	2.3	-4.8	-31.2	38.5
Blade B	-8.1	0.5	-8	-31.7	34.7
Blade C	-7.4	-0.7	-7.5	22.7	-24.6

10.3 C-Rotated Blade

The results for *C-Rotated Blade Case* are presented and compared to the *Base Case*.

Table 10.8: Magnitude of the pressure (P), viscous (V), lift (L), drag (D), tangential (T) force.

Blades	*P*	*V*	*L*	*D*	*T*	*GlideRatio*
Blade A	33 605.1	234.4	33 474.3	2821.3	10 521.5	11.86
Blade B	43 050.8	294.7	42 911.3	3094.7	12 980.1	13.86
Blade C	44 937.1	228.1	44 887.4	2132.6	8354.8	21.04

Table 10.9: Only Blades Case: Force difference relative to the base case in %

Blades	*P*	*V*	*L*	*D*	*GlideRatio*
Blade A	16.9	-2.9	16.7	101.2	-42
Blade B	12.4	2.2	12.3	32.6	-15.3
Blade C	119.6	-12.1	119.3	216.6	-30.8

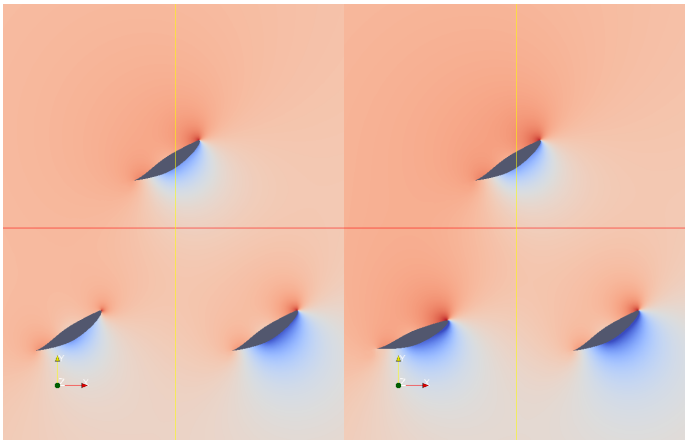


Figure 10.1: *Pressure distribution for the non rotated case (left) and rotated case (right). One can clearly see the influence of the plates and beams and also the uneven distribution of the pressure on the three blades.*

10.4 Summary of the 3D Simulations

Charts are presented to see the difference between the different cases.

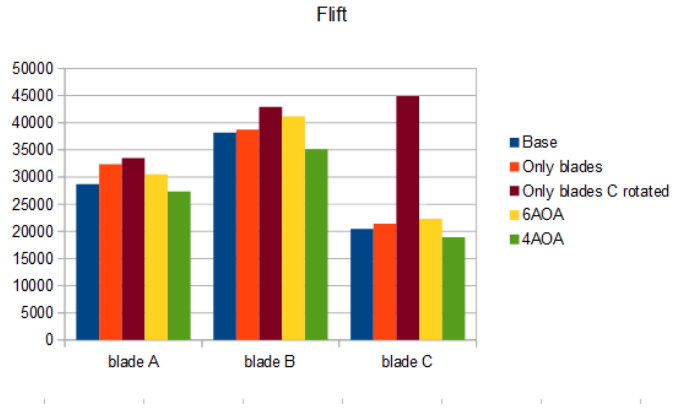


Figure 10.2: *Lift force on each blade for the different cases.*

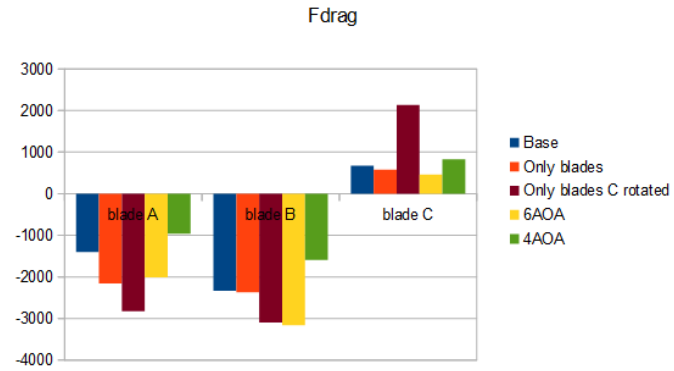


Figure 10.3: *Drag force on each blade for the different cases.*

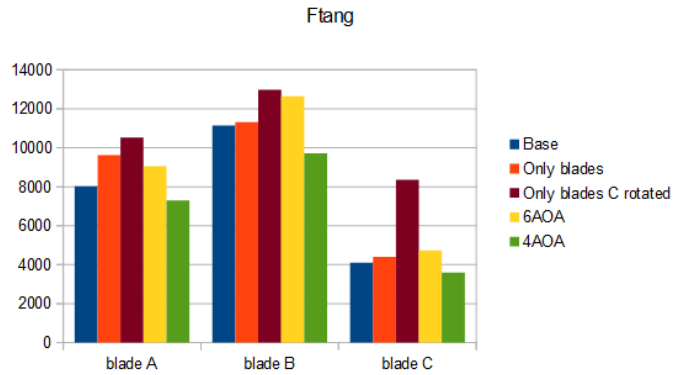


Figure 10.4: *Tangential force on each blade for the different cases.*

Chapter 11

Discussion

In this chapter a discussion of the results are presented. The the results are first interpreted for the uniblade case followed by the triblade case.

11.1 Uniblade Case 2D

11.1.1 Effect of initial values and Reynold's number

Both turbulence models showed sensitivity to the initial values, especially kkl-ω. This can be seen in Table 8.1 and 8.2, where negative values of C_d were obtained. These cases with the negative C_d didn't converge properly and that can be a reason. In case 2, the drag is so small that it resulted in a very high and unrealistic glide ratio. Spalart-Allmaras is less sensitive but one can see that changing the initial values for the turbulent viscosity gave higher drag, see Table 8.3 and 8.4. The turbulent viscosity was increased in case 4. When it comes to the effect of low and high Reynolds number, it showed that kkl-ω worked better for low velocities (15 m/s) and Spalart for higher velocities (90 m/s). All cases converged except case 1, which can clearly bee seen in Figure 8.1 (a). It could be a numerical error or that something wrong with some cells in the mesh. Better convergence was obtained with Spalart-Allmaras, see Figure 8.3.

11.1.2 Mesh Sensitivity

The gmsh showed better results than the snappy mesh for kkl-ω, see Table 8.5. There is not much difference in the lift and drag coefficients, but gmsh gave lower drag which resulted in higher glide ratio. For Spalart-Allmaras it was the opposite. The lift and drag were a little bit higher with the snappy mesh, see Table 8.6. The difference between the gmsh and snappy for kkl-ω is presented in Table 11.1 as follows:

Table 11.1: Difference between gmsh and snappy in %

$Difference$	C_l	C_d	$GlideRatio$
%	1.2	24.7	25.7

It could be interesting to compare the results with results from tunnel tests. Unfortunately, only data from other simulations were available. The following data presented in Table 11.2 and 11.3 are taken from NREL home page and the simulations are carried out with a so called Eppler Code [3].

Table 11.2: Data for NREL s831 airfoil for rough surface

Re	2 000 000	3 000 000	4 000 000	6 000 000	9 000 000
C_l	1.167	1.178	1.186	1.195	1.205
C_d	0.017128	0.016323	0.015731	0.014889	0.014063
GlideRatio	68.13	72.168	75.39	80.26	85.68

Table 11.3: Data for NREL s831 airfoil for smooth surface

Re	2 000 000	3 000 000	4 000 000	6 000 000	9 000 000
C_l	1.232	1.241	1.244	1.244	1.220
C_d	0.0077244	0.006686	0.0065201	0.008956	0.010037
GlideRatio	170	185.6	190.79	136.6	121.5

Comparing the results for kkl-ω for 80 m/s (Table 8.7) with the results for smooth surface (Table 11.3) for Re 6 000 000 from NREL, the results only differs 5 % according to:

$$\frac{136 - 129.67}{136} = 0.05 \tag{11.1}$$

Comparing the results obtained with the results for the rough surface in Table 11.2, the results don't differ much and are realistic. However, negative drag coefficients weren't obtained with the snappy mesh.

11.2 Triblade Case 2D

11.2.1 Effect of Reynold's number

The values in Table 9.1 and 9.2 obtained are realistic and there are no sudden jumps between the lift and drag coefficients when the velocities were increased. Spalart-Allmaras gave higher drag and lower lift coefficient than kkl-ω. The only incorrect value is for kkl-ω, where the simulation didn't give appropriate values for high velocities (90 m/s), see Table 9.1. Again a negative value of the drag is obtained and the lift is high. By looking at Fig.9.1, one can see that the case didn't converge yet. In Fig.9.1 (a), the residuals started to stabilize. Maybe it was something wrong with the mesh for the triblade or the simulation required more iterations. However the convergence was better with Spalart-Allmaras, see Fig.9.3.

One reason that could have caused higher drag in the simulations was that the boundary layer was resolved enough so there is no need for wall functions. By realising that y^+ are less than 30, the wall functions were excluded. As seen in Table 9.3 and 9.4, not much difference is obtained when including and excluding the wall functions.

That kkl-ω worked better for lower velocities and Spalart-Allmaras for higher velocities is only an assumption made so far. The same conclusion can be made as in the uniblade case. The only problem is the drag coefficient that needs to be predicted correctly (neither under-or over estimated). The gmsh can't be manipulated so to solve this issue, what was left was the mesh quality which is investigated in the next section.

11.2.2 Mesh Sensitivity

Again the gmsh showed better results than the snappy mesh for kkl-ω (Table 9.5) and for Spalart-Allmaras it was the opposite. The difference between the gmsh and snappy is presented in Table 11.5 as follows:

Table 11.4: Difference between gmsh and snappy in %

$Mesh$	C_l	C_d	$GlideRatio$
%	0.99	8.35	6.9

The highest difference is in the drag coefficient but when it comes to the convergence, the snappy mesh showed better convergence than the gmsh, see Fig.9.5.

11.2.3 Comparison between uni-and triblade case

The lift and drag for the uni- and triblade case are compared for the snappy mesh as follows:

Table 11.5: Comparison of uni-and triblade, kkl-ω

$Case$	$Velocity[m/s]$	Re	C_l	C_d	$GlideRatio$
Uniblade	80	6 000 000	1.33782	0.0103169	129.67
Triblade	80	6 000 000	1.2741	0.0137099	92.9

Table 11.6: Comparison of uni-and triblade in %

$Difference$	C_l	C_d	$GlideRatio$
%	4.7	-32	28

By looking at Table 11.5 and 11.6 one can clearly see that for the triblade case, the lift coefficient is lower and drag coefficient higher than for the uniblade case. The difference is only 4.7 % for the lift coefficient but the highest difference is in the drag coefficient. The geometry in the triblade case is different than in the uniblade case, so the plates and beams disturbs the flow and could cause higher drag.

11.3 Triblade Case 3D

Comparing the different cases, one can clearly see that the forces are different on each blade for the different cases. Including and excluding a part of the geometry does have an effect on the results. This can be seen when comparing the base case with the only blades case. Excluding the beams increases the forces, see Table 10.3. The beams disturb the flow and give lower efficiency.

When it comes to the sensitivity of the angle of attack, case 6AOA gives higher lift and drag than the base case, see Table 10.5. For the base 4AOA, the opposite happens (see Table 10.7). An increase of the angle of attack gives higher forces (lift and drag) and a decrease lower forces.

By trying to adjust the pitch for blade C to give the correct angle of attack, higher tangential force is obtained. This can be seen in Figure 10.3. In Figure 10.1-10.3, the lift, drag and tangential force are presented for each blade for all cases. One can clearly see that the forces are highest for the C-rotated case and lowest for the base case. This means that the geometry can be optimized better if the three blades have different pitch angles.

Chapter 12

Conclusion and Further Work

After investigating the results one comes to the conclusion that the kkl-ω turbulence model is best suitable for predicting the flow. It is more realistic since it models a transitional flow. Spalart-Allmaras is less good and gave higher drag. The most difficult and challenging part in the simulations was to evaluate the drag coefficient. A lower drag is desirable but one should not under estimate the drag either, so it is important to achieve a correct predicted value of the drag. One way to solve over estimating the drag with Spalart-Allmaras is to divide the computational domain in a laminar and turbulent zone. Thus one could have a transitional model even though Spalart-Allmaras assumes full turbulence everywhere. This is done in a tutorial by Hamidreza Abedi, see [1].

There is no doubt that the turbulence models are sensitive to parameter changes. Improper initial conditions can lead to devastating results. Rather than starting with the turbulence model variables themselves, it's often easier to think of turbulence intensity I and length scale l. The kinetic energy k, specific dissipation rate ω and turbulent viscosity ν are not easy to guess. One way is to calculate them by:

$$k = \frac{3}{2}(\bar{U}I)^2 \tag{12.1}$$

$$\omega = \frac{3}{2}(\bar{U}I)^2 \tag{12.2}$$

$$\nu = \sqrt{\frac{3}{2}}\bar{U}Il \tag{12.3}$$

Using a mesh of adequate geometrical quality is an important part of controlling the results. A good mesh helps the CFD solver to converge to the correct answer while minimizing the computer resources. Every solver yields an accurate answer with a good mesh, but it takes the most robust of solvers to get an answer to a bad mesh. One could have set up everything right, but with a bad mesh nothing will be correct. It's much easier to define a bad mesh than judging a good mesh. Significant measures of mesh quality may be categorized as measures of mesh orthogonality, expansion and aspect ratio. Badness usually means whether there are any negative volume cells. There were probably some bad cells somewhere in the Gmsh which resulted in negative drag coefficients for kkl-ω. As seen in the results, kkl-ω worked well for high velocities in the snappy mesh. Constantly improving the mesh will give better results.

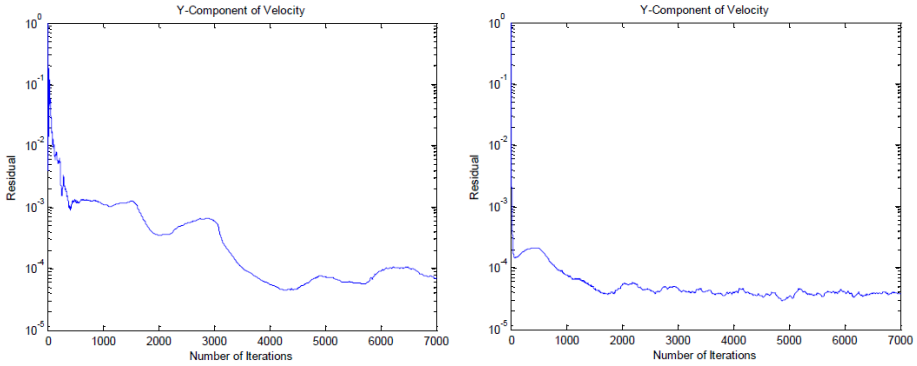
The two dimensional simulations are not enough to analyze the flow. The structure is not two dimensional and all the beams and plates that are included in the triblade do affect the flow behaviour. Therefore three dimensional simulations were necessary. It is important to know how big the forces that are applied on the blades are. The incoming velocity profile is never perfect, therefore it was convenient to investigate the sensitivity to the fluctuations in angle of attack. After studying 3D effects one comes to the conclusion that by changing the pitch angle and optimizing the geometry, better efficiency is obtained.

It was planned to make a Large Eddy Simulation (LES) to study the end effects. To be able to do this, one has to have a closed geometry, which wasn't available. One could try to make a closed obj-file, but there wasn't time enough to make it.

Due to the limited time of only 20 weeks, this project is not aiming to evaluate all areas of the wide range of CFD applications, but mainly aerodynamics. There are limited pre-processing and not much mesh generation involved. In the snappy mesh, there are many settings that is hard to understand for the one who is new to OpenFOAM. The evaluation process is only based on the already existing solvers, cases and applications, which means that no programming is done during this project.

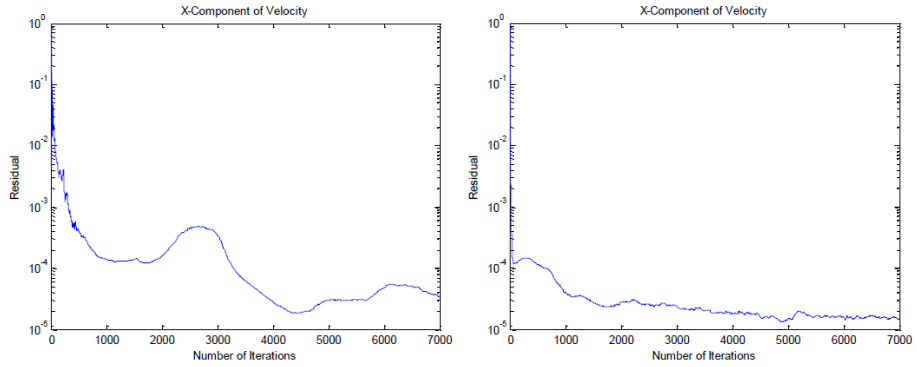
Appendix A : Uniblade Case 2D

kkl-omega



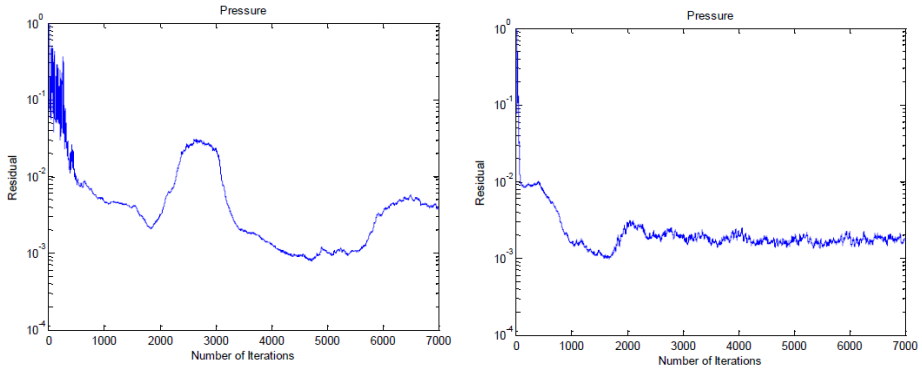
(a) *Convergence for case 1* (b) *Convergence for case 2*

Figure 12.1: *Residuals for y-component of the velocity as function of number of iterations for kkl-ω, 15 m/s.*



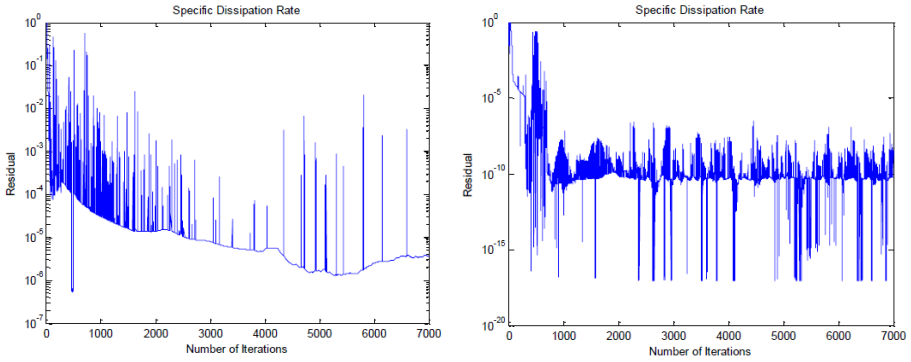
(a) *Convergence for case 1* (b) *Convergence for case 2*

Figure 12.2: *Residuals for x-component of the velocity as function of number of iterations for kkl-ω, 15 m/s.*



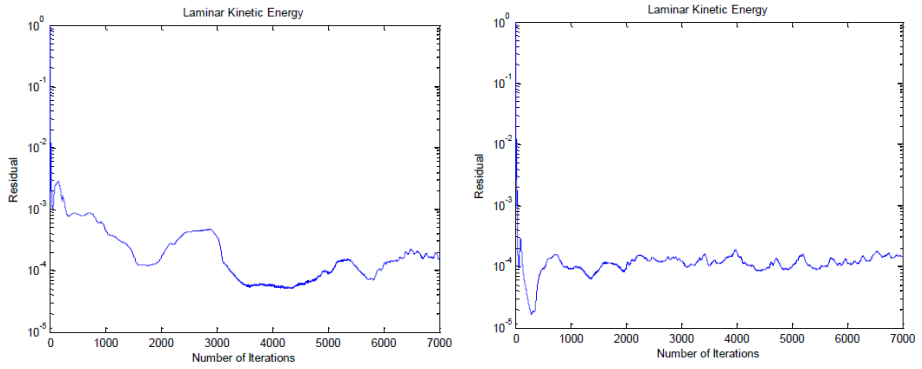
(a) *Convergence for case 1* (b) *Convergence for case 2*

Figure 12.3: *Residuals for pressure of function of number of iterations for kkl-ω, 15 m/s.*



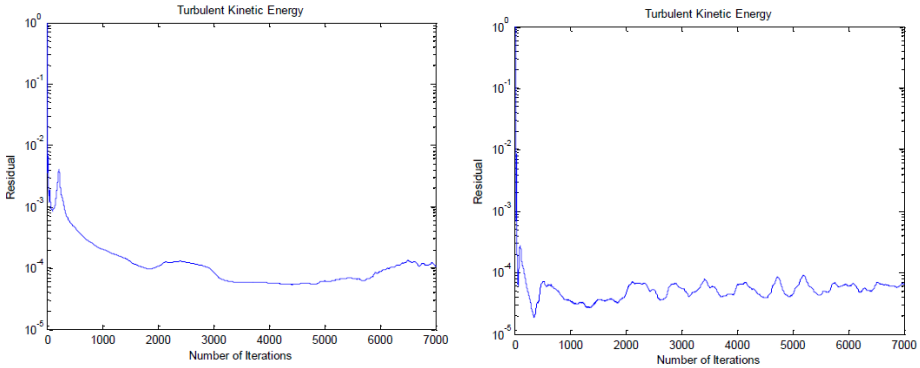
(a) *Convergence for case 1* (b) *Convergence for case 2*

Figure 12.4: *Residuals for the specific dissipation rate as function of number of iterations for kkl-ω, 15 m/s.*



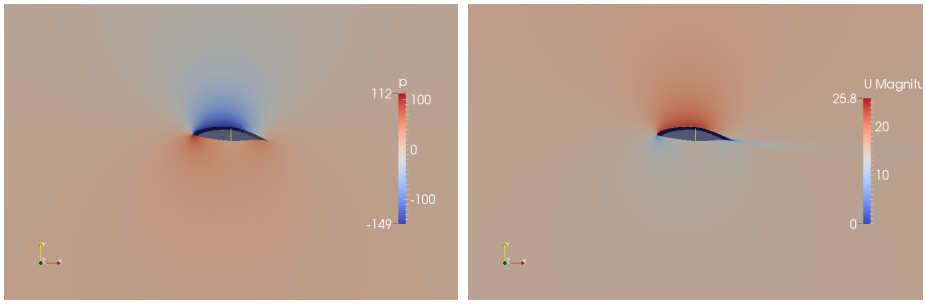
(a) *Convergence for case 1* (b) *Convergence for case 2*

Figure 12.5: *Residuals for the laminar kinetic energy as function of number of iterations for kkl-ω, 15 m/s.*



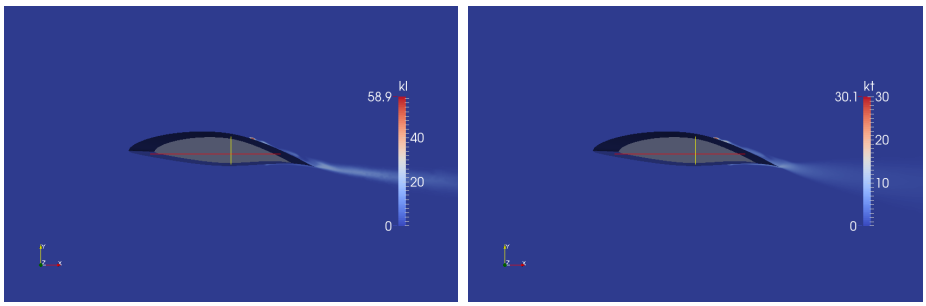
(a) *Convergence for case 1* (b) *Convergence for case 2*

Figure 12.6: *Residuals for the turbulent kinetic energy as function of number of iterations for kkl-ω, 15 m/s.*



(a) *Pressure* (b) *Magnitude of the velocity*

Figure 12.7: *Pictures case 2, kkl-ω, 15 m/s.*



(a) *Laminar Kinetic Energy* (b) *Turbulent Kinetic Energy*

Figure 12.8: *Pictures case 2, kkl-ω 15 m/s.*

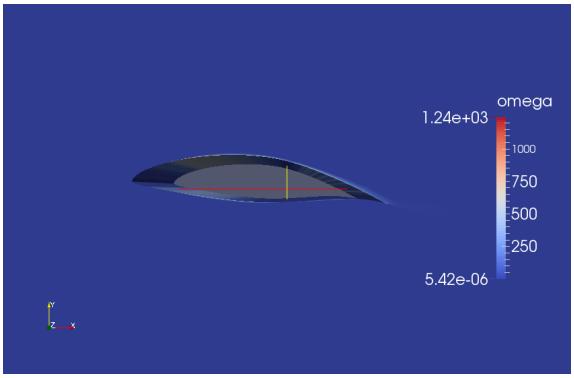
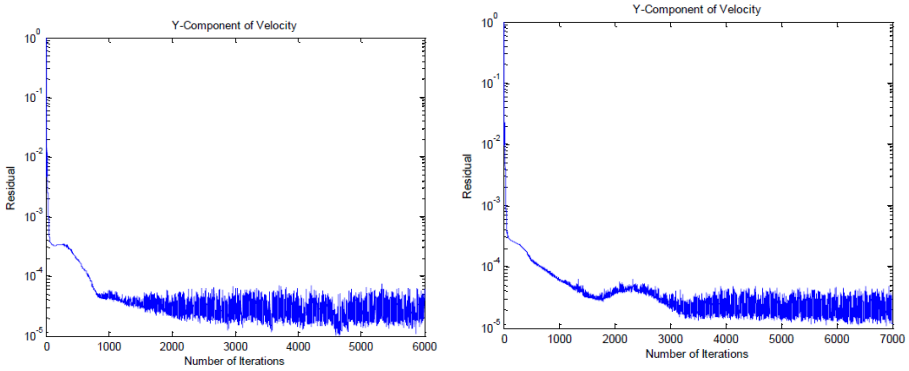


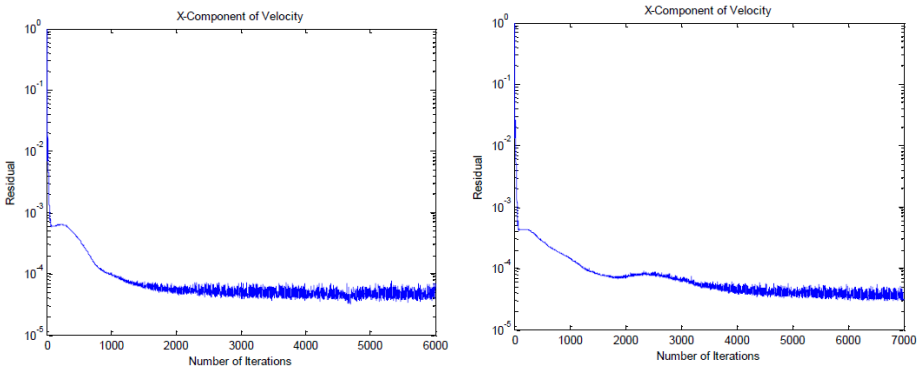
Figure 12.9: *Specific Dissipation Rate for case 2, kkl-ω 15 m/s.*

Spalart-Allmaras



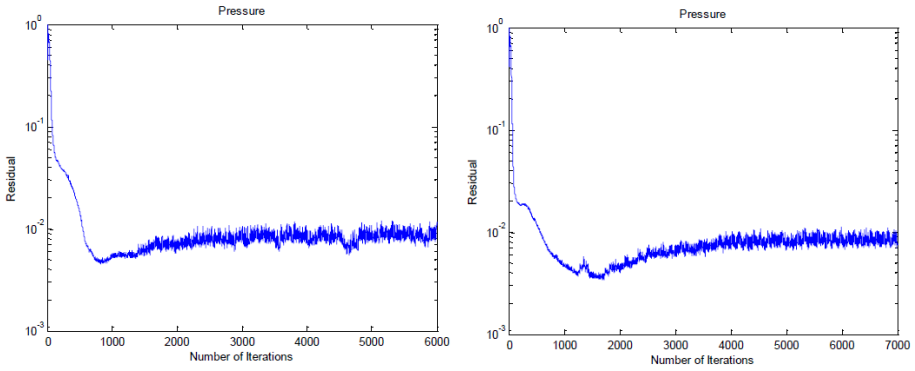
(a) *Convergence for case 3* (b) *Convergence for case 4*

Figure 12.10: *Residuals for the y-component as a function of number of iterations for, kkl-ω, 15 m/s.*



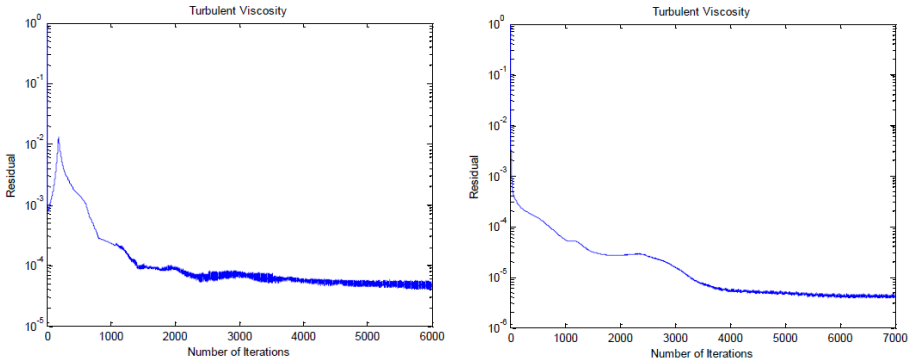
(a) *Convergence for case 3* (b) *Convergence for case 4*

Figure 12.11: *Residuals for the x-component as a function of number of iterations for, kkl-ω, 15 m/s.*



(a) *Convergence for case 3* (b) *Convergence for case 4*

Figure 12.12: *Residuals for the pressure as a function of number of iterations for, kkl-ω, 15 m/s.*



(a) *Convergence for case 3* (b) *Convergence for case 4*

Figure 12.13: *Residuals for the turbulent viscosity as a function of number of iterations for, kkl-ω, 15 m/s.*

SnappyHexMesh Results

Spalart-Allmaras

Table 12.1: Results for Spalart-Allmaras after 7000 iterations, Mesh 1

$Mesh1$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	90	6 750 000	1.33296	0.0193836	68.767
No WF	90	6 750 000	0.851147	0.05247	16.22

Table 12.2: Results for Spalart-Allmaras after 7000 iterations, Mesh 2

$Mesh2$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	90	6 750 000	1.30127	0.02117	61.46
No WF	90	6 750 000	0.933266	0.0455106	20.506

Table 12.3: Results for Spalart-Allmaras after 7000 iterations, Mesh 3

$Mesh3$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	90	6 750 000	1.25824	0.0243845	51.599
No WF	90	6 750 000	1.03313	0.0414094	24.94

kkl-omega

Table 12.4: Results for $kkl - \omega$ after 7000 iterations, Mesh 1

$Mesh1$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	15	1 125 000	1.28359	0.0146488	87.624
No WF	15	1 125 000	1.27859	0.0149236	85.67

Table 12.5: Results for $kkl - \omega$ after 7000 iterations, Mesh 2

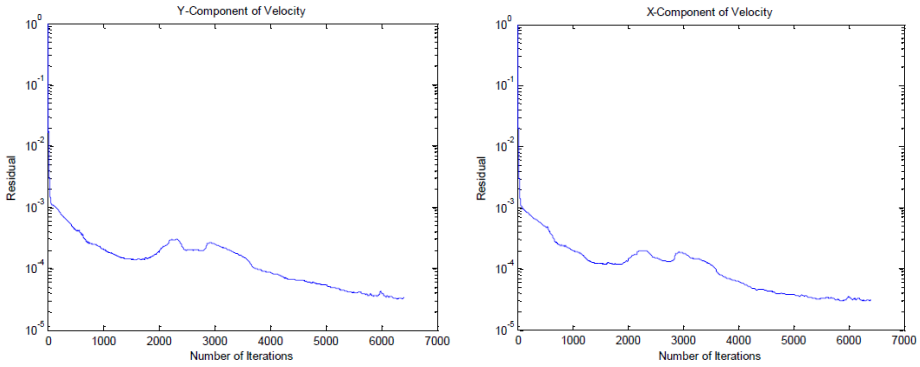
$Mesh2$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	15	1 125 000	1.28597	0.0143898	89.366
No WF	15	1 125 000	1.28351	0.0141512	90.69

Table 12.6: Results for $kkl - \omega$ after 7000 iterations, Mesh 3

$Mesh3$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	15	1 125 000	1.28552	0.013756	93.451
No WF	15	1 125 000	1.28649	0.013898	92.56

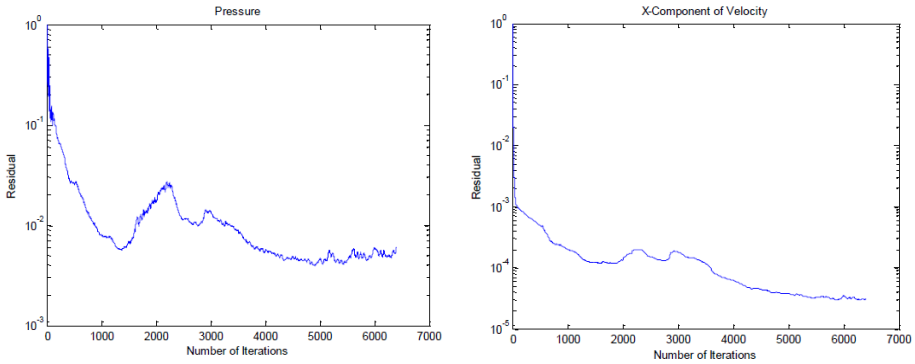
Appendix B : Triblade Case 2D

kkl-omega



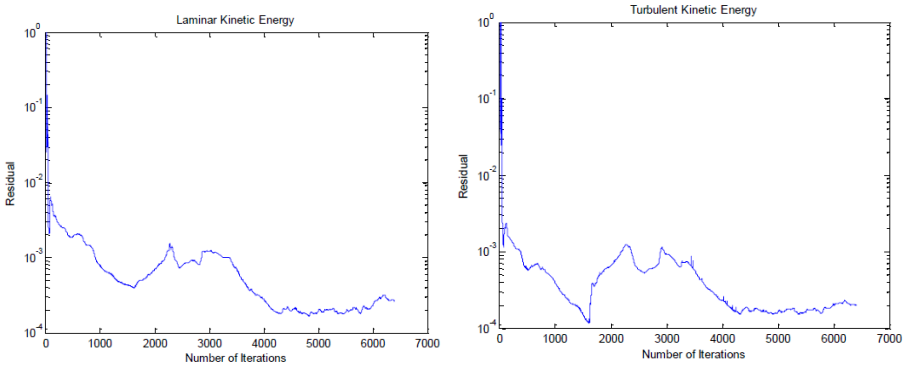
(a) *y-component of the velocity* (b) *x-component of the velocity*

Figure 12.14: *Residuals as function of number of iterations for for kkl-ω, 15 m/s.*



(a) *Pressure* (b) *Specific Dissipation rate*

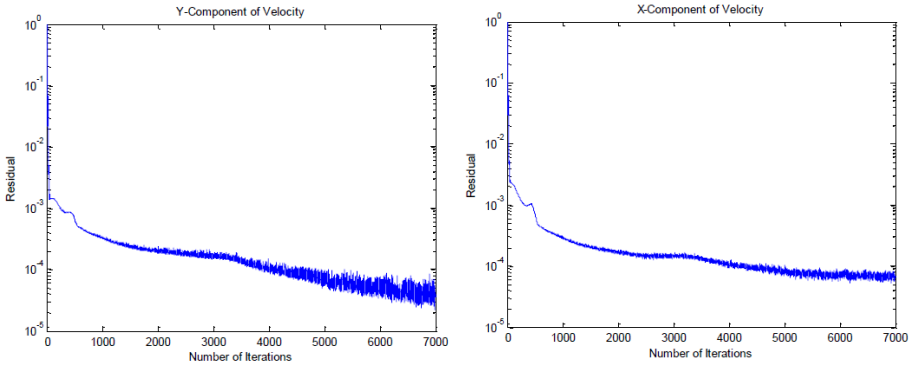
Figure 12.15: *Residuals as function of number of iterations for for kkl-ω, 15 m/s.*



(a) *Laminar Kinetic Energy* (b) *Turbulent Kinetic Energy*

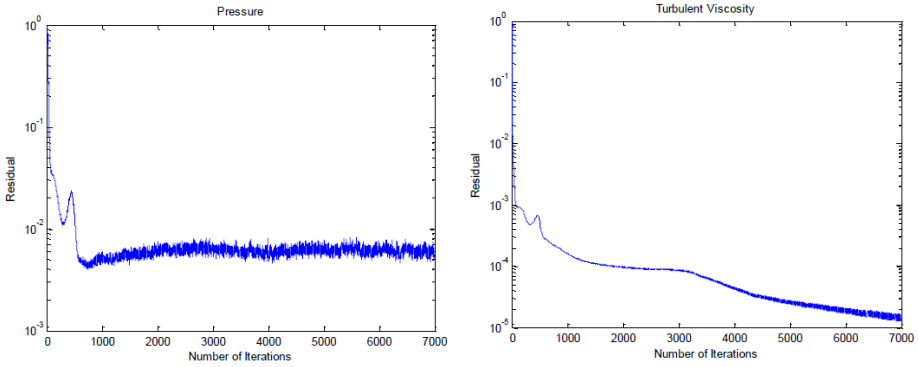
Figure 12.16: *Residuals as function of number of iterations for for kkl-ω, 15 m/s.*

Spalart-Allmaras



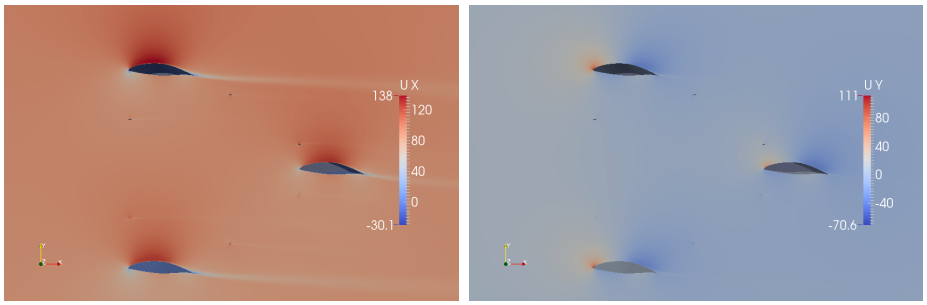
(a) *y-component of the velocity* (b) *x-component of the velocity*

Figure 12.17: *Residuals as function of number of iterations for for Spalart-Allmaras, 90 m/s.*



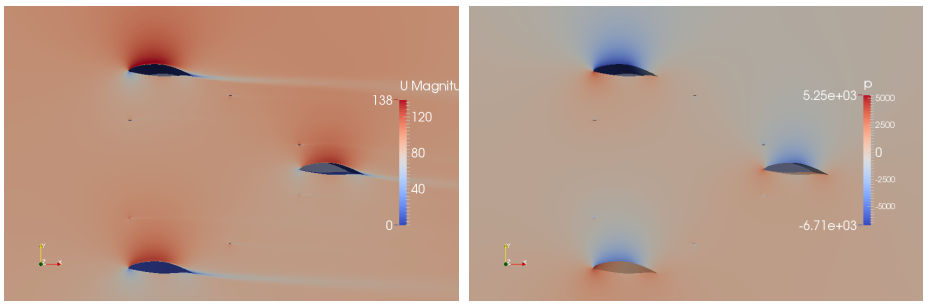
(a) *Pressure* (b) *Turbulent Viscosity* $\tilde{\nu}$

Figure 12.18: *Residuals as function of number of iterations for for Spalart-Allmaras, 90 m/s.*



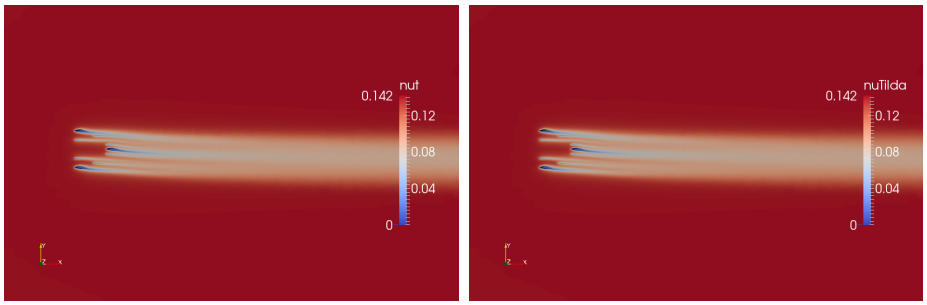
(a) *x-component of velocity* (b) *y-component of the velocity*

Figure 12.19: *Pictures for Spalart-Allmaras, 90 m/s.*



(a) *Magnitude of the velocity* (b) *Pressure*

Figure 12.20: *Pictures for Spalart-Allmaras, 90 m/s.*



(a) *Turbulent Viscosity ν* (b) *Turbulent Viscosity $\tilde{\nu}$*

Figure 12.21: *Pictures for Spalart-Allmaras, 90 m/s.*

SnappyHexMesh Results

Spalart-Allmaras

Table 12.7: Results for Spalart-Allmaras after 7000 iterations, Mesh 1

$Mesh1$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	90	6 750 000	1.28154	0.0212467	60.317
No WF	90	6 750 000	0.813672	0.0553013	14.71

Table 12.8: Results for Spalart-Allmaras after 7000 iterations, Mesh 2

$Mesh2$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	90	6 750 000	1.24941	0.0230279	54.25
No WF	90	6 750 000	0.0905974	0.0602873	15.02

Table 12.9: Results for Spalart-Allmaras after 7000 iterations, Mesh 3

$Mesh3$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	90	6 750 000	1.20363	0.0266119	45.22
No WF	90	6 750 000	1.01645	0.0450128	22.581

kkl-omega

Table 12.10: Results for kkl-ω after 7000 iterations, Mesh 1

$Mesh1$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	15	1 125 000	1.21795	0.0178978	68.05
No WF	15	1 125 000	1.21759	0.0179165	67.95

Table 12.11: Results for kkl-ω after 7000 iterations, Mesh 2

$Mesh2$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	15	1 125 000	1.223662	0.0176349	69.38
No WF	15	1 125 000	1.22238	0.0171472	71.287

Table 12.12: Results for kkl-ω after 7000 iterations, Mesh 3

$Mesh3$	$Velocity$	Re	C_l	C_d	$GlideRatio$
With WF	15	1 125 000	1.22518	0.0170231	71.97
No WF	15	1 125 000	1.22494	0.0170922	71.66

Appendix C : Triblade Case 3D

Base Case

Table 12.13: Base Case: Pressure Forces

Blades	P_x	P_y	P_z	P_{mag}
Blade A	1,67E+004	-2,34E+004	8,94E+002	28736,3534604573
Blade B	2,26E+004	-3,09E+004	-1,51E+003	38304,0087298887
Blade C	1,06E+004	-1,75E+004	-3,16E+001	20466,0979554967

Table 12.14: Base Case: Viscous Forces

Blades	V_x	V_y	V_z	V_{mag}
Blade A	-1,93E+002	-1,45E+002	1,07E+000	241,3253536696
Blade B	-2,29E+002	-1,75E+002	3,10E-001	288,3336850916
Blade C	-2,14E+002	-1,47E+002	3,39E-001	259,5604951693

Table 12.15: Base Case: Lift and Drag Forces, tangential component of the forces is denoted F_{tang}

Blades	F_{lift}	F_{drag}	F_{tang}	$Glide - Ratio$
Blade A	28695,8595176596	-1401,9146247908	8030,7814958815	-20,4690492632
Blade B	38210,091532773	-2333,2922994864	11147,2197383314	-16,376041502
Blade C	20472,4533458349	673,5278345165	4101,4986191079	30,3958534402

Only Blades Case

Table 12.16: Only Blades Case: Pressure Forces

Blades	P_x	P_y	P_z	P_{mag}
Blade A	1,93E+004	-2,61E+004	9,95E+002	32464,818727395
Blade B	2,29E+004	-3,13E+004	-1,53E+003	38842,2642469659
Blade C	1,11E+004	-1,83E+004	-2,75E+001	21389,9740342838

Table 12.17: Only Blades Case: Viscous Forces

Blades	V_x	V_y	V_z	V_{mag}
Blade A	-1,95E+002	-1,50E+002	3,18E-001	245,9739434359
Blade B	-2,31E+002	-1,76E+002	-8,75E-002	290,0770837202
Blade C	-2,18E+002	-1,50E+002	-3,12E-001	264,5298101115

Table 12.18: Only Blades Case: Lift and Drag Forces, tangential component of the forces is denoted F_{tang}

Blades	F_{lift}	F_{drag}	F_{tang}	$Glide - Ratio$
Blade A	32383,1936946932	-2156,3328272263	9621,2742893313	-15,0177158581
Blade B	38746,6212623683	-2370,5606915707	11308,1262315621	-16,3449184828
Blade C	21398,4214684961	581,2107533785	4406,4299514077	36,8169744694

Table 12.19: Only Blades Case: Force difference relative to the base case in %

Blades	F_{pmag}	F_{vmag}	F_{lift}	F_{drag}	$Glide - Ratio$
Blade A	13	1,9	12,8	53,8	-26,6
Blade B	1,4	0,6	1,4	1,6	-0,2
Blade C	4,5	1,9	4,5	-13,7	21,1

6AOA Case

Table 12.20: 6AOA Case: Pressure Forces

Blades	P_x	P_y	P_z	P_{mag}
Blade A	1,82E+004	-2,46E+004	9,39E+002	30585,5397552583
Blade B	2,49E+004	-3,30E+004	-1,64E+003	41343,7561559296
Blade C	1,17E+004	-1,90E+004	-2,20E+001	22309,1972591234

Table 12.21: 6AOA Case: Viscous Forces

Blades	V_x	V_y	V_z	V_{mag}
Blade A	-1,86E+002	-1,43E+002	7,83E-001	234,420603693
Blade B	-2,25E+002	-1,75E+002	4,52E-001	284,642296235
Blade C	-2,14E+002	-1,49E+002	2,50E-001	261,1668628376

Table 12.22: 6AOA Case: Lift and Drag Forces, tangential component of the forces is denoted F_{tang}

Blades	F_{lift}	F_{drag}	F_{tang}	$Glide - Ratio$
Blade A	30510,9366462201	-2007,9380191367	9041,9379046565	-15,1951585933
Blade B	41194,9790261683	-3161,8772837453	12646,6423665129	-13,0286457472
Blade C	22320,4696434405	461,1700462064	4737,4154688886	48,3996517706

Table 12.23: 6AOA Case: Force difference relative to the base case in %

Blades	F_{pmag}	F_{vmag}	F_{lift}	F_{drag}	$Glide - Ratio$
Blade A	6,4	-2,9	6,3	43,2	-25,8
Blade B	7,9	-1,3	7,8	35,5	-20,4
Blade C	9,0	0,6	9,0	-31,5	59,2

4AOA Case

Table 12.24: 4AOA Case: Pressure Forces

Blades	P_x	P_y	P_z	P_{mag}
Blade A	1,56E+004	-2,25E+004	8,59E+002	27353,057779192
Blade B	2,03E+004	-2,87E+004	-1,38E+003	35214,8548750089
Blade C	9,62E+003	-1,63E+004	-3,85E+001	18944,7064723436

Table 12.25: 4AOA Case: Viscous Forces

Blades	V_x	V_y	V_z	V_{mag}
Blade A	-1,98E+002	-1,47E+002	8,52E-001	246,877493927
Blade B	-2,32E+002	-1,74E+002	2,74E-001	289,853100322
Blade C	-2,13E+002	-1,45E+002	1,91E-002	257,8003495264

Table 12.26: 4AOA Case: Lift and Drag Forces, tangential component of the forces is denoted F_{tang}

Blades	F_{lift}	F_{drag}	F_{tang}	$Glide - Ratio$
Blade A	27331,375447243	-964,2487725467	7288,0667572799	-28,3447344974
Blade B	35160,7211510769	-1593,693604063	9719,3661968878	-22,062409651
Blade C	18945,1852771556	826,5236641462	3597,8422855627	22,9215279598

Table 12.27: 4AOA Case: Force difference relative to the base case in %

Blades	F_{pmag}	F_{vmag}	F_{lift}	F_{drag}	$Glide - Ratio$
Blade A	-4,8	2,3	-4,8	-31,2	38,5
Blade B	-8,1	0,5	-8	-31,7	34,7
Blade C	-7,4	-0,7	-7,5	22,7	-24,6

C-Rotated Case

Table 12.28: C-Rotated Case: Pressure Forces

Blades	P_x	P_y	P_z	P_{mag}
Blade A	2,04E+004	-2,67E+004	1,02E+003	33605,0958728296
Blade B	2,57E+004	-3,45E+004	-1,71E+003	43050,7988823984
Blade C	2,23E+004	-3,90E+004	2,07E+002	44937,0602134773

Table 12.29: C-Rotated Case: Viscous Forces

Blades	V_x	V_y	V_z	V_{mag}
Blade A	-1,84E+002	-1,45E+002	2,31E-001	234,3747123052
Blade B	-2,33E+002	-1,80E+002	6,32E-002	294,7746542471
Blade C	-1,97E+002	-1,14E+002	-3,75E-001	228,0610343367

Table 12.30: C-Rotated Case: Lift and Drag Forces, tangential component of the forces is denoted F_{tang}

Blades	F_{lift}	F_{drag}	F_{tang}	$Glide - Ratio$
Blade A	33474,3451064814	-2821,2667302956	10521,5312928579	-11,8650054414
Blade B	42911,3031287784	-3094,7515966524	12980,1264652546	-13,8658311624
Blade C	44887,362214814	2132,6551199011	8354,8911881312	21,0476423478

Table 12.31: C-Rotated Case: Force difference relative to the base case in %

Blades	F_{pmag}	F_{vmag}	F_{lift}	F_{drag}	$Glide - Ratio$
Blade A	16,9	-2,9	16,7	101,2	-42,0
Blade B	12,4	2,2	12,3	32,6	-15,3
Blade C	119,6	-12,1	119,3	216,6	-30,8

Bibliography

[1] Hamidreza Abedi. "A simpleFoam tutorial". In: (2011). DOI: http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2011/HamidrezaAbedi/report_task_3.pdf.

[2] *AERODYNAMICS AND THEORY OF FLIGHT*. URL: http://www.langleyflyingschool.com/Pages/CPGS+4+Aerodynamics+and+Theory+of+Flight+Part+1.html.

[3] *Airfoil Data*. URL: http://wind.nrel.gov/airfoils/Coefficients/.

[4] John D. Anderson. *Fundamentals of Aerodynamics 4th Edition*. 2007.

[5] JONAS BREDBERG. "On the Wall Boundary Condition for Turbulence Models". In: (2000). DOI: http://www.tfd.chalmers.se/~lada/postscript_files/jonas_report_WF.pdf.

[6] Davor Cokljat D. Keith Walters. "A Three-Equation Eddy-Viscosity Model for Reynolds-Averaged Navier-Stokes Simulations of Transitional Flow". In: *Journal of Fluids Engineering* 130 (D2008). DOI: http://www.ewp.rpi.edu/hartford/~ferraj7/ET/Other/References/Walters2008.pdf.

[7] Karl Hylen. "OpenFOAM Calculations on Winfoor Triblades, Internal Report". In: (2014).

[8] A.L. Rogers J.F. Manwell J.G. Mcgowan. *Wind Energy Explained: Theory,Design and Application*. 2009.

[9] *Lift induced drag*. URL: http://en.wikipedia.org/wiki/Lift-induced_drag.

[10] *Lift induced drag*. URL: https://johncarlosbaez.wordpress.com/2012/05/30/fluid-flows-and-infinite-dimensional-manifolds-part-3/.

[11] *OpenFOAM User Guide: 4.1 File structure of OpenFOAM cases*. URL: http://cfd.direct/openfoam/user-guide/case-file-structure/.

[12] *OpenFOAM User Guide: 4.4 Numerical schemes*. URL: http://cfd.direct/openfoam/user-guide/fvschemes/.

[13] *OpenFOAM User Guide: 5 Mesh generation and conversion*. URL: http://cfd.direct/openfoam/user-guide/mesh/.

[14] *OpenFOAM User Guide: 5.2 Boundaries*. URL: http://cfd.direct/openfoam/user-guide/boundaries/.

[15] S.B. Pope. *TURBULENT FLOWS*. 2000.

[16] Gerald Recktenwald. "The SIMPLE Algorithm for Pressure-Velocity Coupling". In: (2014). DOI: http://web.cecs.pdx.edu/~gerry/class/ME448/notes/pdf/SIMPLEslides.pdf.

[17] Abdulnaser Sayma. *Computational Fluid Dynamics*. 2009.

[18] *Standard Solvers*. URL: http://www.openfoam.org/features/standard-solvers.php.

[19] *The Dreese Airfoil Primer*. URL: http://www.dreesecode.com/primer/airfoil4.html.

[20] Kyle Mooney Tomislav Maric Jens Höpken. *The OpenFOAM Technology Primer*. 2014.

[21] *Turbulence Modeling Resource*. URL: http://turbmodels.larc.nasa.gov/spalart.html.

[22] Blogger Template by Way2blogging. *Wind Turbine Design*. 2011. URL: http://www.learnengineering.org/2013/08/Wind-Turbine-Design.html.