Simulation Verification and Optimization of a Vertical Axis Wind Turbine using CFD

Bachelor Thesis

Matthias Kortleven

S. Kenjeres  
A.W. Heemink  
Y.M. Blanter  
M.B. van Gijzen  
E.M. van Elderen  

Transport Phenomena,  
Department of Chemical Engineering,  
Delft University of Technology  

July, 2016
Contents

Abstract iii

1 Introduction 1

2 Theory 3

2.1 Physics of the Vertical Axis Wind Turbine 3

2.2 Turbulence Modeling 5

2.2.1 Reynolds Averaged Navier-Stokes Equations 6

2.2.2 Reynold Stresses 7

2.2.3 The k-equation 8

2.2.4 The k-ε Model 9

2.2.5 The k-ω Model 11

2.2.6 The k-ω SST Model 11

2.2.7 The Transition SST Model 11

2.2.8 Model Assumptions 12

2.3 Discretization Schemes 12

2.3.1 Spatial Discretization 13

2.3.1.1 First Order Upwind Scheme 14

2.3.1.2 Second Order Upwind Scheme 14

2.3.2 Temporal Discretization 14

2.3.2.1 First Order Implicit Scheme 14

2.3.2.2 Second Order Implicit Scheme 15

2.3.3 Evaluation of Gradients and Derivatives 15

2.3.4 The Pressure-Based Solver 16

2.3.4.1 Pressure Interpolation Schemes 17

2.3.4.2 The Coupled Algorithm 17

2.3.4.3 Under-Relaxation in the Iterative Process 18

2.3.4.4 Overview of the Time-Advancement Algorithm of FLUENT 18

3 Simulation Procedure 19

3.1 Overview of the Experimental Turbine and Data 19

3.2 Meshing 21

3.3 Solver Settings 25

3.3.1 General Solution Setup 25

3.3.2 Models 25

3.3.3 Materials 26

3.3.4 Cell Zone Conditions 26

3.3.5 Boundary Conditions 26

3.3.6 Mesh Interfaces 26

3.3.7 Reference Values 26

3.3.8 Solution Methods 26

3.3.9 Monitors 26

3.3.10 Solution Initialization 27

3.3.11 Calculation Settings 27

3.3.12 Automation of the Set-up of the Solver 27

3.4 Verification of the Simulations against the Experimental Data 27

3.5 Determining the ‘optimal’ Solution Method 27
4 Results and Discussion
4.1 Verification of the Models against the Experimental Data 29
4.2 Determining the ‘optimal’ Solution Method 30
   4.2.1 ‘Optimal’ Spatial Discretization 30
   4.2.2 ‘Optimal’ Temporal Discretization 34
4.3 Physical Verification of the Results 34
4.4 Results of the 3D simulation 35

5 Conclusion 43

References 44

Appendix A: ANSYS Journal Code Example 46

Appendix B: Matlab Scripts 50
Abstract

In this research computational fluid dynamics (CFD) is used to model a vertical axis wind turbine. Well known turbulence models — i.e. the $k-\varepsilon$ and $k-\omega$ SST model — are used and compared in their performance against each other and a set of experimental data. The formal research question is “What models are available for the simulation of vertical axis wind turbines and what considerations should be taken into account to most effectively obtain the power coefficient of those turbines?”

The results show that the 2D $k-\varepsilon$ simulations approximate the experimental data the closest, but that the difference between the simulations and the experiment in the $k-\omega$ model can be explained by the simulations being 2D and the experiment being 3D. To improve the results and verify that a 3D simulation indeed does produce better results, another study should be conducted in 3D when more computation power is readily available. As to what considerations should be taken into account: Meshing determines largely how long a simulation will take and to what accuracy a result can be calculated. The sliding mesh method should be used to decrease calculation time, so that the mesh does not need to be recalculated every time step. The orders of the different variables should be set to second order accuracy for the pressure $p$, the momentum, the turbulent kinetic energy $k$ and the time derivative. The second order calculation of these variables results in a significant increase in accuracy of the simulations relative to simulations conducted in first order accuracy.

For further improvement a research could be set up that involves both the experimental and the CFD part. That way the practical limitations to building a wind turbine could be modelled in the simulations, possibly resulting in even better synergy between the measurements and the simulations.
1 Introduction

Society as we know it today would not exist without electricity. In recent years the focus of energy production has shifted from fossil fuels to sustainable energy, as the fossil fuels are starting to run out. Examples of sustainable energy include the use of solar power, power extracted from water (hydroelectric power stations) and wind energy, which is the focus of this research. Wind energy covers just over 15 percent of the total energy generated in the EU as of 2015. [23] The main source of wind energy are the wind farms spread out over Europe, mostly consisting of regular horizontal axis wind turbines. This research will however focus on a less known type of wind turbine, the vertical axis wind turbine (VAWT), specifically the H-type VAWT. An overview of a few different types of turbines is given in figure 1.1.

![Regular horizontal axis wind turbine](a)

(a) Regular horizontal axis wind turbine, as seen regularly in for example wind farms. [A]

![Small scale H-type vertical axis wind turbine](b)

(b) Small scale H-type vertical axis wind turbine, the type that is used in this research. [B]

![Darrieus type vertical axis wind turbine](c)

(c) Darrieus type vertical axis wind turbine, also called 'the eggbeater', because of its shape. [C]

Figure 1.1: Different types of wind turbines.

Advantages of the VAWT over regular HAWT’s are numerous. First is that they are omni-directional, so no complicated mechanisms are needed to direct the turbine along the wind. Furthermore, VAWT’s are mechanically more simple than HAWT’s, as their blades have a less complicated airfoil and their gearbox can simply be set up at the ground, where the central hub is attached. This last point makes them easier to build and maintain, as when something breaks, the repair can most of the time be performed on the ground level. Lastly, VAWT’s can be grouped more closely in wind farms and can even be set up under existing wind farms to maximize performance. [20] There are however a few big disadvantages, the most important of which is the problems with dynamic stall due to the quickly varying angle of attack. [3] This effect causes the turbine to have an unsteady torque output over a rotation, which results in varying net forces on the blades. Up until recently — with the rise of composite materials — no suitable materials were available that could withstand this kind of load without breaking too soon.

Interest for this kind of wind turbine has risen in the past couple of years because of a few niche applications, for example in urban areas. [9] VAWT’s in general can be better integrated in building designs and with their lower rotational speed than HAWT’s, give a safer alternative turbine that produces less noise and vibrations in the process. The omni-directional working of the turbine also has an advantage in this scenario. In order to create vertical axis wind turbines that produce power as efficiently as possible, it is important that the energy production of a turbine can be calculated beforehand. An important tool in the prediction of the energy production of wind turbines in general is computational fluid dynamics (CFD). Simulating VAWT’s however, is significantly harder than simulating HAWT’s, because of the constantly changing angle of attack. Where a steady state simulation
on a single blade of a HAWT can be enough to determine its power output to decent accuracy, a full transient simulation on three blades has to be done on VAWT’s to obtain a good result. This research dives in the CFD simulation of VAWT’s with ANSYS [1] and tries to find a compromise between calculation time and reliable results using real experimental results. The research question therefore is: “What models are available for the simulation of vertical axis wind turbines and what considerations should be taken into account to most effectively obtain the power coefficient of those turbines?”

This research is carried out at the department of Chemical Engineering at the Delft University of Technology and is part of the Applied Physics and Applied Mathematics bachelor program. I would like to thank Dr. Sasa Kenjeres and Prof. Arnold Heemink for their assistance and insight and making this project possible.
2 Theory

The main goal of this research is the accurate and efficient modeling of vertical axis wind turbines (VAWT). To obtain this goal the commercial computational fluid dynamics (CFD) package ANSYS FLUENT is used. In order to be able to understand the results some theoretical background is required. This chapter is split up into three major sections: the physics of a VAWT, a piece on turbulence modeling and a section devoted to the discretization of those turbulence models.

2.1 Physics of the Vertical Axis Wind Turbine

To get a feeling for the characteristics of a VAWT and to define the variables of interest, the theoretic part of this research will start with a short overview of the physics behind such a wind turbine. In figure 2.1 a VAWT is cut in half perpendicular to its axis of rotation. This gives a top-down view of the VAWT, halfway through the geometry while under operation.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.1.png}
\caption{Schematic representation of a Darrieus type VAWT. Shown are the relevant wind velocities and angles. \(\theta\) designates the current position of the blade, \(v_\infty\) is the wind velocity 'far' in front of the turbine, \(v_a\) is the wind velocity at the blade, \(v_w\) is the velocity in the wake of the turbine, \(\omega\) is the angular velocity of the turbine and \(R\) is the radius.}
\end{figure}

The variables in figure 2.1 are defined in the following way: \(\theta\) is the angle between the current position of the blade and a reference point, in this case defined at the top of the circle. \(v_\infty\) is the wind velocity ‘far’ in front of the wind turbine, i.e. where the turbine does not influence the wind velocity yet. \(v_a\) is the wind velocity at the blade. Note that this velocity can vary as \(\theta\) varies and is by no means necessarily a constant. There are models available that provide an expression for \(v_a\), for example the single stream tube model as described by Kevat et al. [2] Continuing, \(v_w\) is the velocity in the wake of the turbine. The angular velocity of the turbine is \(\omega\), which — with the radius equal to \(R\) — gives a tip velocity of \(v_r = \omega \times R\). The tangential velocity \(v_t\) and the normal velocity \(v_n\) of the blade with respect to the wind are given by:

\begin{align}
\vec{v}_t &= \vec{v}_r + \|\vec{v}_a\| \cos(\theta) \hat{\vec{v}}_r \\
\vec{v}_n &= \|\vec{v}_a\| \sin(\theta) \hat{\vec{R}}
\end{align}

Here \(\hat{\vec{\cdot}}\) is the unit vector in the direction of the used vector. The relative velocity of the wind with respect to the blade is then given as \(\vec{v}_{rel} = -\vec{v}_t - \vec{v}_n\). Lastly, the angle of attack \(\alpha\) is defined as the angle between \(-\vec{v}_{rel}\) and \(\vec{v}_t\) as pictured in the figure, or in formula form:

\[\alpha = \arccos \left( \frac{-\vec{v}_{rel} \cdot \vec{v}_t}{\|\vec{v}_{rel}\| \|\vec{v}_t\|} \right)\]

Now, the wind exerts a force on the blade. This force can be split up into a drag force \(\vec{F}_D\) along the effective wind vector \(\vec{v}_{rel}\) and a lift force \(\vec{F}_L\) perpendicular to it. A schematic representation of
these forces is displayed in figure 2.2.

![Figure 2.2: Schematic representation of the forces on a single blade. The relative wind causes a drag force along the direction of the wind and a lift force perpendicular to the wind vector. Both forces can be split up into a force along the direction of movement of the blade and a normal force perpendicular to the moving direction.](image)

Next to the already mentioned drag force and lift force two additional forces are displayed in figure 2.2: the tangential force $F_t$ along the path of the blade and the normal force $F_n$ perpendicular to the path of the blade. It is assumed here that the instantaneous drag and lift forces are a function of the angle of attack $\alpha$ in steady flow. These forces then are the result of taking the components of the drag force and lift force in their respective directions. The lift and drag coefficient are then given as follows:

**Definition 2.1 ([19] [6])**

The lift coefficient $C_L$ and drag coefficient $C_D$ are defined as:

$$C_L = \frac{||F_L||}{\frac{1}{2} \rho ||v_{rel}||^2 S} \quad C_D = \frac{||F_D||}{\frac{1}{2} \rho ||v_{rel}||^2 S}$$

Here $S$ is the effective area of the blade.

Since these coefficients are proportional to their respective forces, the following also holds for the coefficients of the tangential force and normal force (defined analogously to the coefficients in definition 2.1):

$$C_t = C_L \sin(\alpha) - C_D \cos(\alpha) \quad (2.3a)$$

$$C_n = C_L \cos(\alpha) + C_D \sin(\alpha) \quad (2.3b)$$

The total torque $T_B$ exerted over a full rotation of a 'slice' of VAWT of height $H$ with $N$ blades of radius $R$ and chord length $C$ is then given by:

$$T_B = \frac{NCHR}{2\pi} \int_0^{2\pi} \frac{1}{2} \rho ||v_{rel}||^2 C_t \, d\theta \quad (2.4)$$

From this quantity, the coefficient of moment is easily calculated. This coefficient is defined as follows:
\textbf{Definition 2.2 ([2])}

The definition of the coefficient of moment $C_m$ is:

\[ C_m = \frac{T_B}{\frac{1}{2} \rho |v_\infty|^2 AR} = \frac{NC}{2\pi \rho R |v_\infty|^2} \int_0^{2\pi} \frac{1}{2} \rho |v_{rel}|^2 C_t \, d\theta , \]

where $A$ is the area swept by the (slice of) VAWT as seen from the front.

Furthermore, the shaft power and maximum available power are given by:

\begin{align*}
  P &= \omega T_B \quad \text{(2.5a)} \\
  P_{\text{max}} &= \frac{1}{2} \rho |v_\infty|^3 A \quad \text{(2.5b)}
\end{align*}

Finally, the quantity we are mainly interested in — the coefficient of power — is given in definition 2.3:

\textbf{Definition 2.3 ([2] [17] [9])}

The coefficient of power $C_p$ is given by:

\[ C_p = \frac{P}{P_{\text{max}}} = \frac{NC\omega}{2\pi \rho |v_\infty|^3} \int_0^{2\pi} \frac{1}{2} \rho |v_{rel}|^2 C_t \, d\theta = \lambda C_m , \]

where $\lambda$ is the tip speed ratio defined as:

\[ \lambda = \frac{\omega R}{|v_\infty|} \]

The reader may already have noticed that this approach does not explicitly take into account effects like blade tip vortices or stall phenomena, which do play a significant role in the determination of the power coefficient. Furthermore, the torques and coefficients of the extra 3D geometry needed for a physical VAWT (e.g. the struts connecting the central hub to the blades) are not explicitly mentioned, but can be determined analogously to those of the blades. This is however a numerical study, and the software that is used delivers data in the form of the coefficient of moment $C_m$. This means that together with the equations in definition 2.3, there is enough information to calculate the power coefficient directly. Because the $C_m$ of the program is calculated numerically, all previously mentioned effects regarding the airflow present in the model are taken into account, just as the extra geometry. This section is not to be seen as a perfect analytical solution to the problem of vertical axis wind turbines — unfortunately there exists no such thing — but more as a list of definitions that indicate what every variable is and what its relations are to other variables.

\subsection*{2.2 Turbulence Modeling}

Turbulence is characterized by its highly irregular and chaotic behavior. It is a three dimensional and time-dependent phenomenon with a random nature that is hard to solve deterministically. Statistical averaging is required to approximate the random fluctuations in turbulent flows. This however, leads to new terms in the governing equations that need to be modeled in order to be able to close the problem. In this section a brief overview of the models used in this research is given.
2.2.1 Reynolds Averaged Navier-Stokes Equations

Turbulence modeling starts at the original Navier-Stokes equations, where incompressible flow is assumed. The governing equations for this case are: [19]

\[ \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 , \]

(Continuity equation)

where \( \nabla \) is the del operator, \( \rho \) is the density (which is assumed constant, because of incompressible flow) and \( \mathbf{u}(x, t) \) is the velocity.

\[ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \nu \Delta \mathbf{u} = -\frac{1}{\rho} \nabla p + \mathbf{g} , \]

(Momentum equation)

where \( \Delta \) is the laplacian, \( \nu = \frac{\mu}{\rho} \) is the kinematic viscosity, \( p(x, t) \) is the pressure and \( \mathbf{g}(x, t) \) represents body accelerations per unit mass (e.g. gravity).

The reader may already have noticed that in the case of incompressible flow the Continuity equation reduces to \( \nabla \cdot \mathbf{u} = 0 \), or in other words, the divergence of \( \mathbf{u} \) is zero. This makes sense, because mass can not be created or destroyed in any control volume, it can only flow in and out.

The Momentum equation can be interpreted in words in the following way: The first two terms on the left hand side account for the inertial forces on a piece of fluid, the third term denotes the viscous forces, the first term on the right hand side accounts for the pressure forces and the last term describes the external forces. Since momentum is conserved, those quantities need to be in balance.

From these equations the Reynolds Averaged Navier-Stokes (RANS) equations can be derived. Due to the transient and three dimensional nature of turbulence we are forced to look at averaged quantities in order to be able to extract useful information. The idea behind Reynolds time-averaging is to express every variable in a mean part and a fluctuating part. So, for example, the velocity \( \mathbf{u}(x, t) \) can be expressed as \( \mathbf{u}(x, t) = \mathbf{U}(x, t) + \mathbf{u}'(x, t) \). Here the uppercase symbol denotes the time averaged component and the apostrophe denotes the fluctuating component. This convention will be used again below. The precise definition of the time averaged component is as follows:

\[ \text{Definition 2.4} \]

The time average of a variable \( \phi(x, t) \) over a time \( \tau \) is given by:

\[ \Phi(x, t) = \bar{\phi}(x, t) = \frac{1}{\tau} \int_{t}^{t+\tau} \phi(x, t) dt. \]

Here it is assumed that the average flow only varies slowly when compared to the turbulent fluctuations in the flow. This means that the flow is described approximately by stationary turbulence, i.e. for flow related variables \( \Phi(x, t) \approx \Phi(x) \). Furthermore, by definition, the average of the fluctuating component \( \mathbf{u}'(x, t) \) is zero.

Time averaging of variables in the above sense has a few interesting properties. The most important properties are listed here. All properties follow from simply working out the equations with the definitions given above.

\[ \text{Theorem 2.1} \]

Given two variables \( \phi(x, t), \psi(x, t) \) the following always holds:

(a.) \( \bar{\phi} \bar{\psi} = 0 \)
Using theorem 2.1 the Continuity equation and the Momentum equation can be rewritten in their time averaged form:

\[ \nabla \cdot U = 0, \quad (\text{RA Continuity equation}) \]
\[ \frac{\partial U}{\partial t} + U \cdot \nabla U - \nu \Delta U = -\frac{1}{\rho} \nabla P + G - \frac{1}{\rho} (ru' \cdot \nabla w'), \quad (\text{RA Momentum equation}) \]

These equations are almost identical to the original Continuity equation and Momentum equation, except for the last term, which will be dealt with in the next subsection.

### 2.2.2 Reynold Stresses

The \(-\rho u' \nabla u'\) term in the RA Momentum equation can be written as \(-\frac{\partial}{\partial x_j} (\rho u_i' u_j')\) in tensor notation. This notation will be used in this and following subsections. The expression between the brackets, \(\tau_{ij} = -\rho u_i' u_j'\), is called the Reynolds stress tensor. The first index indicates the plane along which the stress acts, and the second index indicates the direction. Using the Continuity equation an equation for the Reynold stresses can be determined, the specifics of which are beyond the scope of this research, but can be found in Celik [4]:

\[ \frac{\partial \tau_{ij}}{\partial t} + U_k \frac{\partial \tau_{ij}}{\partial x_k} = -\tau_{jk} \frac{\partial U_i}{\partial x_k} - \tau_{ik} \frac{\partial U_j}{\partial x_k} + \varepsilon_{ij} - \Pi_{ij} + \frac{\partial}{\partial x_k} \left( \nu \frac{\partial \tau_{ij}}{\partial x_k} + C_{ijk} \right) \]  

(Reynold stress equation)

where

\[ \varepsilon_{ij} = 2\mu \frac{\partial u_i'}{\partial x_k} \frac{\partial u_j'}{\partial x_k} \]
\[ \Pi_{ij} = p' \left( \frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i} \right) \]
\[ C_{ijk} = -\rho u_i' u_j' u_k' + p' u_i' \delta_{jk} + p' u_j' \delta_{ik} \]

One of the problems of turbulence modeling is that the stresses are not known on forehand, thus they need to be modeled. The concept of eddy viscosity is one method to model the Reynold stresses and is defined as follows:

**Definition 2.5**

For general flows the Boussinesq eddy viscosity approximation is given by:

\[ -u_i' u_j' = \nu_t \left( \frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij}, \text{ where } k \text{ is the turbulent kinetic energy and } \nu_t \text{ is the eddy viscosity.} \]

It then remains to find an expression for this eddy viscosity \(\nu_t\). There are several ways available to close the model, including some algebraic or mixing length models. In this research however, more sophisticated two-equation models and a four-equation model are used. These models share an equation for modeling the turbulent kinetic energy \(k\), which will be covered first.
2.2.3 The $k$-equation

Since we want to describe turbulent flows in a better way than simply using an algebraic approach — by means of using more equations — parameters which can be modeled are needed. As indicated before, the turbulent kinetic energy $k$ is a well suited candidate for the first equation. The logical choice of the characteristic velocity scale of a turbulent flow would be the square root of the kinetic energy of the turbulent fluctuations in the flow. We can therefore define the Reynolds averaged turbulent kinetic energy as:

**Definition 2.6**
The Reynolds averaged turbulent kinetic energy $k$ is given by

\[ k = \frac{1}{2} \left( u'_x u'_x + u'_y u'_y + u'_z u'_z \right). \]

One can easily see that this is exactly one half times the sum of the normal Reynolds stresses for which an expression is given in the previous subsection. Using the Reynolds stress equation an equation for $k$ can be determined, the specifics of which are beyond the scope of this research:

\[ \rho \frac{\partial k}{\partial t} + \rho \frac{\partial}{\partial x_j} (U_j k) = \tau_{ij} \frac{\partial U_i}{\partial x_j} - \mu \frac{\partial u'_i}{\partial x_k} \frac{\partial u'_k}{\partial x_j} + \frac{\partial}{\partial x_j} \left( \mu \frac{\partial k}{\partial x_j} - \frac{1}{2} \rho u'_i u'_j - p' u'_j \right) \]  
(2.6)

This equation can be interpreted physically in the following way: The terms on the left hand side denote rate of change of $k$ in time and the transport of $k$ through convection. The first term on the right hand side represents the transition from kinetic energy of the main flow to turbulent kinetic energy, or in other words, how eddies gain energy from the mean flow strain rate. The second term on the right hand side represents the rate at which $k$ is dissipated to heat on a molecular level. This dissipation is denoted by $\varepsilon$ and is given by:

**Definition 2.7**
The dissipation of turbulent kinetic energy $k \varepsilon$ is given by

\[ \varepsilon = \nu \frac{\partial u'_i}{\partial x_k} \frac{\partial u'_i}{\partial x_k}. \]

Since the $\frac{\partial u'_i}{\partial x_k}$ term is multiplied with itself $\varepsilon$ will always be positive. This means that energy will always be dissipated, i.e. the amount of ‘useful’ energy in the system will go down by this term. Moving on, the first term between the brackets represent the diffusion of $k$ through the flow and the remaining terms represent the rate at which turbulent energy is transported through turbulent fluctuations and pressure fluctuations respectively.

At this point we are still stuck with more unknowns than equations and seek for a way to close the model. The first assumption made is that the Boussinesq eddy viscosity approximation given in definition 2.5 is correct. The following then holds:

\[ \tau_{ij} = \mu_t \left( \frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) - \frac{2}{3} \rho k \delta_{ij}, \]  
(2.7)

where the eddy viscosity $\mu_t$ is now given by

\[ \mu_t \approx \rho \sqrt{kl}. \]  
(2.8)

Here $l$ is some turbulent length scale, usually similar to the mixing length of the turbulent flow. The characteristic velocity scale is given by $\sqrt{k}$, as mentioned before. Note that it is assumed here that the time scale of the turbulence is proportional to the time scale of the mean flow, an assumption that is justified for most flows considered in engineering applications, but certainly not all flows.
The focus now is on the dissipation term $\varepsilon$ given in definition 2.7. The dissipation can be estimated by considering an eddy of size $l$ moving at speed $u$. The energy dissipated per unit mass can then be written as:

$$\varepsilon \approx \frac{\text{(velocity)(drag)}}{\text{(mass)}} = \frac{u(\rho u^2 l^2)}{\rho l^3} = \frac{u^3}{l} \quad (2.9)$$

When we now take the characteristic velocity scale to be the square root of the turbulent kinetic energy as defined in definition 2.6 this expression reduces to:

$$\varepsilon \approx \frac{k\sqrt{k}}{l} \quad (2.10)$$

Lastly, the terms for transport of turbulent kinetic energy (the last two term between the brackets of equation 2.6) need to be approximated. A common method for accomplishing this is to assume that this transport depends on the gradient of the turbulent kinetic energy. Normally only terms regarding turbulent fluctuations are taken into account when using this method, but since the pressure fluctuation terms tend to be small for incompressible flows, it is common practice to take them into account as well. This results in the following approximation:

$$\frac{\mu_t}{\sigma_k} \frac{\partial k}{\partial x_j} = -\frac{1}{2} \rho \left[ u'_i, k u'_j, k + u'_k, i u'_k, j \right] \frac{\partial U_i}{\partial x_j} - \rho \varepsilon + \frac{\partial}{\partial x_j} \left( \left[ \mu + \mu_t \sigma_k \right] \frac{\partial k}{\partial x_j} \right), \quad (2.11)$$

where $\sigma_k$, known as the Prandtl-Schmidt number for $k$, is a closure coefficient.

Combining the modeled terms from equations 2.7, 2.10 and 2.11 with the original $k$-equation displayed in equation 2.6 yields: [22]

$$\rho \frac{\partial k}{\partial t} + \rho U_j \frac{\partial k}{\partial x_j} = \tau_{ij} \frac{\partial U_i}{\partial x_j} - \rho \varepsilon + \frac{\partial}{\partial x_j} \left( \left[ \mu + \mu_t \sigma_k \right] \frac{\partial k}{\partial x_j} \right), \quad (2.12)$$

where $\varepsilon$ and $\mu_t$ are given by:

$$\varepsilon = C_D \frac{k\sqrt{k}}{l} \quad (2.13a)$$

$$\mu_t = C_D \rho \sqrt{kl}, \quad (2.13b)$$

with $C_D$ and $C_D$ modeling parameters. The attentive reader may have noticed that the value of $l$ is still needed to close the model. This is indeed the case, but since the interest of this research lies with two equation models — in which the value of $l$ does not need to be known or modeled — we’ll just leave it at this.

### 2.2.4 The $k$-$\varepsilon$ Model

$\varepsilon$ as a variable has already been defined in definition 2.7. Based on the way this variable is defined, the following operator can be used to determine an equation for the turbulent dissipation $\varepsilon$:

$$L(\cdot) = \left( \frac{\partial u'_i}{\partial x_j} \right) \frac{\partial}{\partial x_j} (\cdot) \quad (2.14)$$

When this operator is applied to both sides of the incompressible Navier-Stokes equation — given in this paper as the Momentum equation — the following equation can be derived: [4]

$$\rho \frac{\partial \varepsilon}{\partial t} + \rho U_j \frac{\partial \varepsilon}{\partial x_j} = 2 \mu \left( u'_i, k u'_j, k + u'_k, i u'_k, j \right) \frac{\partial U_i}{\partial x_j} - 2 \mu_t \left( u'_i, k u'_j, k \right) \frac{\partial^2 U_i}{\partial x_k \partial x_j} - 2 \mu u'_i, k u'_k, m u'_m, k, m + \frac{\partial}{\partial x_j} \left( -\mu u'_j, k u'_i, m - 2 \nu \rho' u'_j, m + \mu_u \frac{\partial \varepsilon}{\partial x_j} \right) \quad (2.15)$$
Physically speaking, the following interpretation for this equation can be given: As usual the terms on the left hand side indicate the change in time and the transport through convection of the dissipation $\varepsilon$. The first two terms on the right hand side are due to the interactions between the mean flow and the turbulent fluctuations and represent the production of dissipation. The third and fourth term on the right hand side represent the destruction of $\varepsilon$ due to velocity fluctuations. The remaining term again represents transport due to turbulent fluctuations, pressure variations and diffusion, respectively.

Remark now that equation 2.15 represents the actual dissipation of energy of the smallest eddies, while we were actually looking for a way to express the energy dissipation in the large eddies — which delivers the largest part of the energy dissipation — at the end of the previous section. Fortunately, we do not have to meet our Waterloo here, because the dissipation $\varepsilon$ was already coupled to large scale estimates in the previous section. One could say that, because the closure approximations are all based on large scale eddies, the model represents the transfer of energy from large scale eddies to smaller ones in an empirical way.

One of the major assumptions made to close the model is that the production and dissipation of turbulent kinetic energy are in local equilibrium. In formula form this yields:

$$\tau_{ij} \frac{\partial U_i}{\partial x_j} \approx \varepsilon$$

(2.16)

This also implies that the production terms (the first two terms on the right hand side of equation 2.15) must scale with the destruction term of the $k$ equation (the first term on the right hand side of equation 2.6):

$$-2\mu \left( \frac{u'_i k'_j}{k} + \frac{u'_k u'_k}{k} \right) \frac{\partial U_i}{\partial x_j} - 2\mu \frac{u'_k u'_i}{\partial x_j} \frac{\partial U_j}{\partial x_j} \approx \frac{\tau_{ij} \frac{\partial U_i}{\partial x_j}}{t_c},$$

(2.17)

where $t_c$ is a characteristic time scale for the production of the dissipation $\varepsilon$. Because of the local equilibrium the change in $k$ should scale with the change in $\varepsilon$. This implies that:

$$t_c = \frac{k}{\varepsilon}$$

(2.18)

In a similar manner an expression for the third and the fourth term of equation 2.15 can be determined:

$$-2\mu u'_i k'_m u'_i k'_m - 2\mu u'_i k'_m u'_i k'_m \approx \frac{\varepsilon}{t_c}$$

(2.19)

The last term on the right hand side of equation 2.15 is approximated completely analogously to derivation of equation 2.11, yielding:

$$\frac{\partial}{\partial x_j} \left( -\mu u'_i u'_m u'_i, m - 2\nu p''_m u'_i, m + \mu \frac{\partial \varepsilon}{\partial x_j} \right) = \frac{\partial}{\partial x_j} \left( \left[ \mu + \mu \frac{\varepsilon}{\sigma} \right] \frac{\partial \varepsilon}{\partial x_j} \right)$$

(2.20)

Substitution of these modeled terms into equation 2.15, adding constants where necessary, yields the final $k$-$\varepsilon$ turbulence model:

**Result 2.1**

The $k$-$\varepsilon$ turbulence model consists of two extra equations alongside the Reynolds Averaged Navier-Stokes equations; [22][4]

The $k$-equation:

$$\rho \frac{\partial k}{\partial t} + \rho U_j \frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j} \left( \left[ \mu + \frac{\mu}{\sigma} \right] \frac{\partial k}{\partial x_j} \right) + \tau_{ij} \frac{\partial U_i}{\partial x_j} - \rho \varepsilon$$

The $\varepsilon$-equation:
\[ \rho \frac{\partial \varepsilon}{\partial t} + \rho U_j \frac{\partial \varepsilon}{\partial x_j} = \frac{\partial}{\partial x_j} \left( \mu + \frac{\mu_t}{\sigma_\varepsilon} \right) + C \varepsilon \frac{\tau_{ij}}{K} \frac{\partial U_i}{\partial x_j} - C_2 \rho \frac{\varepsilon^2}{K} \]

Here the eddy viscosity \( \mu_t \) is given by:

\[ \mu_t = \rho C_k k^2 \varepsilon. \]

The closure coefficients were empirically determined and are given by:

\[ C_{\varepsilon 1} = 1.44, \quad C_{\varepsilon 2} = 1.92, \quad C_\mu = 0.09, \quad \sigma_k = 1, \quad \sigma_\varepsilon = 1.3 \]

### 2.2.5 The \( k-\omega \) Model

The \( k-\omega \) model is similar to the \( k-\varepsilon \) model, where \( \omega \) in this case is defined in definition 2.8.

\[ \text{Definition 2.8} \]

The specific dissipation rate \( \omega \) is given as \( \omega = c \sqrt{\frac{k}{l}} \), where \( c \) is a constant.

The subtle difference with the \( k-\varepsilon \) model is that \( \varepsilon \) solves for the destruction of turbulent kinetic energy \( k \), while \( \omega \) solves for the rate at which this destruction occurs. To further explain this: \( \omega \) is dimensionally the same as \( \frac{k}{l} \). Again considering the processes listed before (convection, diffusion, production, destruction) the \( \omega \)-equation can be derived analogously to the \( \varepsilon \)-equation. When balancing for the dimensions of the variables the following result can be derived:

\[ \text{Result 2.2} \]

The \( k-\omega \) turbulence model consists of two extra equations alongside the Reynolds Averaged Navier-Stokes equations; [22][4]

**The \( k \)-equation:**

\[ \rho \frac{\partial k}{\partial t} + \rho U_j \frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j} \left( \mu + \sigma_k \mu_t \right) \frac{\partial k}{\partial x_j} + \tau_{ij} \frac{\partial U_i}{\partial x_j} - \beta^* \rho k \omega \]

**The \( \omega \)-equation:**

\[ \rho \frac{\partial \omega}{\partial t} + \rho U_j \frac{\partial \omega}{\partial x_j} = \frac{\partial}{\partial x_j} \left( \mu + \sigma_\omega \mu_t \right) \frac{\partial \omega}{\partial x_j} + \alpha \frac{\omega}{k} \tau_{ij} \frac{\partial U_i}{\partial x_j} - \beta \rho \omega^2 \]

Here the eddy viscosity \( \mu_t \) is given by: \( \mu_t = \frac{pk}{\omega} \).

The closure coefficients were empirically determined by Wilcox [22] and are given by:

\[ \alpha = 0.5, \quad \beta = 3, \quad \beta^* = 0, \quad \sigma_\omega = 1, \quad \sigma_k = 1 \]

### 2.2.6 The \( k-\omega \) SST Model

Another widely used turbulence model is the the \( k-\omega \) SST model. It combines the \( k-\varepsilon \) model and the \( k-\omega \) model, using the first in areas of free shear flow and the latter in areas near the wall, taking advantage of the enhanced simulation accuracy near the wall of the \( k-\omega \) model, while also having the favorable properties of the \( k-\varepsilon \) model far away from walls in the free stream. The specifics of this model are beyond the scope of this research and can be found in Menter [12].

### 2.2.7 The Transition SST Model

The transition SST model is another turbulence model that will be used in this research. It is an extension of the \( k-\omega \) SST model, adding two equations to that model, one for intermittency and one for the transition criterion in terms of the momentum thickness Reynolds number. In short it provides a way to simulate laminar sub-layers within the turbulent flow. Here the intermittency can be thought
of as a measure of whether the SST model should be used or a laminar model, while the other equation controls this transition criterion. Again the specifics are beyond the scope of this research and can this time be found in Langtry and Menter [13]. This model is included mainly for its possibilities for user input, with which the model can be calibrated, as was demonstrated in previous research by Lanzafame, Messina and Mauro [16].

2.2.8 Model Assumptions

To underline the assumptions made in the simplifications of the two-equation models — and hence the four equation model as well — this subsection will list them one more time. The first major assumption states that the turbulent fluctuations are equal locally, something that is true for the small scale eddies, but not necessarily true for the larger ones. This means that the normal Reynolds stresses are considered equal in all directions. The second major assumption was stated before in equation 2.16, it is assumed that the production and dissipation of turbulent kinetic energy $k$ are approximately the same locally.

2.3 Discretization Schemes

Disclaimer: As ANSYS FLUENT is commercial software, not all code and inner workings are available for review. This section is written with the information publicly available in the online manual of FLUENT, which means there will unfortunately be some small gaps in the theory of this section.

The models described above are generally considered fit for use in computational fluid dynamics. They can’t however be put to use without the right discretization, since it is not possible to evaluate the models over infinitesimally small pieces of volume, as would be needed to obtain ‘the’ solution. Compromises have to be made, meaning that the control volume we are interested in can only be divided in a finite amount of cells. Therefore the differential equations also need to be discretized. The easiest way of illustrating this is considering the unsteady conservation transport equations of certain scalar $\phi$. Such an equation in integral form for a certain arbitrary control volume $V$ looks as follows:

$$
\int_V \frac{\partial \rho \phi}{\partial t} \, dV + \oint_S \rho \phi \mathbf{u} \cdot d\mathbf{A} = \oint_S \Gamma_\phi \nabla \phi \cdot d\mathbf{A} + \int_V S_\phi \, dV
$$

Equation 2.21 is applied to all cells in the computational domain. A two dimensional example of this is given in figure 2.3.

![Figure 2.3: A set of two dimensional cells representing control volumes. The vectors in these cells illustrate the discretization of a scalar transport equation.](image-url)
Discretization of equation 2.21 yields:

\[
\frac{\partial \rho \phi}{\partial t} V + \sum_{\text{enc.faces}} \rho_f u_f \phi_f \cdot A_f = \sum_{\text{enc.faces}} \Gamma_{\phi} \nabla \phi_f \cdot A_f + S_{\phi} V
\]  

(2.22)

Here the subscript \( \cdot f \) denotes value of a variable at an enclosing face \( f \). An example for the Momentum equation is given in example 2.1:

**Example 2.1 (Discretization of the Momentum equation)**

When we take the Momentum equation from the start of the previous subsection and integrate over a control volume \( V \) we obtain:

\[
\int_V \left[ \frac{\partial u}{\partial t} + u \cdot \nabla u \right] dV = \int_V \left[ -\frac{1}{\rho} \nabla p + \nu \Delta u + g \right] dV
\]

(2.23)

If this equation is now split up into its \( x, y \) and \( z \) components it can be simplified further. Continuing in tensor notation and applying the divergence theorem to the advection, pressure gradient and diffusion terms then yields:

\[
\frac{\partial u_i}{\partial t} V + \int_S u_i u_j n_j \ dA = -\frac{1}{\rho} \int_S p n_i \ dA + \int_S \nu \frac{\partial u_i}{\partial x_j} n_j \ dA + g_i V ,
\]

(2.24)

where \( n \) is the normal of a surface of the control volume \( V \). If the values of variables on a face (e.g. the side of a triangle, if the control volume is a triangle) are assumed constant the discretization becomes:

\[
\frac{\partial u_i}{\partial t} V + \sum_{\text{enc.faces}} u_i,u_j,f n_j,f A_f = -\frac{1}{\rho} \sum_{\text{enc.faces}} p n_i,f A_f + \sum_{\text{enc.faces}} \nu \frac{\partial u_i,f}{\partial x_j} n_j,f A_f + g_i V
\]

(2.25)

This equation, and most discretization equations alike, is not linear with respect to the variable that is sought. In order to be able to solve the problem the system is linearized. [8] A general linearized form of equation 2.22 is given by:

\[
a_p \phi = \sum_{nb} a_{nb} \phi_{nb} + b ,
\]

(2.26)

where \( \cdot \ nb \) refers to neighboring cells and \( a_p \) and \( a_{nb} \) are the linearized coefficients for \( \phi \) and \( \phi_{nb} \) respectively. The amount of neighbors depends completely on the geometry of the cells, but — as a rule of thumb — equals the number of faces enclosing the cell most of the time. A linearization of equation 2.25 described in example 2.1 will be given at the end of this section, after the needed theory is given.

### 2.3.1 Spatial Discretization

FLUENT stores the values of cell variables at the center of the cells. Since the values of these variables are also needed at the faces between the cells — as was seen in equation 2.22 — interpolation must be done. FLUENT offers several schemes, including first order upwind, second order upwind, QUICK and central differencing. In this research only the first and second order upwind schemes will be used, since they can be used for most meshes in the pressure-based solver. Furthermore, they are unconditionally stable regarding the mesh size, which means that convergence will not be affected by these schemes. The following is applicable to most variables except the pressure, which is given a special treatment.
2.3.1.1 First Order Upwind Scheme

When first order accuracy is required it is simply assumed that the values of variables at the cell center hold for the whole cell. The face value $\phi_f$ in the discretization for a cell is then determined by the cell center value of the upstream cell:

$$\phi_f = \phi_{\text{upstream}}$$  \hspace{1cm} (2.27)

A visualization of this scheme is given in figure 2.4.

2.3.1.2 Second Order Upwind Scheme

When second order accuracy is required, the values at the faces are estimated by applying a correction on the cell center value of the variable of the upstream cell. This correction is determined by the Taylor expansion around the cell center of the cell we are interested in. The resulting expression for $\phi_f$ is given by:

$$\phi_f = \phi_{\text{upstream}} + \nabla \phi_{\text{upstream}} \cdot r_f ,$$  \hspace{1cm} (2.28)

where $r_f$ is the displacement vector from the upstream cell to the centroid of the face $f$. The correction value is limited such that no new minima or maxima are introduced. A visualization of this scheme is given in figures 2.5a and 2.5b.

2.3.2 Temporal Discretization

For a transient simulation discretization in space only is not enough, discretization in time is also required. FLUENT offers explicit and implicit time integration, in first and second order accuracy. In this research only implicit time integration will be used, since those schemes are unconditionally stable with regard to the time step $\delta t$. A general expression for the time evolution of a variable $\phi$ is given by:

$$\frac{\partial \phi}{\partial t} = F(\phi) ,$$  \hspace{1cm} (2.29)

where $F(\phi)$ can be any spatial discretization of the variable $\phi$.

2.3.2.1 First Order Implicit Scheme

When first order accuracy is required the following scheme can be used:

$$\frac{\phi^{n+1} - \phi^n}{\delta t} = F(\phi^{n+1}) ,$$  \hspace{1cm} (2.30)
where the superscript \( n \) indicates the value of the variable at time step \( n \). This equation describes a relation where the new value of a variable \( \phi \) depends on its old value and the new values of its neighboring cells. This equation is solved iteratively at every time step to obtain a solution. This can be done by initially setting the iteration variable \( \phi^i \) to the known previous value \( \phi^n \) and iterating the following equation:

\[
\phi^i = \phi^n + \delta t F(\phi^i) \tag{2.31}
\]

### 2.3.2.2 Second Order Implicit Scheme

When second order accuracy is required the following scheme can be used:

\[
\frac{3\phi^{n+1} - 4\phi^n + \phi^{n-1}}{2\delta t} = F(\phi^{n+1}) \tag{2.32}
\]

This equation describes a relation where the new value of a variable \( \phi \) partly depends on its old values and the new values of its neighboring cells. This equation is again solved iteratively at every time step to obtain a solution.

### 2.3.3 Evaluation of Gradients and Derivatives

The keen-eyed reader may have already noticed that the exact discretization of the gradient in equation 2.28 is still a mystery up to this point. For the evaluation of gradients a least squares cell based method was chosen, because of the wide applicability among different meshes. This method assumes that the solution varies linearly. In order to visualize this, five neighboring cells are drawn in a 2D schematic representation in figure 2.6.

As before, the variable values are stored at the cell centers. Furthermore, the displacement vector \( \delta r_i \) is given. Since it was just concluded that the least squares method assumes a linear solution to the problem, the change in cell values between cells \( c_0 \) and \( c_i \) can be written as:

\[
(\nabla \phi)_{c_0} \cdot \delta r_i = \phi_{c_i} - \phi_{c_0} \tag{2.33}
\]

Of course, similar equations can be written for the other three cells surrounding \( c_0 \). All these equations can be written into a compact system of the following form:

\[
[J] (\nabla \phi)_{c_0} = \delta \phi , \tag{2.34}
\]
Figure 2.6: Schematic representation of five neighboring cells with the values of a variable $\phi$ stored at their cell centers. The displacement vector $\delta r_i$ to one of the neighboring cells is also drawn.

where $[J]$ is the so-called coefficient matrix, purely dependent on the geometry of the problem. We are now left with an over-determined linear system that can be solved using a least squares method. The regular approach of using the normal equation however, can lead to problems in stretched meshes. Therefore the Gram-Schmidt method is used to obtain a QR-decomposition of the $[J]$ matrix: [24] [8]

$$[J] = QR,$$  \hspace{1cm} (2.35)

where $Q$ is an orthogonal matrix and $R$ an upper triangle matrix. The solution is then obtained as follows:

$$(\nabla \phi)_{c0} = R^{-1}Q^{-1} \delta \phi$$ \hspace{1cm} (2.36)

2.3.4 The Pressure-Based Solver

Returning to example 2.1, there now is enough information to further simplify equation 2.25:

**Example 2.2 (Discretization of the Momentum equation (continued))**

Filling in the results of the previous subsections gives us a linearization of equation 2.25. Looking at the first term we get (in first order approximation):

$$\frac{\partial u_i}{\partial t} V \Rightarrow \frac{u_i^n - u_i^{n-1}}{\delta t} V$$ \hspace{1cm} (2.37)

Here the other variables in equation 2.25 also get the superscript $^n$ to keep the algorithm implicit. The only other non-linear term then is the second-to-last term, which can now be rewritten to:

$$\sum_{\text{enc. faces}} \nu \frac{\partial u_i, f}{\partial x_j, f} n_{j, f} A_f = \sum_{\text{enc. faces}} \nu a_{j, u, nb} (u_i - u_{i, nb}) n_{j, f} A_f$$ \hspace{1cm} (2.38)

We are then left with only linear terms that can be written into a general form like in equation 2.26.

Rearranging terms the final discretized form of the $x$-momentum equation can be derived:

$$a_p u_x = \sum_{nb} a_{nb} u_{x, nb} + \sum_{\text{enc. faces}} P_f A_f \cdot i + S,$$ \hspace{1cm} (2.39)
where $P = \xi$ and $\hat{i}$ is the unit vector in the $x$-direction and $a$ indicates a constant. The first constant in this equation — $a_P$ — is called the momentum equation coefficient.

The discretization of the Continuity equation is given by:

$$\sum_{\text{enc. faces}} M_f A_f = 0 ,$$  \hspace{1cm} (2.40)

where $M_f$ is the mass flux through the face $f$: $\rho u_{n,f}$. The only thing left to do now is relating the face values $u_{n,f}$ to the cell center values. Linear interpolation unfortunately yields unphysical checker-boarding of the pressure. FLUENT therefore uses a momentum weighted averaging function that can be written as follows: [8]

$$M_f = \rho_f \frac{a_{P,c_0} u_{n,c_0} + a_{P,c_1} u_{n,c_1}}{a_{P,c_0} + a_{P,c_1}} + d_f ((p_{c_0} + (\nabla p)_{c_0} \cdot r_0) - (p_{c_1} + (\nabla p)_{c_1} \cdot r_1)) ,$$  \hspace{1cm} (2.41)

where the $\cdot c_0$ and $\cdot c_1$ and the displacement vectors refer to two cells in the situation described in figure 2.3 and $d_f$ is a function of the average of the $a_P$ coefficients of both sides of the face.

### 2.3.4.1 Pressure Interpolation Schemes

As mentioned above, the pressure gets a special treatment with regard to its interpolation schemes. The default scheme in FLUENT interpolates the pressure at the faces between cells using its momentum equation coefficients. [8] This method will be referenced to in the remainder of this paper as the standard method for interpolating the pressure. For the face $f$ between two cells $c_0$ and $c_1$ this method yields:

$$P_f = \frac{P_{c_0}}{a_{P,c_0}} + \frac{P_{c_1}}{a_{P,c_1}}$$  \hspace{1cm} (2.42)

This scheme works when the gradient is sufficiently low, i.e. for very high pressure differences over short distances this interpolation scheme fails and produces overshots or undershoots in the cell velocity. To fix this problem another scheme can be used. The simplest scheme is the linear scheme, which just interpolates along the displacement vector of two cells linearly. Lastly a second order interpolation scheme — as described in paragraph 2.3.1.2 — can be used to interpolate the pressure more accurately. Note however, that the second order scheme can not be used when the pressure gradient is discontinuous and that it might have trouble converging at the start of a simulation or in combination with bad meshes.

### 2.3.4.2 The Coupled Algorithm

In this research the coupled algorithm is used over the regular segregated algorithms, such as SIMPLE. The coupled solver is more efficient and thus performs better than the standard algorithms. Furthermore, for transient flows, the coupled algorithm performs much better when the mesh quality is low, or when the time steps are large. [8] In this algorithm, the continuity and momentum equations are solved simultaneously, instead of separately. The pressure gradient for a component $m$ in the momentum equation (analogous to equation 2.39) can be written in the form:

$$\sum_{\text{enc. faces}} P_f A_f \cdot \hat{m} = - \sum_{nb} a_{nb}^{u_m} P_{nb} ,$$  \hspace{1cm} (2.43)

where the coefficient $a_{j,nb}^{u_m}$ is the coefficient derived from the Gauss divergence theorem and the coefficients of the used pressure interpolation scheme. Finally then, the discretized calculation form of the momentum equation for the $m$-component in a cell $i$ is given by:

$$\sum_{nb} a_{i,nb}^{u_m} u_{m,nb} + \sum_{nb} a_{i,nb}^{u_m} P_{nb} = b_i^{u_m}$$  \hspace{1cm} (2.44)
Now substituting equation 2.41 into 2.40 and discretizing, the discretized calculation form of the continuity equation becomes:

$$\sum_{m} \sum_{nb} a_{i, nb}^P u_{m, nb} + \sum_{nb} a_{i, nb}^P P_{nb} = b_i^P$$  \hspace{1cm} (2.45)

The system of equations 2.44 and 2.45 can be written in matrix form:

$$\sum_{nb} [A]_{i, nb} X_{nb} = B_i,$$  \hspace{1cm} (2.46)

where:

$$A_{i, nb} = \begin{bmatrix} a_{i, nb}^{pp} & a_{i, nb}^{pu_x} & a_{i, nb}^{pu_y} & a_{i, nb}^{pu_z} \\ a_{i, nb}^{u_x p} & a_{i, nb}^{u_x u_x} & a_{i, nb}^{u_x u_y} & a_{i, nb}^{u_x u_z} \\ a_{i, nb}^{u_y p} & a_{i, nb}^{u_y u_x} & a_{i, nb}^{u_y u_y} & a_{i, nb}^{u_y u_z} \\ a_{i, nb}^{u_z p} & a_{i, nb}^{u_z u_x} & a_{i, nb}^{u_z u_y} & a_{i, nb}^{u_z u_z} \end{bmatrix}, \quad X_{nb} = \begin{bmatrix} p'_{i} \\ u'_{x, i} \\ u'_{y, i} \\ u'_{z, i} \end{bmatrix}, \quad B_i = \begin{bmatrix} -r_{i, p} \\ -r_{i, u_x} \\ -r_{i, u_y} \\ -r_{i, u_z} \end{bmatrix}$$  \hspace{1cm} (2.47)

$X_{nb}$ is the unknown vector and $B_i$ is the vector of residuals. This system is solved by the AMG coupled solver of FLUENT.

### 2.3.4.3 Under-Relaxation in the Iterative Process

To improve the convergence of the process explicit under-relaxation is used. This means that the correction applied to a variable each iteration is reduced by a certain factor to reduce the effects of nonlinearity. If $\alpha < 1$ is the under-relaxation factor for a variable $\phi$, this translates to the following in formula form:

$$\phi_{new} = \phi_{old} + \alpha \delta \phi,$$  \hspace{1cm} (2.48)

where $\delta \phi$ is the correction on $\phi$ calculated by the algorithms for the current iteration. FLUENT uses a value of 0.75 for the under-relaxation factor out of the box. A similar method is used to apply under-relaxation to equations.

### 2.3.4.4 Overview of the Time-Advancement Algorithm of FLUENT

A short overview of the time-advancement scheme used by FLUENT is given in figure 2.7.

![Figure 2.7: Overview of the coupled time-advancement scheme in FLUENT.](image-url)
3 Simulation Procedure

There are multiple goals in this research, all involving the simulation of a vertical axis wind turbine. The first goal is to verify that the models simulated by FLUENT are indeed a reasonable representation of reality, and to compare the results against prior research. In this part of the research three models will be compared: the \( k-\varepsilon \) model, the \( k-\omega \) model and the transition SST model. The second goal is to then optimize the simulation to take up as little time as possible, while maintaining sufficient accuracy. In order to achieve the first goal experimental data is needed. The sought data was found in a paper written by J. Kjellin et al. called "Power coefficient measurement on a 12 kW straight bladed vertical axis wind turbine". [10] This paper provides experimental data in the form of power coefficients \( C_p \) for different tip speed ratio’s \( \lambda \), while also providing a detailed description of the dimensions of the wind turbine that was used.

3.1 Overview of the Experimental Turbine and Data

The turbine used in the experiment has the following properties according to J. Kjellin et al.: [10]

\[
\begin{array}{|l|c|}
\hline
\text{Property} & \text{Value} \\
\hline
\text{Power (kW)} & 12 \\
\text{Number of blades} & 3 \\
\text{Swept area (m}^2\text{)} & 30 \\
\text{Hub height (m)} & 6 \\
\text{Turbine radius (m)} & 3 \\
\text{Blade length (m)} & 5 \\
\text{Chord length (m)} & 0.25 \\
\text{Blade airfoil} & \text{NACA0021} \\
\text{Strut airfoil} & \text{shortened NACA0025} \\
\hline
\end{array}
\]

Table 3.1: Overview of the properties and dimensions of the VAWT used in the experiment.

Note that the rated 12 kW is at a wind speed of 12 meters per second, a fairly strong wind. Next to this information, the following is known about the geometry of the turbine: the blades are straight but tapered. The tapering begins 1 meter from the tip, where the chord length at the tip is 60% of the chord length given in table 3.1. The three blades are attached to the central rotating hub by three pairs of struts. The struts have a NACA0025 inspired profile that uses this airfoil shape, but is shortened at the trailing end. A schematic representation of the strut is given in figure 3.1.

![Figure 3.1](image.png)

Figure 3.1: Dimensions of the struts connecting the central hub to the blades, top and side view respectively. The short end attaches to the blades, while the long end attaches to the central hub.
These struts are connected to the central hub at an angle of 17.6° relative to the horizontal plane. There is enough information now to reconstruct the turbine and build a model. The model was created with SOLIDWORKS 2015 [18], where the NACA airfoils were generated with a standard 4-digit NACA airfoil generator [15]. The final model is displayed in figure 3.2.

![Figure 3.2: The 3D model of the vertical axis wind turbine as described in table 3.1. Some important dimensions are indicated in the figure.](image)

Since running a simulation on this 3D model is computationally extremely expensive two ‘lesser’ versions of this full model were derived. The first being a 2D model (depicted in figure 3.3) and the second being a symmetry version of the full model. Unfortunately it is not possible to use easy symmetry planes as with a regular HAWT, where only one blade has to be simulated and symmetry can be used to obtain a solution for all blades. With a VAWT however, the blades are in each others flow, so that radial symmetry cannot be used to simplify the problem. The only symmetry plane is the one right through the wind turbine in between the struts in the $xz$-plane. This is however not a real symmetry plane, since the struts are tilted in the same direction at both sides. Furthermore the central hub extends further in one direction then in the other. If we assume that the direction of the tilt and the extension of the hub do not influence the flow significantly a symmetric version of the wind turbine can be derived. This simplified version of the turbine is shown in figure 3.3.
Figure 3.3: The first subfigure shows the 2D version of the turbine. Here the chord length is chosen to be the standard 0.25 meters and the hub diameter 0.36 meters. On the right the ‘cut-in-half’ symmetry model of the VAWT is displayed. Note that this symmetry does not account for the long part of the hub at the bottom and struts being aligned the same way.

The experimental conditions under which the turbine operated are given in table 3.2.

Table 3.2: Experimental conditions. [10] Note that the Weibull factors determine a distribution for the wind speed.

<table>
<thead>
<tr>
<th>Wind speed range (m/s)</th>
<th>0-11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weibull scale factor</td>
<td>5.24</td>
</tr>
<tr>
<td>Weibull form factor</td>
<td>1.94</td>
</tr>
<tr>
<td>Rotational speed (rpm)</td>
<td>48 and 57</td>
</tr>
<tr>
<td>Type of site</td>
<td>Field</td>
</tr>
</tbody>
</table>

Fortunately the wind turbine rotated at a fixed speed in the experiment. This means that in the simulation the boundary conditions are well-determined for a set tip speed ratio, as the velocity at the inlet can be directly calculated using definition 2.3. The data is supplied in the form of a graph, with on the x-axis the tip speed ratio $\lambda$ and on the y-axis the power coefficient $C_p$, including error bars. The data is acquired by taking 10 minute averages over measurements measured at 1 kHz, neglecting measurements where the reference wind velocity sensor was in the wake of the turbine. The results are then binned by tip speed ratio and graphed. The resulting graph can be found in figure 3.4. Note that this set of data is built up out of statistical averages of wind velocity, since it very rarely consists of a constant flow of air with same velocity. To get consistent results, the simulations will be run at fixed wind velocities. It is assumed that — since the data from the experiment is delivered in statistical averages — those situations are comparable and the results will not differ much.

3.2 Meshing

Now that the geometry of the problem is clear, we can continue onto meshing. In order to be able to create a mesh, a suitable control volume has to be defined first. The wake of a VAWT extends almost exclusively in the direction of the wind, with minimal break-out in the direction perpendicular to the wind. This means that the control volume does not have to be very extended in this perpendicular direction. Furthermore, since the interest of this research lies mainly with the power coefficient, the
control volume only has to extend to where the pressure has settled at the standard ambient pressure, since this is the boundary condition for the outflow. The control volume for the 2D simplification is given in figure 3.5.

Figure 3.5: Control volume for the 2D case. The control volume consists of two disks within a rectangle to accommodate the use of a sliding mesh further on in the process.

The total control volume consists of a rectangle of 30 meters by 16 meters. Shown are a velocity inlet,
which will be set to the wind speed at which the simulation will take place. Furthermore a pressure outlet is shown, which just lets the air flow through at ambient pressure. On the sides two symmetries are added to indicate that nothing special is happening there. Making those sides walls would cause problems later on, as we will see in the following section. Just left of the center a disk and an annulus of radius 2 meters and 4 meters respectively can be seen, where the annulus contains the blades. Both the lines connecting the disk and annulus to each other and the rest of the control volume are indicated as interface. This will become important later on when the turbine is supposed to be moving in the transient case.

An overview of the main mesh used in this research is shown in figure 3.6. This mesh, as well as other meshes in this research, is created with the ANSYS ICEM CFD mesher [1]. The dimensions in this main mesh were determined by running a test simulation at a tip speed ratio of 2.5 with a rotational speed of 48 rpm. The dimensions were lowered until the solution did not change any more.

![Figure 3.6: Overview of the mesh for the 2D case. Note that the mesh is more refined around the hub and the blades.](image)

A close up of the mesh at one of the blades reveals more information on the actual mesh. This is shown in figure 3.7. A few important things are going on in this mesh. First off, it is clear that the mesh is much more refined near the blades and the hub. This is because the used models — $k$-$\omega$ and derivatives in particular — require resolution near walls in order to work accurately. This again ties in with why the sides of the control volume are symmetries, as if they were walls, extra resolution would be needed to model them accurately, essentially throwing away computation time unnecessarily. Furthermore, the highest gradients of variables will be found close to the blades, so for accurate modelling, the resolution should be higher. Secondly, in the small square a close up of the mesh very close to the blade is shown. It clearly shows that near the blade the cells change shape and become smaller and smaller rectangles as we move closer to the blade. This layer is called an inflation layer and essentially ensures that the simulation near the wall is accurate, since the wind velocity will quickly go from high to low, as the boundary condition at the wall of course requires the wind speed to be zero.

The mesh settings are as follows: An edge sizing of 0.4 mm was applied to the blade, in combination with an inflation layer which was 50 cells thick with a growth rate of 1.08. A similar approach was
made for the hub, with the sizing being 1.5 mm and the inflation layer being 20 cells thick with a growth rate of 1.08. The overall settings for the mesh that are not default are summarized in table 3.3. These settings result in a mesh of 446,108 nodes and 443,236 cells. The average skewness has a value of $0.1 \pm 0.1$ with a maximum of 0.93. This maximum occurs in 3 cells, so that the skewness in the rest of the cells remains under 0.88. As a rule of thumb a skewness of $>0.95$ indicates bad cells, a skewness of 0.88-0.95 indicates sufficient cell quality and everything below a skewness of 0.88 indicates a good mesh quality, so we can be satisfied. Other 2D meshes used in this research are created in a similar way, where coarser meshes are always created by simply setting the edge sizing at the blades to a higher number. The blade edge sizing and the total number of cells will always be mentioned when another mesh is used.

Table 3.3: Overview of the general mesh settings used in the 2D mesh.

<table>
<thead>
<tr>
<th>Advanced Size Function</th>
<th>Proximity and Curvature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relevance Center</td>
<td>Fine</td>
</tr>
<tr>
<td>Smoothing</td>
<td>Medium</td>
</tr>
<tr>
<td>Max Size (m)</td>
<td>0.20</td>
</tr>
<tr>
<td>Max Face Size (m)</td>
<td>0.15</td>
</tr>
<tr>
<td>Growth Rate</td>
<td>1.05</td>
</tr>
</tbody>
</table>

The 3D case is a bit harder, because the required resolution to obtain a good solution is fairly high. As mentioned before, the standard mesh for the 2D case already consists of over 400,000 cells. A 3D mesh of the same resolution would contain too many cells to fit into the physical memory of a computer. Therefore a 3D mesh of about 7 million cells was created to at least give an indication of what is possible with a 3D simulation at this time. This mesh already requires a minimum of 32 GB of physical memory to run, meaning that a computation cluster is the way to go. In order to get a resolution as high as possible, the symmetry geometry is used and the actual calculation of the coefficient of moment is done on a single blade only. The other parts are modelled more course
to at least approximate the airflow around them. Even after those measures the blade edge sizing of the main blade still had to be increased to 5 mm, about ten times higher than in the 2D case. Furthermore the number of inflation layers was reduced to 10 and the blade edge sizing of the struts, hub and remaining blades was set to 1 cm. Lastly the growth rate had to be set to the higher value of 1.17. To give an impression of the 3D mesh, an overview with a cut through one of the blades is given in figure 3.8.

![3D mesh with a cut through one of the blades](image)

**Figure 3.8:** Close up of the 3D mesh with a cut through one of the blades. Note that the growth rate is significantly higher than in the 2D case and that the overall block size is larger. Further notice the outcome of the strut halfway through the blade.

### 3.3 Solver Settings

Numerous settings had to be adjusted in order to be able to run simulations. In this section a short overview of the settings that were used in FLUENT is given, presenting details where necessary. The most convenient lay-out for this section is simply holding on to the lay-out of FLUENT itself, which is the lay-out of the version of FLUENT included with ANSYS 15.

#### 3.3.1 General Solution Setup

Here the most general settings can be adjusted. The pressure-based solver is used, most of the simulations are transient and 2D and 3D settings are used when appropriate.

#### 3.3.2 Models

All available models were kept ‘off’ except the viscous model. Here $k-\varepsilon$, SST $k-\omega$ and transition SST were selected depending on the simulation.
3.3.3 Materials

For the air used in the experiment the ANSYS FLUENT standard air was used. This air has a density of $\rho = 1.225 \text{ kg m}^{-3}$ and a dynamic viscosity of $\mu = 1.7894 \cdot 10^{-5} \text{ kg m}^{-1}\text{s}^{-1}$.

3.3.4 Cell Zone Conditions

The cell zone conditions sub-menu returns a list of distinct zones in the mesh. In the case of the mesh in figure 3.6, there are three zones, the outer fluid zone, the inner fluid zone around the hub and the annulus in which the blades can be found. In order to simulate the rotating turbine, the mesh of the blade fluid zone is rotated round the center of the turbine with a constant angular velocity $\omega = -5.03 \text{ rad s}^{-1}$. This speed is negative, because the turbine should be spinning clockwise according to the geometry. This method is called the sliding mesh method, where the variables at the boundary are calculated using interpolation at every time step. This method is preferred over the dynamic or adaptive mesh method, because it does not require the mesh to be regenerated every single time step. This especially helps with more complicated meshes, such as in 3D.

3.3.5 Boundary Conditions

In this sub-menu the boundary conditions for every object in the mesh can be specified. As mentioned before in figure 3.5, the sides are symmetries and the circles are interfaces. The mesh in between is listed as interior. The blades are moving walls, which move along with the mesh with relative speed 0. The hub is a regular wall. Moving on, the outlet is a pressure outlet that only specifies the static pressure to be equal to the ambient pressure (which is set to the standard 101,325 pascal). Lastly, the inlet is a velocity inlet with a gauge pressure of 0 pascal. The velocity input value depends on the simulation and is always set perpendicular to its boundary.

3.3.6 Mesh Interfaces

The regular interfaces created automatically by FLUENT were used. Matching was checked to improve the interpolation at the interfaces.

3.3.7 Reference Values

This is an important sub-menu, as FLUENT derives its value of the coefficient of moment $C_m$ from this panel. Internally, this coefficient is calculated as described in definition 2.2. FLUENT will use the values in this panel as the values in the denominator of that expression. In the 2D case the problem is modelled as if the turbine has a Depth of 1 m, so the the area $A$ becomes $6 \text{ m}^2$, the depth 1 m, the Length $R 3 \text{ m}$ and the velocity and air properties equal to the previously mentioned boundary conditions and material properties. In the 3D symmetry case $A$ becomes $15 \text{ m}^2$ and the Depth $2.5 \text{ m}$. The full 3D case of course has a swept area $A$ of $30 \text{ m}^2$ and a depth of 5 m.

3.3.8 Solution Methods

In this sub-menu the orders in which the variables should be calculated at the faces is chosen. The Coupled scheme is used for all calculations. The Gradients are calculated with a least squares cell based approximation as described in the previous chapter. The pressure, momentum, $k$ and $\varepsilon$ or $\omega$ orders are set to the wanted values. The transient formulation is set to second order implicit for most of the simulations. High order term relaxation is enabled with under-relaxation factor 0.75.

3.3.9 Monitors

Here the residuals and the coefficients of moment are captured. The current time step is considered converged if the residuals for all equations dive under $10^{-3}$. The coefficients of moment are saved to
a file on the disk every time step for each individual object, i.e. the \( C_m \) for the blades and hub (and struts in 3D) are saved separately. Furthermore, the creation date and last modified date of these files were used to benchmark the time elapsed for every simulation.

### 3.3.10 Solution Initialization

A standard initialization is used, where the initial values are calculated from the inlet. This results in the whole flow field having the same velocity as the inlet in every cell. The pressure is initialized at ambient pressure.

### 3.3.11 Calculation Settings

This sub-menu provides the necessary settings to change the time step and the number of time steps that is calculated in the simulation. The amount of iterations per time step was set to 20 by default, which is enough to make the solution converge at every time step in the steady state situation.

### 3.3.12 Automation of the Set-up of the Solver

In order to run multiple simulations in a row without user interference the built-in ANSYS journalling system is used. This system allows for automation of ANSYS FLUENT, including all the drop-down menu’s. Code from this journalling system is quite extensive and not very enlightening, but an example script for the simulation of a single order determination data set is provided in Appendix A.

### 3.4 Verification of the Simulations against the Experimental Data

The first set of simulations is done to establish the validity of the models in practice. This is done by calculating the power coefficient \( C_p \) for different tip speed ratio’s for the different models. To obtain reliable data, the mesh and time step are first refined at a tip speed ratio of 2.5 until the solution did not change any more. The model used in this calibration is the transition SST model. All simulations are run for 8 seconds of flow time with a time step of 0.004 seconds. The mesh used is the main mesh described in section 3.2. The numerical scheme used in these simulations is the default scheme in FLUENT, which calculates momentum and pressure at second order accuracy and the remaining variables in first order accuracy. The turbine is spinning at 48 rpm, directly locking in the inlet velocity for each tip speed ratio. The inlet velocities belonging to the tip speed ratio’s are summarized in table 3.4. The results are then averaged over the last 3 rotations and plotted against the experimental results from figure 3.4 and put out against similar research.

#### Table 3.4: A list of tip speed ratio’s with their respective inlet velocities.

<table>
<thead>
<tr>
<th>TSR</th>
<th>1.5</th>
<th>2</th>
<th>2.5</th>
<th>3</th>
<th>3.5</th>
<th>4</th>
<th>4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inlet Wind Velocity (m/s)</td>
<td>10.06</td>
<td>7.55</td>
<td>6.04</td>
<td>5.03</td>
<td>4.31</td>
<td>3.77</td>
<td>3.35</td>
</tr>
</tbody>
</table>

### 3.5 Determining the ‘optimal’ Solution Method

To optimize performance an ‘optimal’ solution method is sought. Of course this ‘optimal’ method would be optimal only for the exact geometry and problem specification given in this research, but it should be a good starting point for similar research. The used model for optimization is the \( k-\omega \) model, because of its wide acceptance in turbine modelling. [21] Then the order of convergence of different combinations of the orders of solution variables as given in subsection 3.3.8 is determined, all while simultaneously clocking the time elapsed for each simulation. This is done by creating two groups of three meshes, the first group (Group X) having blade edge sizing (m) 0.005, 0.01 and 0.02 respectively and the second (Group Y) having blade edge sizing (m) 0.0004, 0.0006928 and 0.0012,
with the hub edge size scaled proportionally in both cases. The total number of cells of these meshes are respectively: 94,762, 56,479, 41,223, 443,236, 302,310 and 219,226. Two cases are taken in order to be able to say a bit more qualitatively about the simulations. The remainder of the settings is similar to the case described in the previous subsection. The approximate order of convergence $q$ can then be determined from the simulation result of a variable $\phi$ by:

$$q = \frac{1}{\log(r)} \cdot \log \left( \frac{\epsilon_{32}}{\epsilon_{21}} \right),$$

where

$$\epsilon_{ab} = \phi(r^bh) - \phi(r^{b-1}h),$$

$h$ is the characteristic length — the blade edge sizing in this case — and $r$ is the ratio between the characteristic lengths of the meshes. It follows that $r$ is 2 for the first group and 1.732 for the second group. For the variable $\phi$ the average coefficient of moment over the last rotation is taken. The different simulation settings are summarized in the first rows of table 3.5.

<table>
<thead>
<tr>
<th>Simulation #</th>
<th>p</th>
<th>Momentum</th>
<th>k</th>
<th>$\omega$</th>
<th>Time</th>
<th>BES (m)</th>
<th>Time step (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>X, Y</td>
<td>0.003</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>X, Y</td>
<td>0.003</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>X, Y</td>
<td>0.003</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>X, Y</td>
<td>0.003</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>X, Y</td>
<td>0.003</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.004</td>
<td>0.005, 0.01, 0.02</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.004</td>
<td>0.005, 0.01, 0.02</td>
</tr>
</tbody>
</table>

The same is done completely analogous for the order of convergence of time for the first and second order implicit schemes with the time steps 0.005, 0.01 and 0.02 seconds. $\phi$ is again chosen to be the average of the coefficient of moment. The orders of the variables other than the time can be found at the bottom of table 3.5. The tip speed ratio in all cases is equal to 2.5. All the results are compared qualitatively and quantitatively.

All 2D simulations are run on the 8 threads of a machine with an i7 6700 @ 3.67 GHz, 32 GB of DDR4 memory and an SSD storage drive. The 3D simulation is run on the hpc11 cluster of the Delft University of Technology. The elapsed time $et$ is divided by $t_0 = 3600$ s to get a dimensionless 'time cost' — lower is better — relative to an hour of calculation time. All calculated values of $C_m$ and plots are multiplied with -1, to reflect the values for positive tip speed ratio’s (the values become negative because of the right hand rule in combination with the chosen direction of rotation). This is allowed, since the clockwise case has complete symmetry with the counter-clockwise case. Scripts and plots are made with MATLAB 2015b [11] and ANSYS CFD-Post [1], where the scripts can be found in Appendix B.
4 Results and Discussion

4.1 Verification of the Models against the Experimental Data

The results of the verification simulations are given in figure 4.1.

![Graph showing verification results](image)

*Figure 4.1: Results of the verification of the model. The experimental data is displayed in black with the grey area being the confidence interval. The $k$-$\varepsilon$ model is plotted in red, the $k$-$\omega$ SST model in blue and the transition SST model in green.*

In the plot some interesting results can be seen. Surprisingly the $k$-$\varepsilon$ model seems to be the model that best follows the experimental data, despite being generally considered as the lesser model for turbine modelling, because of its lack of accuracy near walls. [5][4][21] Furthermore, we see that the transition SST model without calibration does nothing to improve accuracy, it even produces results worse than the $k$-$\omega$ SST model it is derived from. It has been shown in the past that with calibration the transition SST model can produce very accurate results in these types of flows, but the calibration process is very tedious and requires more extensive measurement data from the turbine. [16]

Another apparent tendency in this graph is the overshoot of the estimated power coefficients at higher tip speed ratios of the models. This is a trend that can be spotted in similar numerical vs experimental research in the vertical axis wind turbine area. [16][7][14]. This overshoot could have several causes, the two most important being that the models describe a 2D turbine that does not suffer from the negative side effects from 3D turbines, such as more parasitic geometry (e.g. struts) and tip vortices, and the increased influence of the mechanical energy losses in the turbine (e.g. friction). As the turbine is spinning at a fixed speed, the wind velocity gets lower as the tip speed ratio gets higher. This can also be seen in table 3.4. The total amount of power available in the wind gets
quadratically lower when the wind velocity is reduced, meaning that the mechanical power losses in the turbine will play a more significant role in the determination of the power coefficient. The used experimental data does not account for this effect. [10] A comparison between a 3D simulation and a 2D simulation has shown that the 3D effects, such as extra geometry and tip vortices, play a significant role in the simulated power coefficient. Howell et. al [7] has performed research on this matter, and their paper shows a similar overshoot at high tip speed ratio’s in 2D, but a better resemblance with the measurements in 3D, including the sudden dip at high tip speed ratio’s. For better results, the simulations should therefore be repeated in 3D, when the appropriate amount of computation power has become more mainstream.

Something else to consider is the accuracy of the geometry provided in the simulation. While the dimensions of the turbine were taken directly from the specifications, small differences with the actual model could very well occur, e.g. by small practical imperfections introduced in the building stages of the turbine. In the conclusion of Kjellin et. al [10] it is stated that with small improvements to the blades the power coefficient in similar research could be brought up from 0.29 to about 0.39, a gigantic 35% difference with the outcome of the original experiment. This means that for obtaining more reliable results, both the numerical and experimental part could be performed in future research, thus making it so that the researcher has the perfect one on one dimensions for the turbine.

4.2 Determining the ‘optimal’ Solution Method

4.2.1 ‘Optimal’ Spatial Discretization

The results of the determination of the ‘optimal’ solution method for group X and Y are displayed in tables 4.1 and 4.2 respectively.

Table 4.1: Results of the determination of the order of different combinations of discretization schemes for group X. The first six columns are an exact copy of the contents of table 3.5, the seventh column (BES) indicates the blade edge sizing, the eight column gives the average coefficient of moment of the last revolution in the simulation, the ninth column gives the time cost (relative to \( t_0 = 3600 \) s) and the last column gives the apparent order.

<table>
<thead>
<tr>
<th>Sim. #</th>
<th>p</th>
<th>M</th>
<th>k</th>
<th>( \omega )</th>
<th>T</th>
<th>BES (m)</th>
<th>( C_{m_t} )</th>
<th>( et/t_0 )</th>
<th>( g )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.005</td>
<td>-0.0230</td>
<td>1.03</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.005</td>
<td>-0.0134</td>
<td>1.00</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.01</td>
<td>-0.0316</td>
<td>0.98</td>
<td>-0.7290</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0.01</td>
<td>-0.0223</td>
<td>0.92</td>
<td>-0.7383</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.01</td>
<td>-0.0254</td>
<td>0.93</td>
<td>-0.7980</td>
</tr>
</tbody>
</table>

Both tables return some rather unexpected and unrealistic results. Most of the obtained orders are negative and nowhere close to the expected range of (1, 2). A calculation error might be expected, but
Table 4.2: Results of the determination of the order of different combinations of discretization schemes for group Y. The first six columns are an exact copy of the contents of table 3.5, the seventh column (BES) indicates the blade edge sizing, the eight column gives the average coefficient of moment of the last revolution in the simulation, the ninth column gives the time cost (relative to $t_0 = 3600$ s) and the last column gives the apparent order.

<table>
<thead>
<tr>
<th>Sim. #</th>
<th>p</th>
<th>M</th>
<th>k</th>
<th>$\omega$</th>
<th>T</th>
<th>BES (m)</th>
<th>$C_m$</th>
<th>$et/t_0$</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.0004</td>
<td>0.0733</td>
<td>3.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0012</td>
<td>0.0354</td>
<td>1.62</td>
<td>-0.2992</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.0004</td>
<td>0.0746</td>
<td>3.58</td>
<td>-0.3463</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0012</td>
<td>0.0373</td>
<td>2.77</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0.0004</td>
<td>0.0531</td>
<td>3.37</td>
<td>-2.5177</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0012</td>
<td>0.0538</td>
<td>2.77</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>0.0004</td>
<td>0.0735</td>
<td>5.24</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0012</td>
<td>0.0703</td>
<td>2.47</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.0004</td>
<td>0.0641</td>
<td>5.30</td>
<td>-1.3895</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0012</td>
<td>0.0686</td>
<td>2.75</td>
<td></td>
</tr>
</tbody>
</table>

This is not the case. Looking at the actual measurements it indeed appears that the delta between the smallest and second smallest measurement is greater than the delta between the second smallest and largest measurement, something that is not expected for converging simulations at all. The results in group Y can be partly explained in the following way: looking at simulations 3-5, the values of $C_m$ are close to each other, indicating a converged solution. The same is true for parts of the first two plots, as apparently the required resolution for a converged solution differs with the rotation $\theta$ of the turbine. One of the conditions for formula 3.1 to work is that it is used on simulations where the discretization scheme does not yet lead to a converged solution. This still does not explain why the apparent order of the other simulations behaves in such a strange way. To verify that the data itself does actually make sense, the simulations of groups X and Y are plotted in figures 4.2 and 4.3.

The figures reveal some extra information about the simulations, which can be processed qualitatively. Bringing up figure 4.2 first, it becomes clear that the difference in quality of the solution between the different schemes is significant. Where the 11112 and 21112 schemes do not come close to the actual solution — the converged solution of the 22212 and 22222 scheme of figure 4.3 — the simulations where the pressure and momentum order is at 2 show promising results for the most refined mesh. This also shows in the time elapsed for the simulations, since setting the momentum order to 2 induces a performance hit of about 40%. This still does not give an explanation for the strange behavior of the convergence. We can talk about convergence, because in figure 4.3 the solution does converge if we keep refining the mesh. One explanation could be that the variable chosen — the coefficient of moment $C_m$ — is not a suitable variable to base the determination on. If the coefficient of moment would for example gain a significant boost in accuracy for a certain resolution, while remaining mostly insensitive for the resolutions before that point, this could manifest itself as effects like in this research in the simulations. This essentially means that the courser mesh might not be the limiting factor for the coefficient of moment in the plots. This could be looked in to in a follow-up research that takes other variables into account as well. Another explanation for the odd behavior of the orders lies somewhat deeper under the surface. ANSYS FLUENT, being a commercial software package, is also...
Figure 4.2: Plots accompanying the simulations from table 4.1. The five number sequences above the plots correspond to the orders of the variables listed in the table. Clear is that the simulation data shows the appropriate physical behavior (i.e. the variation in $C_m$ three times every revolution), but not the desired mathematical behavior. The solutions seem to take a sudden jump when going from blade edge sizing 0.01 m to 0.005 m.

used by people that do not have as much of a physical background and by people that have less time. To make sure that solutions converge in a wide variety of cases — even when they normally wouldn’t — some optimizations might be done behind the scenes in order to ensure this. In the second group (Y) the results are too close to give an accurate estimate of the order of convergence. In a successive
research simulations with other variables could be done to solve this problem, as well as simulations with other mesh intervals and tip speed ratio's.

Returning to what we can conclude from these simulations, it generally seems to be a good idea
to run simulations with a second order discretization scheme for the pressure $p$, the momenta and the turbulent kinetic energy $k$. Whether $\omega$ should be run to second order accuracy is up to the reader.

The last two plots in figures 4.2 and 4.3 show no significant differences, but tables 4.1 and 4.2 also show no significant gain in time cost. Adding the turbulent kinetic energy $k$ to the variables calculated to second order accuracy however, does increase the accuracy of the simulation — as both cases show a more converged result when this is done — while not having a significant impact on the time cost of the simulation. This is clear from the third and fourth sub-figure of figure 4.3, where the first still shows small but visible differences, while the second shows minimal inequalities.

### 4.2.2 'Optimal' Temporal Discretization

The results of the determination of the ‘optimal’ temporal solution method are displayed in table 4.3.

<table>
<thead>
<tr>
<th>Group</th>
<th>Time</th>
<th>Sim. #</th>
<th>$p$</th>
<th>M</th>
<th>$k$</th>
<th>$\omega$</th>
<th>T</th>
<th>Time step (s)</th>
<th>$C_m$</th>
<th>$ct/t_0$</th>
<th>$q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td>0.005</td>
<td>0.0444</td>
<td>3.82</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.01</td>
<td>0.0401</td>
<td>2.42</td>
<td>2.1867</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.02</td>
<td>0.0262</td>
<td>1.12</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td>0.005</td>
<td>0.0457</td>
<td>3.92</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.01</td>
<td>0.0404</td>
<td>2.03</td>
<td>2.1701</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.02</td>
<td>0.0224</td>
<td>1.05*</td>
<td></td>
</tr>
</tbody>
</table>

Again the raw results do not tell the whole story. Therefore a plot with the results is included in figure 4.4. The plot clearly shows that the second order scheme converges faster to its final value and that the smallest time step shows more detail than the same time step in the first order scheme. Furthermore, no significant time cost is added when the better scheme is used, so the conclusion is really a no-brainer: The second order temporal discretization scheme should be used at all times.

### 4.3 Physical Verification of the Results

First off, the results in the previous section deserve another look. In all plots the rapidly varying behavior of the turbine can be spotted. As mentioned before, dynamic stall due to the quickly changing angle of attack plays a large role in the disadvantages of VAWT’s in general. Clear in these plots is that the coefficient of moment changes fast and even becomes negative for short periods of time, which hurts the overall performance and causes the rapidly varying forces on the blade, accelerating wear.

In order to further verify that the obtained results are indeed physical, some plots are included of a converged flow. The specific simulation is simulation 4 from table 3.5. The flow in the figures always goes from left to right — they are oriented in the same way as the flow in figure 3.5 — and the kind of plot is indicated in the top left of each figure. First off, a plot of the pressure contours is given in figure 4.5, then a plot of the velocity contours is given in figure 4.6 followed by a plot of a velocity vector field and some streamlines in figures 4.7 and 4.8 and lastly a plot of the turbulence kinetic energy $k$ is included in figure 4.9.
Figure 4.4: Plot of the results of the determination of the temporal order of the simulations. Note that these simulations were run back to back to reduce influence of initial conditions and ensure faster convergence.

4.4 Results of the 3D simulation

On top of the results in the previous subsection, some figures of the 3D simulation are included. In figure 4.10 some streamlines around the turbine are plotted. Note that the simulation was the 3D symmetry case, but that the CFD-post software recreates the turbine from its symmetry plane. Lastly, a figure is included that nicely shows the influence of the parasitic geometry on the flow around the turbine. A small vortex can be seen behind the strut connecting the hub to the blade in figure 4.11.
Figure 4.5: Plot of the pressure contours in the converged flow of a rotating 2D vertical axis wind turbine. The areas around the blades have been enhanced for a better look at what happens at those locations. The bottom blade shows a familiar blade pressure profile, with pressure build-up at the tip of the blade and a sudden drop in pressure along the sides. This perfectly ties in with the velocity profile. The other blades show similar behavior, both deferring slightly from the bottom blade because of the different angle of attack.
Figure 4.6: Plot of the velocity contours in the same situation as figure 4.5. The similarities between the figures are readily seen, as the velocity is generally high where the pressure is low and the other way around. This is shown perfectly for the top left blade, where the area of high velocity is quite large, just like the large pressure area for the same blade in figure 4.5. Furthermore, the wake of the turbine can be clearly seen in this figure, including the slowly fading turbulence right behind the central hub.
Figure 4.7: Plot of the velocity vector field in the same situation as figures 4.5 and 4.6. This vector field shows the flow around the bottom blade in both figures. The plot is essentially the same as the velocity contour plot, but now giving direction to the velocity vector. The vector field ties in beautifully with the explanation given in the other figures: The air velocity is high where the pressure is low. Furthermore, it can be seen that the layer of air sticks to the blade and is transported to the trailing edge of the blade very fast.
Figure 4.8: Plot of the same situation as figure 4.7, to illustrate the airflow some more. It can be seen that streams coming from far away do not generally get very close to the blade, they are transported around the blade instead.
Figure 4.9: Plot of the turbulent kinetic energy $k$ in the same situation as figures 4.5 and 4.6. This plot essentially shows where the areas of heavy turbulence are. The blade on the right has a massive bubble of turbulence behind it, while the other blades are in a more favorable air stream. A nice unsteady trail can be spotted behind the central hub, indicating once more the unsteady nature of turbulence.
Figure 4.10: Some streamlines of the flow around the 3D model of the wind turbine. The 3D effect of the struts on the flow can be clearly observed, since the slight tilt of the struts deflects the air at a significant angle.
Figure 4.11: Plot of the velocity vectors in the standard frame in a plane right behind one of the blades. A small vortex can be seen right behind the strut and slightly above it.
5 Conclusion

After the results and discussion in the previous chapter, we can now answer the research question: “What models are available for the simulation of vertical axis wind turbines and what considerations should be taken into account to most effectively obtain the power coefficient of those turbines?” Although the verification of the experimental results indicated that the $k-\varepsilon$ model simulation follow the experimental line the closest, it would be ill-advised to recommend this model for further simulation of wind turbines, because we know that this model has trouble producing accurate results at walls. For general purposes, the $k-\omega$ SST model should give the best results, in contrast to the uncalibrated transition SST model, which offers no increase in accuracy at all and even takes longer to compute. Furthermore, it can be concluded that for better results, a full 3D simulation should be conducted, when the required computation power is more generally available.

As to what considerations should be taken into account: Meshing determines largely how long a simulation will take and to what accuracy a result can be calculated. Meshes should be more refined near blades — as the $k-\omega$ model requires resolution near the wall — and inflation layers should be added to further increase this refinement. The sliding mesh method should be used to decrease calculation time, so that the mesh does not need to be recalculated every time step. On to optimizations, the orders of the different variables should be set to second order accuracy for the pressure $p$, the momentum and the turbulent kinetic energy $k$. The second order calculation of these variables results in a significant increase in accuracy of the simulations. Whether the specific dissipation $\omega$ should be calculated to second order accuracy in up to the reader, since it does not seem to have a significant impact on both the time cost and the accuracy of a simulation. Lastly, the time derivative should always be calculated to second order accuracy, since it does not impact the time cost of a simulation at all, but does increase the accuracy of the result.

To get better results more simulations in 3D should be done. Furthermore, if a specific turbine was to be simulated, it might pay off to calibrate the transition SST model to obtain the best results. This does require more data, for which research in a wind tunnel might be required. Continuing on this point, being able to both control the experimental results and the simulations would be a more ideal situation, since the 3D model of the turbine could be modelled exactly after the practical limitations of building a real turbine.
References


Appendix A: ANSYS Journal Code Example

In this appendix the source code of an order determination data set simulation is shown. This specific code runs simulation 4 from table 3.5.

```plaintext
1 # encoding: utf-8
2 SetScriptVersion(Version="15.0")
3 system1 = GetSystem(Name="FFF 3")
4 fluentLauncherSettings1 = setup1.GetFluentLauncherSettings()
5 fluentLauncherSettings1.SetEntityProperties(Properties=Set(EnvPath={}))
6 setup1.Edit()
7 setup1.SendCommand(Command='(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton12("Reference Values")")')
8 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Reference Values*DropDownList1(Compute from)" (8))(cx-gui-do cx-activate-item "Reference Values*DropDownList1(Compute from)")")
9 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Pressure)" (0))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Pressure)")")
10 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList3(Momentum)" (1))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList3(Momentum)")")
11 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)" (1))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)")")
12 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList5(Specific Dissipation Rate)" (0))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList5(Specific Dissipation Rate)")")
13 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Transient Formulation)" (1))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Transient Formulation)")")
14 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)" (1))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)")")
15 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList5(Specific Dissipation Rate)" (0))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList5(Specific Dissipation Rate)")")
16 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Transient Formulation)" (1))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Transient Formulation)")")
17 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)" (1))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)")")
18 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList5(Specific Dissipation Rate)" (0))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList5(Specific Dissipation Rate)")")
19 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Transient Formulation)" (1))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Transient Formulation)")")
20 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)" (1))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)")")
21 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList5(Specific Dissipation Rate)" (0))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList5(Specific Dissipation Rate)")")
22 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Transient Formulation)" (1))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Transient Formulation)")")
23 setup1.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)" (1))(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)")")
```
item "Monitors*Frame1*Table1*List1(Residuals, Statistic and Force Monitors)"
)
50 setup2.SendCommand(Command="(cx-gui-do cx-set-list-selections "Monitors*Frame1*Table1* List1(Residuals, Statistic and Force Monitors)"
')( 3))''(cx-gui-do cx-activate-item "Monitors*Frame1*Table1*List1(Residuals, Statistic and Force Monitors)"
))
51 setup2.SendCommand(Command="(cx-gui-do cx-activate-item "Monitors*Frame1*Table1*Frame2* Table2*PushButton2(Edit)"
))
52 setup2.SendCommand(Command="(cx-gui-do cx-set-text-entry "Force Moment Monitor*Frame1*Table1*Frame1_Options*Table1_options*Frame2*Table2*TextBox1(Force Moment Monitor*Frame1*PanelButtons*PushButton1(OK))"
))
53 setup2.SendCommand(Command="(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton17(Solution Initialization)"
))
54 setup2.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Initialization*Frame1*Table1*DropDownList1(Compute from)"
' '( 1))''(cx-gui-do cx-activate-item "Solution Initialization*Frame1*Table1*DropDownList1(Compute from)"
))
55 setup2.SendCommand(Command="(cx-gui-do cx-activate-item "Solution Initialization*Frame1* Table1*ButtonBox8*PushButton(Initialize)"
))
56 setup2.SendCommand(Command="(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton19( Run Calculation)"
))
57 setup2.SendCommand(Command="(cx-gui-do cx-set-integer-entry "Run Calculation*Frame1*Table6*Table6*IntegerEntry2(Number of Time Steps)" 1250)''(cx-gui-do cx-activate-item "Run Calculation*Frame1*Table6*Table6*IntegerEntry2(Number of Time Steps)"
))
58 setup2.SendCommand(Command="(cx-gui-do cx-activate-item "Run Calculation*Frame1*Table6*Table6*IntegerEntry2(Number of Time Steps)"
))
59 setup2.SendCommand(Command="(cx-gui-do cx-activate-item "Information*OK")''(cx-gui-do cx-activate-item "Warning*OK")'
))
60 setup2.SendCommand(Command="(cx-gui-do cx-activate-item "MenuBar*FileMenu*Close Fluent")'
))
61 system3 = GetSystem(Name="FFF 1")
62 setup3 = system3.GetContainer(ComponentName="Setup")
63 fluentLauncherSettings3 = setup3.GetFluentLauncherSettings()
64 fluentLauncherSettings3.SetEntityProperties(Properties=Set(EnvPath=
65 setup3.Edit()
66 setup3.SendCommand(Command="(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton12( Reference Values)"
))
67 setup3.SendCommand(Command="(cx-gui-do cx-set-list-selections "Reference Values*Frame1*DropDownList1(Compute from)"
' '( 8))''(cx-gui-do cx-activate-item "Reference Values*Frame1*DropDownList1(Compute from)"
))
68 setup3.SendCommand(Command="(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton14( Solution Methods)"
))
69 setup3.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList2(Pressure)"
'( 0))''(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)* Table3(Spatial Discretization)*DropDownList2(Pressure)"
))
70 setup3.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList3(Momentum)"
'( 1))''(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)* Table3(Spatial Discretization)*DropDownList3(Momentum)"
))
71 setup3.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList4(Turbulent Kinetic Energy)"
'( 3))''(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)* DropDownList4(Turbulent Kinetic Energy)"
))
72 setup3.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList5(Specific Dissipation Rate)"
'( 0))''(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)* DropDownList5(Specific Dissipation Rate)"
))
73 setup3.SendCommand(Command="(cx-gui-do cx-set-list-selections "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)*Table3(Spatial Discretization)*DropDownList1(Transcient Formulation)"
'( 3))''(cx-gui-do cx-activate-item "Solution Methods*Frame1*Table1*Frame3(Spatial Discretization)* DropDownList1(Transcient Formulation)"
))
74 setup3.SendCommand(Command="(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton16( Monitors)"
))
75 setup3.SendCommand(Command="(cx-gui-do cx-set-list-selections "Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force Monitors)"
'( 2))''(cx-gui-do cx-activate-
item "Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force Monitors)"")
76 setup3.SendCommand(Command='(cx-gui-do cx-activate-item "Monitors*Frame1*Table1*Frame2* Table2*PushButton2(Edit)")')
77 setup3.SendCommand(Command='(cx-gui-do cx-set-text-entry "Force Moment Monitor*Frame1*Table1 +Frame1*Table1*Frame1(Options)+Frame1(Options)+Frame5Table5*TextEntry2(File Name)" "cm -1-new-4x-22212") (cx-gui-do cx-activate-item "Force Moment Monitor*PanelButtons*PushButton1(OK)")')
78 setup3.SendCommand(Command='(cx-gui-do cx-set-list-selections "Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force Monitors)" (3) (cx-gui-do cx-activate-item "Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force Monitors)")')
79 setup3.SendCommand(Command='(cx-gui-do cx-set-list-selections "Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force Monitors)" (3) (cx-gui-do cx-activate-item "Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force Monitors)")')
80 setup3.SendCommand(Command='(cx-gui-do cx-set-list-selections "Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force Monitors)" (3) (cx-gui-do cx-activate-item "Monitors*Frame1*Table1*Frame2* Table2*PushButton2(Edit)")')
81 setup3.SendCommand(Command='(cx-gui-do cx-set-list-selections "Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force Monitors)" (3) (cx-gui-do cx-activate-item "Monitors*Frame1*Table1*Frame1*List1(Residuals, Statistic and Force Monitors)")')
82 setup3.SendCommand(Command='(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton17{ Solution Initialization}")')
83 setup3.SendCommand(Command='(cx-gui-do cx-set-list-selections "Solution Initialization*Frame1*Table1*DropDownList1(Compute from)" (1) (cx-gui-do cx-activate-item "Solution Initialization*Frame1*Table1*DropDownList1(Compute from)")')
84 setup3.SendCommand(Command='(cx-gui-do cx-activate-item "Solution Initialization*Frame1*Table1*ButtonBox8*PushButton1(Initialize)")')
85 setup3.SendCommand(Command='(cx-gui-do cx-activate-item "NavigationPane*Frame1*PushButton19{ Run Calculation}")')
86 setup3.SendCommand(Command='(cx-gui-do cx-set-integer-entry "Run Calculation*Frame1*Table6*Table6*IntegerEntry2(Number of Time Steps)" 1250) (cx-gui-do cx-activate-item "Run Calculation*Frame1*Table1*Frame6*Table6*IntegerEntry2(Number of Time Steps)")')
87 setup3.SendCommand(Command='(cx-gui-do cx-activate-item "Run Calculation*Frame1*Table1*PushButton21(Calculate)")')
88 setup3.SendCommand(Command='(cx-gui-do cx-activate-item "Information*OK") (%cx-warning-dialog "OK to close Fluent?" #f) (cx-gui-do cx-activate-item "Warning*OK")')
89 setup3.SendCommand(Command='(cx-gui-do cx-activate-item "MenuBar*FileMenu*Close Fluent")')
Appendix B: Matlab Scripts

Included in this appendix are the Matlab scripts used in this research.

Optimization_group_X_Time.m:

```matlab
1 close all;
2 clear all;
3 clc;
4
5 % Load data
6 load('opti_data_new.mat');
7 % Group I spatial
8 h = [0.0004 0.000692820 0.0012];
9 r = 1.732050808;
10
12 m11112 = -[mean(cm11112x1(end-416:end)) mean(cm11112x2(end-416:end)) mean(cm11112x4(end-416 :end))];
13 m21112 = -[mean(cm21112x1(end-416:end)) mean(cm21112x2(end-416:end)) mean(cm21112x4(end-416 :end))];
14 m22112 = -[mean(cm22112x1(end-416:end)) mean(cm22112x2(end-416:end)) mean(cm22112x4(end-416 :end))];
15 m22212 = -[mean(cm22212x1(end-416:end)) mean(cm22212x2(end-416:end)) mean(cm22212x4(end-416 :end))];
16 m22222 = -[mean(cm22222x1(end-416:end)) mean(cm22222x2(end-416:end)) mean(cm22222x4(end-416 :end))];
17
18 eps21 = [m11112(2)-m11112(1); m21112(2)-m21112(1); m22112(2)-m22112(1); m22212(2)-m22212(1);
19 m22222(2)-m22222(1)];
20 eps32 = [m11112(3)-m11112(2); m21112(3)-m21112(2); m22112(3)-m22112(2); m22212(3)-m22212(2);
21 m22222(3)-m22222(2)];
22
23 p = 1/log(r)*log(eps32./eps21);
24
26 subplot(3,2,1)
27 hold on;
28 plot(FlowTime,-cm11112x1);
29 plot(FlowTime,-cm11112x2,'r');
30 plot(FlowTime,-cm11112x4,'g');
31 hold off;
32 title('Time against C_m 11112')
33 legend('Blade edge sizing (m): 0.0004', 'Blade edge sizing (m): 0.00069282', 'Blade edge sizing (m): 0.0012', 'Location', 'southeast');
34 xlabel('t (s)');
35 ylabel('C_m (-)');
36 subplot(3,2,2)
37 hold on;
38 plot(FlowTime,-cm21112x1);
39 plot(FlowTime,-cm21112x2,'r');
40 plot(FlowTime,-cm21112x4,'g');
41 hold off;
42 title('Time against C_m 21112')
43 legend('Blade edge sizing (m): 0.0004', 'Blade edge sizing (m): 0.00069282', 'Blade edge sizing (m): 0.0012', 'Location', 'southeast');
44 xlabel('t (s)');
45 ylabel('C_m (-)');
46 subplot(3,2,3)
47 hold on;
48 plot(FlowTime,-cm22112x1);
49 plot(FlowTime,-cm22112x2,'r');
50 ```
plot(FlowTime,-cm22112x4,'g'); hold off; title('Time against C_m 22112') legend('Blade edge sizing (m): 0.0004', 'Blade edge sizing (m): 0.00069282', 'Blade edge sizing (m): 0.0012', 'Location', 'southeast'); xlabel('t (s)'); ylabel('C_m (-)'); subplot(3,2,4) hold on; plot(FlowTime,-cm22212x1); plot(FlowTime,-cm22212x2,'r'); plot(FlowTime,-cm22212x4,'g'); hold off; title('Time against C_m 22212') legend('Blade edge sizing (m): 0.0004', 'Blade edge sizing (m): 0.00069282', 'Blade edge sizing (m): 0.0012', 'Location', 'southeast'); xlabel('t (s)'); ylabel('C_m (-)'); subplot(3,2,5) hold on; plot(FlowTime,-cm22222x1); plot(FlowTime,-cm22222x2,'r'); plot(FlowTime,-cm22222x4,'g'); hold off; title('Time against C_m 22222') legend('Blade edge sizing (m): 0.0004', 'Blade edge sizing (m): 0.00069282', 'Blade edge sizing (m): 0.0012', 'Location', 'southeast'); xlabel('t (s)'); ylabel('C_m (-)'); % Temporal t = [0.02:0.02:8 8.01:0.01:12 12.005:0.005:16 16.02:0.02:20 20.01:0.01:24 24.005:0.005:28]; figure(); hold on; plot(t, -[cmt2ts002; cmt2ts001; cmt1ts002; cmt1ts001; cmt1ts0005]'); line([8 8], get(gca, 'ylim'), 'Color', 'r', 'LineStyle', '--'); line([12 12], get(gca, 'ylim'), 'Color', 'r', 'LineStyle', '--'); line([16 16], get(gca, 'ylim'), 'Color', 'r', 'LineStyle', '--'); line([20 20], get(gca, 'ylim'), 'Color', 'r', 'LineStyle', '--'); line([24 24], get(gca, 'ylim'), 'Color', 'r', 'LineStyle', '--'); hold off; axis([0 28 get(gca, 'ylim')]); xlabel('t (s)'); ylabel('C_m (-)'); rt = 2; m2t = -[mean(cmt2ts0005(end-500:end)) mean(cmt2ts001(end-250:end)) mean(cmt2ts002(end-125 :end))]; m1t = -[mean(cmt1ts0005(end-500:end)) mean(cmt1ts001(end-250:end)) mean(cmt1ts002(end-125 :end))]; eps21t = [m2t(2)-m2t(1); m1t(2)-m1t(1)]; eps32t = [m2t(3)-m2t(2); m1t(3)-m1t(2)]; pt = 1/log(r)*log(eps32t./eps21t); Optimization_group_Y.m: close all; clear all;
clc;
load('opti_data_new_v2.mat');

h = [0.005 0.01 0.02];
r = 2;

m11112 = -[mean(cm11112x025(end-416:end)) mean(cm11112x050(end-416:end)) mean(cm11112x100(end-416:end))];
m21112 = -[mean(cm21112x025(end-416:end)) mean(cm21112x050(end-416:end)) mean(cm21112x100(end-416:end))];
m22112 = -[mean(cm22112x025(end-416:end)) mean(cm22112x050(end-416:end)) mean(cm22112x100(end-416:end))];
m22212 = -[mean(cm22212x025(end-416:end)) mean(cm22212x050(end-416:end)) mean(cm22212x100(end-416:end))];
m22222 = -[mean(cm22222x025(end-416:end)) mean(cm22222x050(end-416:end)) mean(cm22222x100(end-416:end))];

eps21 = [m11112(2)-m11112(1); m21112(2)-m21112(1); m22112(2)-m22112(1); m22212(2)-m22212(1); m22222(2)-m22222(1)];
eps32 = [m11112(3)-m11112(2); m21112(3)-m21112(2); m22112(3)-m22112(2); m22212(3)-m22212(2); m22222(3)-m22222(2)];

p = 1/log(r) * log(eps32./eps21);

subplot(3,2,1)
hold on;
plot(FlowTime,-cm11112x025);
plot(FlowTime,-cm11112x050,'r');
plot(FlowTime,-cm11112x100,'g');
hold off;
title('Time against C_m 11112')
legend('Blade edge sizing (m): 0.005', 'Blade edge sizing (m): 0.01', 'Blade edge sizing (m): 0.02', 'Location', 'southeast');
xlabel('t (s)');
ylabel('C_m (-)');

subplot(3,2,2)
hold on;
plot(FlowTime,-cm21112x025);
plot(FlowTime,-cm21112x050,'r');
plot(FlowTime,-cm21112x100,'g');
hold off;
title('Time against C_m 21112')
legend('Blade edge sizing (m): 0.005', 'Blade edge sizing (m): 0.01', 'Blade edge sizing (m): 0.02', 'Location', 'southeast');
xlabel('t (s)');
ylabel('C_m (-)');

subplot(3,2,3)
hold on;
plot(FlowTime,-cm22112x025);
plot(FlowTime,-cm22112x050,'r');
plot(FlowTime,-cm22112x100,'g');
hold off;
title('Time against C_m 22112')
legend('Blade edge sizing (m): 0.005', 'Blade edge sizing (m): 0.01', 'Blade edge sizing (m): 0.02', 'Location', 'southeast');
xlabel('t (s)');
ylabel('C_m (-)');
plot(FlowTime,-cm22212x100,'g');
hold off;
title('Time against C_m 22212')
legend('Blade edge sizing (m): 0.005', 'Blade edge sizing (m): 0.01', 'Blade edge sizing (m) : 0.02', 'Location', 'southeast');
xlabel('t (s)');
ylabel('C_m (-)');

subplot(3,2,5)
hold on;
plot(FlowTime,-cm22222x025);
plot(FlowTime,-cm22222x050,'r');
plot(FlowTime,-cm22222x100,'g');
hold off;
title('Time against C_m 22222')
legend('Blade edge sizing (m): 0.005', 'Blade edge sizing (m): 0.01', 'Blade edge sizing (m) : 0.02', 'Location', 'southeast');
xlabel('t (s)');
ylabel('C_m (-)');