NTNU

Norwegian University of
Science and Technology

Wind Turbine Simulations with OpenFOAM

Maria Enger Hoem

Master of Science in Mechanical Engineering
Submission date: January 2018
Supervisor: Reidar Kristoffersen, EPT

Norwegian University of Science and Technology
Department of Energy and Process Engineering

NTNU

Norwegian University
of Science and Technology

Department of Energy
and Process Engineering

EPT-M-2017-36

MASTER THESIS

for

Maria Enger Hoem

Autumn 2017

Wind Turbine Simulations with OpenFOAM

Vindturbinsimuleringer med OpenFOAM

Background and objective

Detailed numerical simulations of wind turbines are very costly, involving 3D turbulent transient flow around rotating turbine blades. In wind park design, knowledge of the wake behind turbines are important, so accurate and simple models of the complex flow are of great interest. The aim of this thesis is to validate actuator disk models with more accurate simulations.

The following tasks are to be considered:

1. Simulations of different wind turbine scenarios using actuator disk model.
2. Case study of moving mesh systems in OpenFOAM.
3. 3D transient simulations of different wind turbine scenarios.
4. Verification and validation of the simulations.

-- " --

Within 14 days of receiving the written text on the master thesis, the candidate shall submit a research plan for his project to the department.

When the thesis is evaluated, emphasis is put on processing of the results, and that they are presented in tabular and/or graphic form in a clear manner, and that they are analyzed carefully.

The thesis should be formulated as a research report with summary both in English and Norwegian, conclusion, literature references, table of contents etc. During the preparation of the text, the candidate should make an effort to produce a well-structured and easily readable report. In order to ease the evaluation of the thesis, it is important that the cross-references are correct. In the making of the report, strong emphasis should be placed on both a thorough discussion of the results and an orderly presentation.

The candidate is requested to initiate and keep close contact with his/her academic supervisor(s) throughout the working period. The candidate must follow the rules and regulations of NTNU as well as passive directions given by the Department of Energy and Process Engineering.

Risk assessment of the candidate's work shall be carried out according to the department's procedures. The risk assessment must be documented and included as part of the final report. Events related to the candidate's work adversely affecting the health, safety or security, must be documented and included as part of the final report. If the documentation on risk assessment represents a large number of pages, the full version is to be submitted electronically to the supervisor and an excerpt is included in the report.

Pursuant to "Regulations concerning the supplementary provisions to the technology study program/Master of Science" at NTNU §20, the Department reserves the permission to utilize all the results and data for teaching and research purposes as well as in future publications.

The final report is to be submitted digitally in DAIM. An executive summary of the thesis including title, student's name, supervisor's name, year, department name, and NTNU's logo and name, shall be submitted to the department as a separate pdf file. Based on an agreement with the supervisor, the final report and other material and documents may be given to the supervisor in digital format.

☐ Work to be done in lab (Water power lab, Fluids engineering lab, Thermal engineering lab)
☐ Field work
☐ The project requires resources/time from technicians

Department of Energy and Process Engineering, 1. September 2017

Reidar Kristoffersen

Reidar Kristoffersen
Academic Supervisor

PREFACE

This report is written in such detail that others may follow it to do the same implementation of an actuator disk model with hub, and create a fully resolved wind turbine model using AMI like I have done, hopefully without all the struggle.

I would like to extend my gratitude to those who have helped me on my way to produce my master thesis. First and foremost I would like to thank my supervisor at NTNU Reidar Kristoffersen for all the constructive discussions, motivational talks and helpful input. I would also like to thank the contributors at the CFD-online OpenFOAM forum that has helped me understand and edit code when I have been absolutely stuck. In particular Louis Gagnon: `http://louisgagnon.com`.

Happy Foaming!

SUMMARY

An actuator disk model is improved by including an impermeable hub and nacelle. This improves the wake flow a lot, but it turns out that the *actuatorDiskExplicitForce* model is mesh dependent. A mesh is made that provides close to the intended forces and these results are quite good for the wake when using k-ω SST turbulence model. The simulation is compared with real measurements of the simulated wind turbine in a wind tunnel performed at The Norwegian University of Science and Technology. There are no asymmetric effects on the wake since the turbine tower is not included in the model. The estimated power output is overestimated with an error of 19.59 % which is too high, but a very common result it turns out.

After conducting a moving mesh test study which concludes that an Arbitrary Mesh Interface (AMI) can be used without it causing unintended disturbances to the flow, a 3D, transient turbine model is created using AMI. This report gives a detailed description of how to recreate this model and what to be aware of when doing so. A mesh refinement study is performed to yield the needed refinement of the mesh for the aerodynamic forces on the turbine to be calculated correctly. Due to limited amount of time, the fine mesh is not used, but a coarser mesh is used to test that the model works. With the relatively coarse mesh the estimated power output has an error of 19.27 %. The wake is not as resolved as suggested by the mesh study, but the velocity field in the wake is quite well recreated. The turbulent kinetic energy field in the wake has too low values, but the shape is resembling the measured data. Again, the asymmetric effects in the wake is not reproduced since the turbine tower is not included in the model.

It is concluded that the actuator disk model may be used with care to get a good initial qualitative estimate of flow behavior after the turbine, while the advanced turbine model using AMI gives better estimations for the wake fields and has the potential to estimate turbine performance well. However, the advanced turbine model requires a high performance computer.

SAMMENDRAG

En aktuator-modell er forbedret ved å introdusere et ugjennomtrengelig nav. Dette forbedrer strømmningen i turbinvaken mye, men det viser seg at *actuatorDiskExplicitForce*-modellen er avhengig av meshet. Et mesh som gir nær tiltenkte krefter er lagd, og disse resultatene for vaken er ganske gode når man bruker k-ω SST turbulensmodell. Simuleringen er sammenlignet med virkelige målinger av den simulerte vindturbinen i en vindtunnel utført ved Norges teknisk-naturvitenskapelige universitet. Det er ingen asymmetriske effekter på vaken siden turbintårnet ikke er inkludert i modellen. Den estimerte effekten er overvurdert med en feil på 19.59%, som er for høy, men et meget vanlig resultat viser det seg.

Etter å ha gjennomført et teststudie på bevegelige mesh, som konkluderer med at et vilkårlig mesh grenesnitt (AMI) kan brukes uten at det forårsaker utilsiktede forstyrrelser i strømningen, skapes en 3D, tidsavhengig turbinemodell ved hjelp av AMI. Denne rapporten gir en detaljert beskrivelse av hvordan man lager denne modellen og hva man skal være oppmerksom på underveis. En studie er utført for å bestemme hvor fint oppløst meshet må være for å beregne de aerodynamiske kreftene som virker på turbinen riktig. På grunn av tidsbegrensning brukes ikke det fine meshet, men et grovere mesh brukes til å teste at modellen fungerer. Med det relativt grove meshet er effekten estimerte med en feil på 19.27%. Vaken er ikke like fint oppløst som foreslått av mesh-studiet, men hastighetsfeltet i vaken er bra gjenskapt. Det turbulente kinetiske energifeltet i vaken har for lave verdier, men formen ligner de målte dataene. Igjen er de asymmetriske effektene i vaken ikke gjengitt siden turbintårnet ikke er inkludert i modellen.

Det konkluderes med at aktuator-modellen kan med forsiktighet brukes til å få et bra kvalitativt førsteutkast av strømningen bak turbinen, mens den avanserte turbinemodellen som bruker AMI gir bedre estimater av vaken, og har potensialet til estimere turbinytelsen godt. Den avanserte turbinemodellen krever imidlertidig en superdatamaskin og mye tid.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ACRONYMS

AD	=	Actuator Disk
ADEF	=	Actuator Disk Explicit Force
AMI	=	Arbitrary Mesh Interface
BC	=	Boundary Conditions
BT4	=	Blind Test 4
CFD	=	Computational Fluid Dynamics
CoR	=	Center of Rotation
CV	=	Control Volume
FOAM	=	Field Operation And Manipulation
FVM	=	Finite Volume Method
hpc	=	High performance computer
IC	=	Initial Conditions
LES	=	Large Eddy Simulation
MATLAB	=	Matrix Laboratory
MRF	=	Multiple Reference Frame
OF	=	OpenFOAM
RANS	=	Reynolds Averaged Navier-Stokes
SST	=	Shear Stress Transport

GREEK LETTERS

δ	=	Distance between cell centers
$\delta_{i,j}$	=	Kronecker delta
ϵ	=	Dissipating energy in eddies
Γ	=	Diffusion coefficient
γ	=	Diffusion coefficient for dynamic mesh
μ	=	Dynamic viscosity
μ_t	=	Turbulent dynamic viscosity
ν	=	Kinematic viscosity
ν_T	=	Eddy viscosity
ω	=	Turbulent frequency
ϕ	=	Generic flux
ρ	=	Density
σ_k	=	k-ϵ model constant
σ_ω	=	k-ω model constant
τ_{ij}	=	Shear rate tensor
$\forall_{AD}$	=	Volume of actuator disk

LATIN LETTERS

A	=	Area
A_d	=	Area of actuator disk
A_θ, A_x	=	Tangential and axial volume force constant
a	=	Axial flow induction factor
a_E, a_N, a_S, a_W	=	East, north, south and west coefficient
a_P	=	Central coefficient
C_P	=	Power coefficient
C_T	=	Thrust coefficient
$C_{1\epsilon}$, $C_{2\epsilon}$, C_μ	=	k-ϵ model constants
D_e, D_n, D_s, D_w	=	Diffusion conductance on east, north, south and west face
dC_P	=	Derivative of power coefficient
F_e, F_n, F_s, F_w	=	Convection flux on eastern, northern, south and western face
$\mathbf{f}$	=	Body forces
$\mathbf{f}(u)$	=	Flux vector function
$\mathbf{f}_{b\theta}$, $\mathbf{f}_{bx}$	=	Volume force in tangential and axial direction
k	=	Turbulent kinetic energy
l	=	Turbulent length scale
$\dot{m}_{st}$	=	Mass flow rate through the stream tube
$\mathbf{n}$	=	Outer normal vector
P	=	Production term
p	=	Pressure
p_d^+, p_d^-	=	Pressure upstream and downstream of the actuator disk
Q	=	Torque
$\mathbf{R}$	=	Diffusion and dissipation stresses
Re	=	Reynolds number
R_H	=	Interior radius of Actuator Disk
R_P	=	Exterior radius of Actuator Disk
r^*	=	Normalized radius of Actuator Disk
r_h'	=	Radius ratio for an Actuator Disk
S_p	=	Non-uniform part of the source term
S_u	=	Uniform part of source term
T	=	Thrust
T_k	=	Kolmogorov time scale for small eddies
T_t	=	Time scale in k-ϵ model
U	=	Velocity
$\vec{U}$	=	Free stream velocity vector
U_∞	=	Free stream velocity
U_d	=	Stream-wise velocity at actuator disk
U_W	=	Velocity field in the wake
u^+	=	Dimensionless velocity
y^+	=	Dimensionless length

OPERATORS

∇	=	Nabla operator

CHAPTER 1 INTRODUCTION

Today there exists commercial wind power in more than 100 countries with an total installed capacity of over 450 GW. The wind power industry keeps on growing every year and now serves about 4.7 % of the worlds electricity use (WWEA, 2016).

Commercial wind turbines usually have a diameter between 40 and 80 m where the power output increases with the diameter (d'Emil et al., 2003). Wind turbines are huge and heavy, which makes them complex to build. There are great investment cost associated with wind farms, making knowledge about the turbines performance essential before building. Computational fluid dynamic (CFD) tools can be used to gain such information.

An actuator disk can be used to make a simple model of a wind turbine. The actuator disk is a generic device where attributes of a turbine can be added to the flow. Instead of solving equations close to the complex geometry of the turbine blades an permeable disk is used. How this can be done will be explained later in section 2.3. For smaller turbines the simple models may provide good results, however, as the turbines grow so does the aeroelastic effects and more advanced models should be considered (Hansen et al., 2006).

Such advanced models may use the actual geometry of the turbine to simulate the effect of the turbine on the surrounding flow. These models are a lot more complex and computationally demanding than the actuator disk model, but has the potential to give much more detailed and specific information of the turbine at hand.

There have been organized four blind test cases by NOWITECH and NORCOWE to see how different experts solve the same CFD problem, and to see how their predictions on total power output compare with real measurements. The fourth blind test case was held in 2015 and aimed to predict wake flow interaction between two in-line model wind turbines in a wind tunnel. The five contributors had 6 months to predict the results that would be discussed in a two day workshop in Trondheim. They all predicted the performance of the upstream turbine quite well, however the results for the downstream turbine was scattered.

Thus, it was concluded that there may be good predictions from such CFD models, but it is still very important to be critical to the results and to validate them (Bartl and Sœtran, 2017). The blind tests give an impression of how hard it is to use CFD tools correctly. The results are always wrong, but it varies how wrong they are and how accurate the predictions have to be to be sufficiently useful.

1.1 Previously made wind turbine models

When simulating a wind turbine one can either make a model as real as possible with fully resolved blade geometry, or one can make a simplified model where only the attributes of the wind turbine is included, such as with an actuator disk.

Kalvig et al. (2014) made a fully resolved turbine simulation trying to replicate measurements done in a wind tunnel, and compared it to simpler models. The fully resolved model stood out in accuracy of the wake predictions. Stergiannis et al. (2016) compared an traditional actuator disk model to a fully resolved wind turbine with good results, however, they only included the hub and not the nacelle which was mentioned in their suggestions for further work. The difficulties associated with advanced CFD models are revealed and discussed in the work by Tande (2011). The mesh may prove to be a difficult task having to make it fine enough to capture all the geometry without making it too big for the calculations, and then the validation of the results may be even harder to accomplish. Knowing the difficulties of advanced models makes it tempting to try simpler models.

An actuator disk has been used to solve many different problems in many different fields. It was the first mathematical model for propellers and wind turbines and was developed by Froude in 1889 based on the work of Rankine from 1865 (Hansen et al., 2006).

Madsen et al. (2013) used an actuator disk to model an offshore vertical axis wind turbine in 2D to look at the aeroelasticity of the turbine. Flow around an open propeller has been calculated using an actuator disk by Bontempo and Manna (2017) and by Conway (1995). Mikkelsen and Sørensen (2004) made a model of a wind turbine with an actuator disk and tested it for various conditions, and Amer et al. (2014) studied the far wake of wind turbines using actuator disks. Sánchez-Caja and Pylkännen (2007) modeled the propeller of a fishing vessel, while Lam et al. (2011) modelled a ship's propeller, both using an actuator disk. These are just some examples on the various cases using an actuator disk.

A recently published article by Stergiannis et al. (2017) compares actuator disk models with fully resolved turbine simulations using a multiple reference frame (MRF) and measurements of a real wind tunnel turbine test, Blind Test 4. They concluded that the classic AD model is underestimating the velocity deficit, while the fully resolved model agrees well with measurements. The study in the present report is similar to the study done by Stergiannis et al. (2016), however, the actuator disk model is improved and the fully resolved model is using a arbitrary mesh interface instead of a moving reference frame. The biggest difference between these two options is that MFR is a steady-state approximation of the transient rotational motion, while the method using AMI is fully transient (simScale Documentation, 2017). The fully resolved model is more complex and takes a lot more time and computer power to generate results, but the desired trade offs are more accurate results and better prediction of the actual wind turbine behaviour. Ideally, one would like to have a working model performing 3D transient simulations for different turbine designs where one would get the power output for the different designs.

This report builds on the work done in the author's project thesis "*Implementation and Testing of an Actuator Disk in OpenFOAM*" (Hoem, 2017). Here, an actuator disk model was created for OpenFOAM 4.1 based on the work of Svenning (2010). This report will provide a detailed description of how to implement an improved version of that actuator disk model, and a fully resolved turbine model with moving mesh. Data from Blind test 4 will be used to validate the results from the final models.

1.2 OpenFOAM

FOAM is short for Field Operation And Manipulation. OpenFOAM is a free open source software used in CFD. It was released by OpenCFD Ltd in 2004 and has over the years been made to compile with Linux, Mac OS X and Windows systems. It is continuously developed and bug checked by the OpenFOAM community. The software can be downloaded from `www.openfoam.com` and requires no expensive licences.

Due to being open source, there are many cases online to guide the users on how to solve their problem. OpenFOAM also provides a range of tutorials. However, the code usually comes with little commenting, and the User Guide (Greenshields, 2016) provides limited information. Online forums such as `www.cfd-online.com/Forums/openfoam` becomes a great source of inspiration and help.

OpenFOAM is not particularly user friendly. The user mainly works in a terminal window and with C++ files. The cases to be solved consists of a number of files at a number of directories, which are all compiled together and the results can be visualized using a program called paraView. On the other hand, the user gets a steep learning curve due to working directly with the code.

The simulations in this report is done with OpenFOAM 5.0, and some with the use of a high performance computer named Vilje at NTNU.

CHAPTER 2 THEORY

2.1 Governing equations

Navier-Stokes equations for mass continuity (Henningson and Berggren, 2005):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{2.1}$$

Navier-Stokes equation for conservation of momentum (Henningson and Berggren, 2005):

$$\frac{\partial}{\partial t}\int_V \rho \mathbf{u}\, dV + \int_A (\rho \mathbf{u}\mathbf{u}) \cdot \mathbf{n}\, dA = -\int_A p \cdot \mathbf{n}\, dA + \int_A \mu(\nabla \mathbf{u}) \cdot \mathbf{n}\, dA + \int_V \rho \mathbf{f}\, dV \tag{2.2}$$

Bernoulli's equation without change in potential energy (Henningson and Berggren, 2005):

$$\frac{1}{2}\rho U^2 + p = \text{constant along a streamline} \tag{2.3}$$

2.2 Turbulence

Turbulence is normally present in real flow, and to accurately calculate the turbulent behavior would be very time-consuming and costly, hence turbulence models are often used instead. The two most known and popular turbulence models are Reynolds averaged Navier-Stokes, RANS, and Large Eddy Simulation, LES.

Turbulence have eddies in many different sizes. LES calculates the large eddies and models the small ones. It is mainly used for 3D simulations and is known to generally give good results. However, LES requires high computational costs due to high resolution demand in 3D and time, and to near-wall equations (Nieto et al., 2015).

RANS uses the Reynolds decomposition which says that the variable of interest consists of an average part and a fluctuating part. Thus, the time average of the variable provides the main properties. RANS is less costly than LES and is often used in stationary flow (Gong and Tanner, 2009). There are several models within RANS and the most popular family is the two-equation models. They all rely on Boussinesq's approximation from 1877 (Müller, 2013) where the stress tensor is modeled in the viscous term of Navier-Stokes equation (2.2).

The $k - \epsilon$ model has been used due to it beeing such a popular and widely used turbulence model, even though it has its limitations, see section 2.2.1. Another model is the $SSTk - \omega$ model which in theory is better suited for problems involving external aerodynamics (Guerrero, 2014), see section 2.2.2.

2.2.1 k $-\ \epsilon$ model

The k- ϵ model is the most used two-equation turbulence models for RANS and is known to produce good results in large channel flow, however it struggles close to walls.

The Reynolds averaged Navier-Stokes equations become:

$$\frac{\partial U_i}{\partial x_i} = 0 \tag{2.4}$$

and

$$\frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial P}{\partial x_i} + \nu \frac{\partial^2 U_i}{\partial x_j^2} - \frac{\partial (\overline{u_i u_j})}{\partial x_j} \tag{2.5}$$

where U is the mean velocity, P is the mean pressure, and $(\overline{u_i u_j})$ is the Reynolds stress tensor that must be modeled.

Boussinesq's approximation of the stress tensor is

$$-(\overline{u_i u_j}) = \nu_T \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) - \frac{2}{3} k \delta_{ij} \tag{2.6}$$

where ν_T is the eddy viscosity. It's definition and the transport equations for the turbulent kinetic energy, k, and dissipating energy, ϵ are as follows.

$$\nu_T = c_\mu \frac{k^2}{\epsilon} \tag{2.7}$$

$$\frac{\partial k}{\partial t} + U_j \frac{\partial k}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_T}{\sigma_k} \right) \frac{\partial k}{\partial x_j} \right] - (\overline{u_i u_j}) \frac{\partial U_i}{\partial x_j} - \epsilon \tag{2.8}$$

$$\frac{\partial \epsilon}{\partial t} + U_j \frac{\partial \epsilon}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\nu + \frac{\nu_T}{\sigma_\epsilon} \right) \frac{\partial \epsilon}{\partial x_j} \right] - C_{1\epsilon} (\overline{u_i u_j}) \frac{\partial U_i}{\partial x_j} \frac{\epsilon}{k} - C_{2\epsilon} \frac{\epsilon^2}{k} \tag{2.9}$$

where equation (2.6) should be inserted in equation (2.8) and (2.9).

Equations (2.6) – (2.9) togheter with the following constants defined in *turbulence-Properties* are known as the $k - \epsilon$ model.

```
kEpsilonCoeffs
{
    Cmu       0.09;
    C1        1.44;
    C2        1.92;
    sigmak    1.0;
    sigmaEps  1.11;
}
```

As seen from equation (2.9) there will be a singularity if the $k - \epsilon$ model is used all the way to the wall where k is vanishing. To handle the computations near the wall the time scale $T_t = k/\epsilon$ is put to $T_t = k/\epsilon + T_k$ where $T_k = (\nu/\epsilon)^{1/2}$ is the Kolmogorov time scale – constant time scale of the smallest eddies. (Yang and Shih, 1993).

When defining a wall function a initial value must be stated. For epsilon and k these values can be found by the following equations (Larsson, 2006b).

$$k = \frac{3}{2}(UI)^2 \quad \text{where U is the mean flow and I is the turbulence intensity.} \tag{2.10}$$

$$\epsilon = C_\mu^{3/4} \cdot \frac{k^{3/2}}{l} \quad \text{where l is the turbulent length scale.} \tag{2.11}$$

The turbulent length scale describes the size of the eddies containing a lot of energy in turbulent flow. It should normally be smaller than the dimensions of the problem. For a turbine problem it is common to set the turbulent length scale to 5% of the channel height. But, when the inlet flow is bounded by walls with turbulent boundary layers, the turbulent length scale can be set to 0.22 of the inlet boundary layer thickness (Larsson, 2006b). A value of 0.09 has been chosen, and yields $l = 0.09 \cdot L$.

2.2.2 k $- \omega$ SST model

SST stands for Shear Stress Transport. The k-ω SST model performs well for the boundary layers along walls even with pressure separation. It is not as sensitive as the k-ω model when it comes to inlet boundary conditions. Thus, it is useful for problems involving aerodynamics or turbomachinery (Guerrero, 2014). The model is a blending of the k-ϵ and k-ω model, where the k-ϵ is used away from the surface and k-ω is used inside the boundary layer (Menter and Esch, 2001). The parameter ω is the turbulent frequency.

The model consists of the following transport equations.

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho U_j k}{\partial x_j} = \tilde{P}_k - \beta^* \rho \omega k + \frac{\partial}{\partial x_j}\left(\Gamma_k \frac{\partial k}{\partial x_j}\right) \tag{2.12}$$

$$\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho U_j \omega}{\partial x_j} = \frac{\gamma}{\nu_t} P_k - \beta \rho \omega^2 + \frac{\partial}{\partial x_j}\left(\Gamma_\omega \frac{\partial \omega}{\partial x_j}\right) + (1 - F_1) 2 \rho \sigma_{\omega^2} \frac{1}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} \tag{2.13}$$

where

$$\Gamma_k = \mu + \mu_t \sigma_k \tag{2.14}$$

$$\Gamma_\omega = \mu + \mu_t \sigma_\omega \tag{2.15}$$

$$P_k = \tau_{ij} \frac{\partial U_i}{\partial x_j} \tag{2.16}$$

$$\tilde{P}_k = min(P_k; c_1 \epsilon) \tag{2.17}$$

$$\mu_t = \rho \cdot \nu_t = \rho \cdot \frac{a_1 k}{max(a_1 \omega; S \cdot F_2)} \tag{2.18}$$

The blending function F_1 is defined as

$$F_1 = tanh\left(\left(min\left[max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right), \frac{4\rho\sigma_{\omega^2}k}{CD_{k\omega}y^2}\right]\right)^4\right) \tag{2.19}$$

with $CD_{k\omega} = max\left(2\rho\sigma_{\omega^2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-10}\right)$ where y is distance to nearest wall.

The second blending function F_2 used inside the first blending function is defined as

$$F_2 = tanh\left(\left(max\left[\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right]\right)^2\right) \tag{2.20}$$

All the constants in the model are computed from the corresponding constants for the k-ϵ and k-ω model using a blending function $\phi = \phi_1 F_1 + \phi_2(1 - F_1)$, where ϕ_1 stands for the coefficients of the k-ϵ model and ϕ_2 stands for the coefficients of the k-ω model. Note that in the implementation for this turbulence model the σ from theory is noted as $\alpha = 1/\sigma$. Thus the names in *turbulenceProperties* are alpha, not sigma. This yields the constants for the SST k-ω model (Menter et al., 2003):

```
kOmegaSSTCoeffs
{
    alphaK1        0.08;
    alphaK2        1.0;
    alphaOmega1    0.5;
    alphaOmega2    0.856;
    beta1          0.075;
    beta2          0.0828;
    betaStar       0.09;
    gamma1         5/9;
    gamma2         0.44;
    a1     0.31;
    b1     1.0;
    c1     10.0;
    F3     no; // Extra feature for rough walls not used.
}
```

For the wall functions using ω the value can be found by the following equation (Larsson, 2006b).

$$\omega = \frac{\sqrt{k}}{l} \tag{2.21}$$

2.2.3 Spalart-Allmaras turbulence model

Spalart-Allmaras is a one-equation turbulence model which gives a transport equation for the modelled eddy turbulent viscosity. It was specifically made for aerodynamic problems since the excising models were complex and unclear and did not solve the boundary layer well enough for detached flows (Spalart and Allmaras, 1994). The Spalart-Allmaras model in OpenFOAM is implemented without the trip-term which can be left out when doing boundary layer calculations, thus the f_{t2} terms from Spalart and Allmaras (1994) final equation is omitted here. The transport equation becomes:

$$\frac{D\tilde{\nu}}{Dt} = c_{b1}\tilde{S}\tilde{\nu} - c_{w1}f_w\left[\frac{\tilde{\nu}}{d}\right]^2 + \frac{1}{\sigma}\left[\frac{\partial}{\partial x_j}\left((\nu+\tilde{\nu})\frac{\partial\tilde{\nu}}{\partial x_j}\right) + c_{b2}\frac{\partial\tilde{\nu}}{\partial x_i}\frac{\partial\tilde{\nu}}{\partial x_i}\right] \tag{2.22}$$

where

$$\tilde{\nu} = \frac{\nu_t}{f_{v1}} \quad , \quad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \quad , \quad \chi \equiv \frac{\tilde{\nu}}{\nu}, \quad \tilde{S} \equiv S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2} \quad ,$$

$$f_{v2} = 1 - \frac{\chi}{1+\chi f_{v1}} \quad , \quad f_w = g\left[\frac{1+c_{w3}^6}{g^6 + c_{w3}^6}\right]^{1/6} \quad , \quad g = r + c_{w2}(r^6 - r) \quad ,$$

$$r \equiv \frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2} \quad , \quad c_{w1} = \frac{c_{b1}}{\kappa} + \frac{1+c_{b2}}{\sigma}$$

The production term of $\tilde{S}$ needs to be limited so that it does not produce nonphysical results, thus Spalart gives the bounded value where $\tilde{S}$ is clipped at $C_s \cdot \Omega$, where

$$\Omega = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} - \frac{\partial U_j}{\partial x_i}\right)$$

The parameter *d* is the distance from the field to the nearest wall. For the boundary conditions use $\tilde{\nu}_{\text{t, wall}} = 0$ and $\tilde{\nu}_{\text{t, far field}} = 3\nu$.

The model constants are:

```
SpalartAllmarasCoeffs
{
    Cb1         0.1355;
    Cb2         0.622;
    Cw2         0.3;
    Cw3         2.0;
    Cv1         7.1;
    Cs          0.3;
    sigmaNut    0.66666;
    kappa       0.41;
}
```

2.2.4 Law of the wall

The Law of the wall assumes that the turbulence close to the boundary is a function of the flow conditions close to the wall, not the flow conditions far away (Brennen, 2016). It is found by dimensional analysis using the following parameters:

1. Distance y from the wall
2. Mean velocity ū(y)
3. Shear stress τ_{w}
4. Fluid density ρ
5. Fluid kinematic viscosity ν

Defining friction velocity u_τ yields dimensionless length y^+ and velocity u^+:

$$u_\tau = \left(\frac{\tau_w}{\rho}\right)^{1/2} \quad , \quad y^+ = \frac{y u_\tau}{\nu} \quad \text{and} \quad u^+ = \frac{\bar{u}}{u_\tau} \tag{2.23}$$

This yields that $u^+ = u^+(y^*+)$ and thus $\tau_w = \rho\nu\frac{\partial u}{\partial y}$ becomes $u^+ = y^+$ very close to the wall in a turbulent flow. This describes the *viscous sublayer*. The turbulent fluctuations dominate further from the wall and the equation for the *log-law layer* becomes $u^+ = \frac{1}{\kappa}ln(y^+) + C$. See figure 2.1 for the different regions.

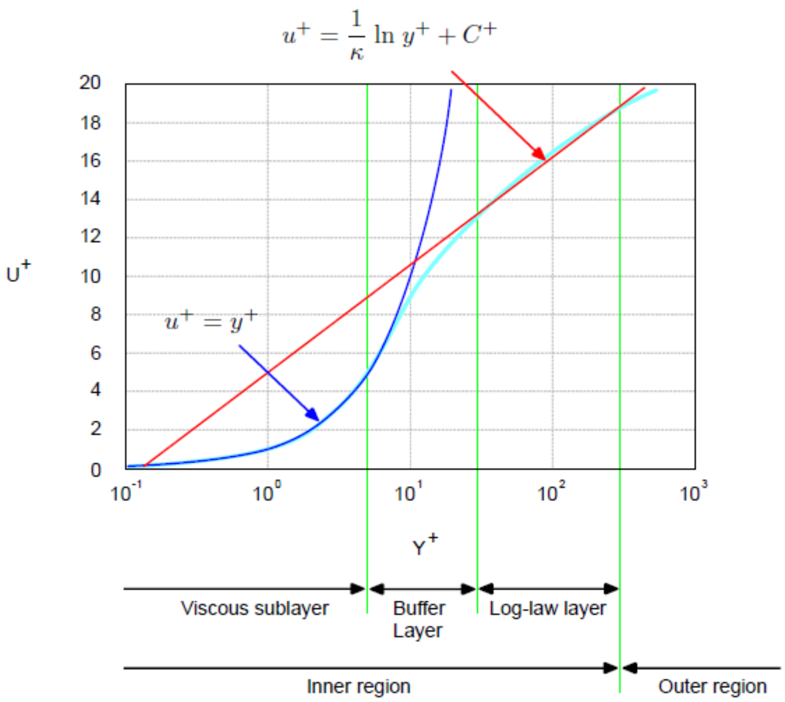


Figure 2.1: Law of the wall; different regions of the boundary layer (Guerrero, 2014).

For near-wall treatment there are three options depending on the value of y^+:
1. $30 \leq y^+ \leq 300 \quad \rightarrow$ use wall-functions.
2. $1 \leq y^+ \leq 300 \quad \rightarrow$ use scalable wall functions.
3. $y^+ \leq 6 \quad \rightarrow$ resolve boundary layer without wall functions.

2.3 Actuator Disk

The aerodynamics of a non specified wind turbine or fan can be analyzed using information about the extracted or added kinetic energy from the flow over the turbine or fan. This can be done by creating a generic device called an actuator disk.

2.3.1 Adding kinetic energy to the flow by a volume force

Forces acting on the flow induced by a fan can be modelled by a volume force. Assuming the volume force follows the Goldstein optimum (Goldstein, 1929) and both total thrust and torque is known, the volume force can be described using equations (2.24)–(2.26) and figure 2.2 and 2.3.

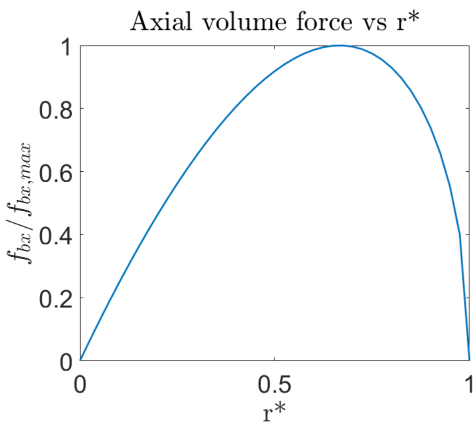


Figure 2.2: Normalized axial volume force vs normalized radius.

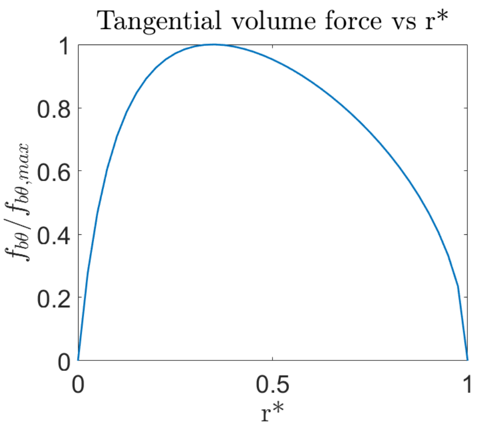


Figure 2.3: Normalized tangential volume force vs normalized radius. $r'_h = 1/9$.

The volume force described here is varying radially and delivers thrust and torque to the fluid. It is calculated and decomposed into axial and tangential components using the distribution described in Erik Svenning's files and the note by FINE/Marine (2009).

The radial volume force component $\mathbf{f}_{br}$ is equal to zero, while the other components are potentially non-zero:

$$\mathbf{f}_{bx} = A_x r^* \sqrt{1-r^*} \quad , \quad \mathbf{f}_{b\theta} = A_\theta \frac{r^*\sqrt{1-r^*}}{r^*(1-r'_h)+r'_h} \tag{2.24}$$

where

$$r^* = \frac{r'-r'_h}{1-r'_h}, \quad r'_h = \frac{R_H}{R_P}, \quad r' = \frac{r}{R_P}$$

The two constants in (2.24) are given by Svenning (2010) as:

$$A_x = \frac{105}{8} \frac{T}{\pi\Delta(3R_H+4R_P)(R_P-R_H)} \tag{2.25}$$

$$A_\theta = \frac{105}{8}\frac{Q}{\pi \Delta R_P(3R_P + 4R_H)(R_P - R_H)} \tag{2.26}$$

The two volume force components $\mathbf{f}_{bx}$ and $\mathbf{f}_{b\theta}$ are in the axial and tangential direction, respectively. The forces are per unit volume and are normalized by $\rho U^2/L$. Summing up all the axial force components for the whole volume of the actuator disk yields the total thrust, and summing up all the tangential components multiplied with their radius yields the total torque. This volume force creates both thrust and swirl of the flow.

2.3.2 Extraction of kinetic energy

The goal of a wind turbine is to extract kinetic energy from the wind, convert it into mechanical energy by rotating a shaft, and finally convert to electrical energy in a generator.

Here the flow is assumed to be homogeneous, steady state incompressible and inviscid. The turbine is assumed to have infinite number of blades (Burton et al., 2008).

If kinetic energy is extracted the flow will slow down and affect the surroundings. Assuming that the air affected by the rotating wind turbine blades is separated from the rest of the flow, a stream tube such as in Figure 2.4 is defined.

Mass flow through a stream tube will always be constant because the stream surfaces acts like impermeable walls.

$$\dot{m}_{st} = \rho \vec{U} A = constant \tag{2.27}$$

The continuity equation (2.27) shows the relationship between mass flow rate $\dot{m}_{st}$, flow velocity $\vec{U}$ and flow area A.

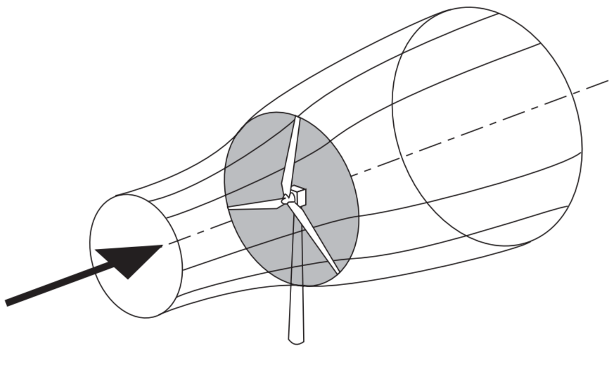


Figure 2.4: Sketch of a stream tube around a wind turbine (Burton et al., 2008)

For constant mass flow of an incompressible fluid, velocity and area are inversely proportional. The area increases when the velocity decreases, and vice versa. A sudden drop in pressure after the wind turbine implies extraction of pressure energy. The air that is flowing towards the wind turbine is slowing down even before it hits the turbine. Due to no energy extraction, the decrease in velocity is compensated for by an increased static pressure to keep the mechanical energy constant. Some of the energy induced by the increased static pressure is extracted from the air as it passes through the rotor disk. Thus, the static pressure drops to under atmospheric levels and we get a decrease in kinetic energy. The flow behind the rotor is now reduced in velocity and pressure. The region between the rotor and far downstream where the pressure is at atmospheric levels again, is called the wake of the flow. For the pressure to increase again, the velocity has to be slowed down even more. This means that from far upstream to far downstream of the wind turbine the pressure is the same, but the kinetic energy is reduced.

2.3.3 Actuator Disk Theory

A way to model the extracted kinetic energy over a wind turbine is by an actuator disk. Due to mass conservation the mass flow rate upstream, at the disk and far downstream must be the same inside the stream tube. Burton et al. (2008) point out that the velocity variation at the actuator disk can be expressed as -aU_∞. This represents the kinetic energy extracted in the region. Thus, the stream-wise velocity at the actuator disk is

$$U_d = U_\infty(1-a), \tag{2.28}$$

where a is the axial flow induction factor; the fraction of the free stream wind energy extracted at the actuator disk. There will be a change in velocity across the actuator disk causing a change in momentum. U_W is the wake velocity, and U_∞ is the far field velocity. Then:

$$\text{Rate of change of momentum} = (U_\infty - U_W)\rho A_d U_d \tag{2.29}$$

This change in momentum is caused by the pressure difference across the actuator disk, which is the only force acting on the stream tube. The stream tube is surrounded by atmospheric pressured air which gives no net force. The pressure difference can be obtained by using Bernoulli's equation (2.3) upstream and downstream of the disk separately. This yields:

$$(p_d^+ - p_d^-) = \frac{1}{2}\rho(U_\infty^2 - U_W^2) \tag{2.30}$$

By inserting equation (2.28) and (2.30) multiplied with the disk area A_d into equation (2.29) we get an expression for the wake velocity:

$$U_W = (1-2a)U_\infty \tag{2.31}$$

At the actuator disk the velocity is the free stream velocity minus the induced flow. However, in the wake the velocity is the free stream velocity minus twice the induced flow. Thus, half of the axial speed loss happens upstream of the actuator disk. Due to the pressure jump over the actuator disk, a force is induced on the air in this region. Using equation (2.30), (2.31) and multiply with the actuator disk area, yields

$$F = (p_d^+ - p_d^-)A_d = 2\rho A_d U_\infty^2 a(1-a) \tag{2.32}$$

The work done by this force, and the power that is extracted from the air, together with the power coefficient, are expressed as

$$Power = FU_d = 2\rho A_d U_\infty^3 a(1-a)^2 \tag{2.33}$$

$$C_P = \frac{Power}{\frac{1}{2}\rho U_\infty^3 A_d} = 4a(1-a)^2 \tag{2.34}$$

The expression in the denominator represents the available power in the air if the actuator disk was not present. Thus, the higher a power coefficient, the more efficient it is. However, there is a upper limit – The Betz limit.

The maximum value of the power coefficient happens when the derivative is zero:

$$\frac{dC_P}{da} = 4(1-a)(1-3a) = 0 \text{ which gives } a = \frac{1}{3}. \tag{2.35}$$

Thus we get a maximum value of

$$C_{P_{max}} = \frac{16}{27} = 0.593 = 59.3\% \tag{2.36}$$

It is not possible for the power coefficient to reach 100 % because this would imply that all the kinetic energy in the air is extracted by the actuator disk, and thus the air after the disk is at rest. If the air after the disk is at rest it will block any more air from entering the disk region, and we will not extract any more energy. If the induction factor a is larger than 0.5, the velocity of the wind after the actuator disk, in the wake, will be negative, see equation (2.31). If this happens, the actuator disk theory no longer applies.

Another useful dimensionless value is the thrust coefficient. It comes from the definition of thrust:

$$T = \frac{1}{2}\rho A_d(U_\infty^2 - U_W^2) = F = \frac{Power}{U_d} \tag{2.37}$$

The thrust coefficient C_T is thus defined as

$$C_T = \frac{Power}{\frac{1}{2}\rho U_\infty^2 A_d} = 4a(1-a) \tag{2.38}$$

An actuator disk using this theory will add thrust to the flow, but no swirl.

2.4 Discreatisation / Finite Volume Method

The Navier-Stokes equation (2.2) can be rewritten in integral form as

$$\oint (\rho\phi\mathbf{u})\cdot\mathbf{n}\,dA = \oint \Gamma\frac{d\phi}{d\mathbf{x}}\,dA + \int S(\phi)\,dV \tag{2.39}$$

where the first term is the convection term, the second is the diffusion term and the third is the source term. The pressure term and the viscous term are included in the diffusion term (Versteeg and Malalasekera, 2007).

Using Gauss theorem to convert a surface integral to a volume integral, and assuming the flux vector function is smooth, the equation on differential form can be turned into the steady one dimensional convection and diffusion equation:

$$\frac{d}{dx}(\rho u \phi) = \frac{d}{dx}\left(\Gamma \frac{d\phi}{dx}\right) + S(\phi) \tag{2.40}$$

which must satisfy the continuity equation $\nabla \cdot (\rho u A) = 0$. Integrating the transport equation 2.40 yields

$$(\rho u A \phi)_e - (\rho u A \phi)_w = \left(\Gamma A \frac{d\phi}{dx}\right)_e - \left(\Gamma A \frac{d\phi}{dx}\right)_w + \bar{S} V_P \tag{2.41}$$

where $\bar{S}$ is the mean value of the source term, V_P is the volume of the central cell P and Γ is the diffusion coefficient.

Defining the convection flux, F, and diffusion conductance, D, as

$$F_e = (\rho u)_e \qquad \text{and} \qquad D_e = \left(\frac{\Gamma}{\delta x}\right)_w$$

and assume that

$$\frac{d\phi_e}{dx} \approx \frac{\Delta \phi_e}{\delta x_{PE}} = \frac{\phi_E - \phi_P}{\delta x_{PE}}$$

and that the terms involving the western face is estimated similarly. Assuming that $A_e = A_w = A$ the terms in 2.41 can be written

$$F_e \phi_e - F_w \phi_w = D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W) + S_u + S_p \phi_P \tag{2.42}$$

Here the source term has been approximated to be consisting of a uniform part, S_u and a non-uniform part $S_p \phi_P$. Continuity equation yields that $F_e - F_w = 0$

To be able to solve equation 2.42 both ϕ_e and ϕ_w must be approximated. This can for example be done by the upwind FVM discretisation described in section 2.4.2.

2.4.1 Discretisation schemes in OpenFOAM

All solvers and cases may have different schemes, and the defaults may change for each of them. They are defined for each case in the *system/fvSchems* file in the case directory. In OpenFOAM the pressure and velocities are defined for a collocated grid, thus they are cell values. See Figure 2.5 (b). An example of schemes in a *fvScheme* file is shown below.

```
gradSchemes
{
    default          Gauss linear;
}
divSchemes
{
    default           none;
    div(phi,U)        bounded Gauss upwind;
    div(phi,epsilon)   bounded Gauss upwind;
    div(phi,k)         bounded Gauss upwind;
    div((nuEff*dev2(T(grad(U)))))      Gauss linear;
}
laplacianSchemes
{
    default            Gauss linear limited corrected 0.33;
}
```

The gradient scheme is used for the pressure term, the divergence scheme is used for the convection term, and the Laplacian scheme is used for the viscous term (Greenshields (2016)).

2.4.2 Upwind finite volume method

When having high velocity inflow, the flow becomes strongly convective. In this case the finite volume method using upwind differencing scheme is beneficial because the flow direction is taken into account.

The convective value of ϕ is taken to be equal to the value of the upstream cell center. Assuming that the flow is in the positive direction such that all velocities and convective fluxes are greater than zero. The convected value of ϕ at a cell face *w* is made equal to the value of the upstream cell center W and similar for ϕ at the face *e*: $\phi_w = \phi_W$ and $\phi_e = \phi_P$, see figure 2.5 (a).

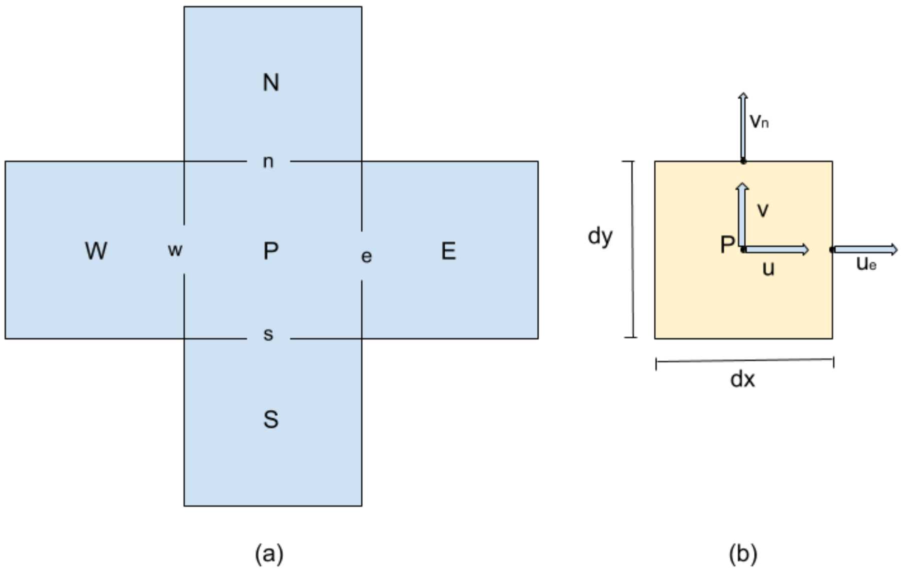


Figure 2.5: Sketch of the upwind differentiating scheme. (a) Sketch of cells and faces for a 2D grid, (b) position of u_e and v_n on faces of cell P.

This means that equation (2.42) can be written as

$$F_e\phi_P - F_w\phi_W = D_e(\phi_E - \phi_P) - D_w(\phi_P - \phi_W) + S_u + S_p\phi_P \tag{2.43}$$

which can be rearranged to give a more general form

$$a_P\phi_P = a_W\phi_W + a_E\phi_E + S_u \tag{2.44}$$

where

$$a_W = D_w + F_w \quad , \quad a_E = D_e \quad \text{and} \quad a_P = a_W + a_E - S_P$$

Remember continuity which yields $F_e - F_w = 0$. This is all done assuming that the flow moves from left to right. Thus, a more general version of the coefficients above, which is valid for both flow directions are

$$a_W = D_w + max(F_w, 0) \quad \text{and} \quad a_E = D_e + max(0, -F_e)$$

The discretisation in equation (2.44) with its coefficients can be expanded to 2D by adding cells north and south of the center cell, as seen in figure 2.5 (a). This yields

$$a_P\phi_P = a_W\phi_W + a_E\phi_E + a_S\phi_S + a_N\phi_N \tag{2.45}$$

where

$$a_W = D_w + Fw \ , \ a_E = D_e, \ , \ a_S = D_s + F_s \ , \ a_N = D_n \text{ and } a_P = a_W + a_E + a_S + a_N$$

Here, this discretisation is done for a 2D convection-diffusion equation. However, to be able to solve the Navier-Stokes equation including the pressure gradient in a transient problem, a solution algorithm like SIMPLE or PIMPLE is needed.

2.5 SIMPLE algorithm

The SIMPLE solution algorithm, used by *simpleFoam* in OpenFOAM, is a steady state, incompressible solver. SIMPLE stands for Semi-Implicit Method for Pressure-Linked Equations and goes through the following iterative process (Passalacqua, 2014):

1. Guess a pressure field.
2. Solve the discretized momentum equation (2.45) for the intermediate velocity field.
3. Compute the mass fluxes at the cells faces.
4. Solve the pressure correction equation defined from continuity equation using under-relaxation.
5. Correct the mass fluxes at the cell faces.
6. Correct the velocities on the basis of the new pressure field.
7. Update the boundary conditions and the pressure field.
8. Repeat until convergence.

Due to being an incompressible and steady state solver, the Navier-Stokes momentum equation (2.2) is rewritten when used in *simpleFoam*, and becomes:

$$\frac{\partial}{\partial t}\int_V \phi \, dV + \int_A (\phi \mathbf{u}) \cdot \mathbf{n} \, dA = \int_A \mathbf{R} \cdot \mathbf{n} \, dA + \int_V \mathbf{S} \, dV \tag{2.46}$$

where $\mathbf{R} = \nabla \cdot \tau + \nabla \cdot p$ and ϕ is the generic mass flux, S is the source(s) added by *fvOptions* and $\boldsymbol{\tau}$ is the shear rate tensor. Since it is an incompressible solver all terms are divided by ρ. Thus, the pressure is divided by the density (Greenshields, 2016).

The solution algorithm is stated in the transport equation for velocity located in */opt/openfoam4/applications/solvers/incompressible/simpleFoam/UEqn.H*. This file shows OpenFOAMs representation of equation (2.46) by the two following sections.

```
    fvm::div(phi, U)
  + MRF.DDt(U)
  + turbulence->divDevReff(U)
 ==
    fvOptions(U)
```

and

```
if (simple.momentumPredictor())
{
    solve(UEqn == -fvc::grad(p);
    fvOptions.correct(U);
}
```

The pressure correction and the coupling between pressure and velocity is given by the file *pEqn.H* also located in the solver folder. The time steps used in the *simpleFoam* solver are integration time steps, not actual time steps since they are not relevant for a steady state solution. Thus, the time step in *controlDict* is set to 1.

2.6 PIMPLE algorithm

The PIMPLE algorithm is a combination of the PISO and SIMPLE algorithms. Is is used to couple the pressure and momentum quantities whilst fulfilling the mass conservation. The PIMPLE algorithm allows for transient calculation at larger currant numbers, which allows for bigger time steps.

There is an outer and inner loop, where the outer correction loop ensures that the explicit parts of the equations are converged. The inner loops goes as follows: a flux field generates a velocity field, the velocity field is used to correct the pressure, and the pressure is used to correct the flux field again. One or more loops are done inside one time step.

To include the outer loop of the algorithm there need to be stated `nOuterCorrectors`, and to include the inner loop `nCorrectors` must also be stated in the *fvSolution* dictionary. These number of correctors states how many correction loops the calculations are to perform. The `nOuterCorrectors` is for the pressure-momentum correction loop.

`nNonOrthogonalCorrectors` is the number of pressure field correction loops that are made, and is similar to the loops made by the outer correctors. To ensure a stable and more robust algorithm under-relaxation is used, and the factors for each field must be stated in *fvSolution* similar to below. (Holzmann, 2016).

```
PIMPLE
{
   correctPhi        no;
   nOuterCorrectors  2;
   nCorrectors       1;
   nNonOrthogonalCorrectors  2;
}

relaxationFactors
{
   "(U|k|epsilon).*"  1;
}
```

The equation of momentum that the PIMPLE algorithm solves is equal to the one in SIMPLE, except that time differential is used for calculations here.

$$\frac{\partial}{\partial t}\int_V \phi \, dV + \int_S \phi \mathbf{u} \cdot \mathbf{n} \, dS = \int_S \mathbf{R} \cdot \mathbf{n} \, dS + \int_V \mathbf{S} \, dV \tag{2.47}$$

where $\mathbf{R} = \nabla \cdot \boldsymbol{\tau} + \nabla \cdot p$ and ϕ is the generic mass flux, **S** is the source(s) added by *fvOptions* and $\boldsymbol{\tau}$ is the shear rate tensor. Since it is an incompressible solver all terms are divided by ρ. Thus, the pressure is divided by the density (Greenshields, 2016).

First term is the time derivation, the second term is the convective term, the third term is shear-rate tensor and the last term is the additional sources term. In OpenFOAM the momentum equation stated in *UEqn* looks like the following:

```
   fvm::ddt(U) + fvm::div(phi, U)
  + MRF.DDt(U)
  + turbulence->divDevReff(U)
==
   fvOptions(U)
```

Which shows the time derivative included. The pressure coupling becomes:

```
if (pimple.momentumPredictor())
{
   solve(UEqn == -fvc::grad(p);
   fvOptions.correct(U);
}
```

2.7 Dynamic mesh

A dynamic mesh is a mesh that can involve moving boundaries. The idea is that with only nine commands you can do everything possible on a mesh. These commands are adding, modifying or remove either points, faces or cells (Gschaider, 2005). Some examples of such situations are planes as they take off and land, missile release and the motion of wing actuators (Jasak, 2009).

The dynamic mesh actions are divided into two groups; mesh deformation and topological changes. Cells may be added or removed when there are significant deformation. While topological changes manipulate the mesh resolution and connectivity to take into account the boundary changes. This is typically done using sliding interfaces and cell layering.

The momentum equation using a dynamic mesh becomes:

$$\frac{\partial}{\partial t}\int_V \phi\, dV + \int_S \phi(\mathbf{u}\text{-}\mathbf{u}_b)\cdot\mathbf{n}\, dS = \int_S \mathbf{R}\cdot\mathbf{n}\, dS + \int_V \mathbf{S}\, dV \tag{2.48}$$

where $\mathbf{u}_b$ is the velocity of the mesh. FVM handles mesh motion, but it requires additional algorithmic steps to solve topological changes. A problem such as a rotating wind turbine with a stator such as the wind tunnel around it would require a mesh that can withstand topological changes.

The position of the mesh points are based on the prescribed boundary motion and determined by the *automatic mesh motion*. When the mesh moves, the mesh must remain geometrically valid by preserving the spatial consistency, as seen in figure 2.6.

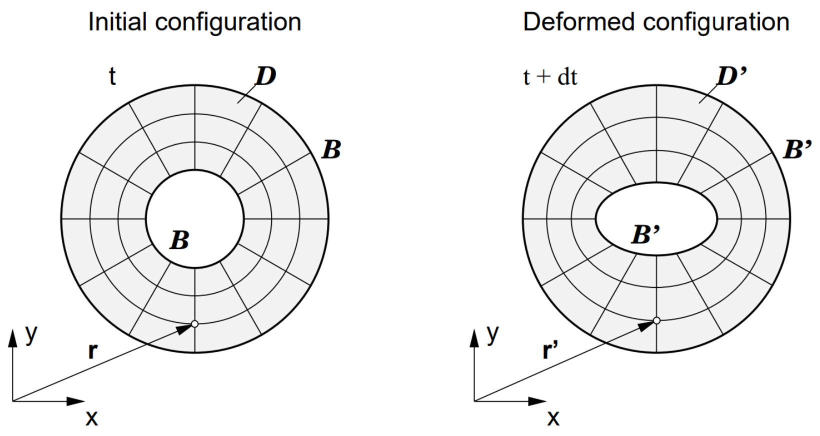


Figure 2.6: Illustration of automatic mesh motion showing the deformation of a configuration (Jasak, 2009).

There are three different areas of topological change handling components; *primitive operations*, *topological modifiers* and *dynamic mesh classes*. Using primitive mesh operations means adding or remove a point, a face or a cell to implement topological changes.

Topology modifiers is quite similar to the primitive mesh operations, but some modifications such as attach/detach boundary, layer addition or removal, and sliding interfaces can be added. The dynamic meshes are made by combining mesh definition and topology modifiers. Last, the dynamic mesh classes use multiple modifiers together in a recognizable geometry which interacts with complex mesh motion. Examples are mixing vessels, pistons or valves. The topological changes are well known even without specifying the geometry for the specific case. (Jasak, 2009).

To solve a transient, incompressible problem with Newtonian fluid using a dynamic mesh the solver *pimpleDyMFoam* can be used (Greenshields, 2016). One such example is the *mixerVesselAMI2D* tutorial which will be used to further explain and investigate moving meshes.

2.8 Wind Tunnel data

The wind tunnel to be used for the simulations in OpenFOAM is the wind tunnel at NTNU in Trondheim, Norway. It is a closed-loop wind tunnel with a rectangular test section of 2.71 m in width, 1.81m in height and 11.15m in length, and is the largest wind tunnel in Norway. The roof is adjusted for zero pressure gradient creating a constant velocity in the entire test area. The wind tunnel is driven by a fan downstream which is able to generate wind speed up to 30 m/s or about 110 km/h. (Bartl and Sœtran, 2017).

Figure 2.7 shows a sketch of the closed-loop wind tunnel. The bottom channel is the testing area.

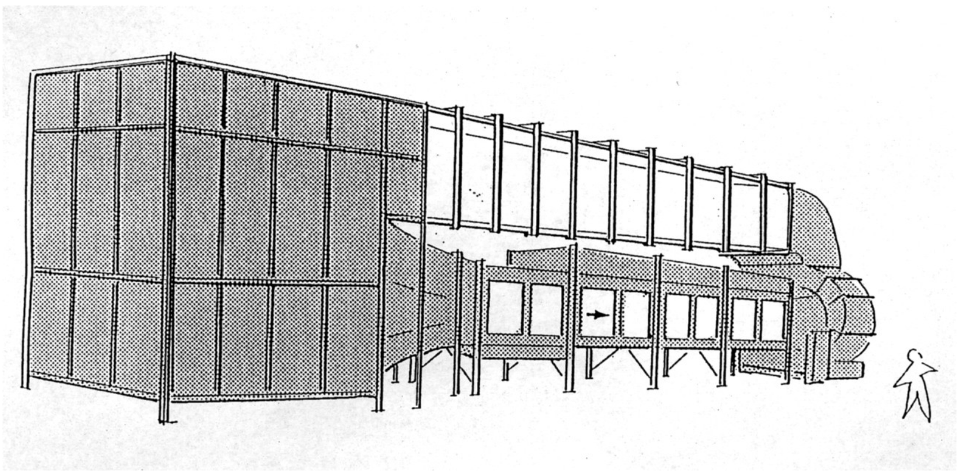


Figure 2.7: Sketch of the Wind Tunnel at NTNU in Trondheim. Showing the closed-loop system and the testing area. The sketch is provided by the Faculty of Engineering (Sœtran, 2017).

2.8.1 Specifications from Blind Test 4

The inflow has a turbulence intensity is 0.23 $\%$ with an inlet length scale of 0.045m (Bartl and Sœtran, 2017). Turbulent intensity $= u'_{rms}/U$ is a measure of mean turbulent kinetic energy per unit mass (Russo and Basse, 2016). With such low turbulent intensity the flow can be put as uniform at the inlet of the tunnel without causing too much errors.

The dimensions given for the turbine are outer diameter of 0.944m, inner diameter of 0.13m and a thickness of 0.04m. The turbine is positioned 2 diameters downstream of the inlet, at 1,788m at a hub height of 0.827m. The bulk velocity is set to 11.5 m/s which together with a Reynolds number of $\sim 10^5$ give $\nu \sim 10^{-4}$. This is the same test case as *Test A* described in the blind test invitation (Sœtran and Bartl, 2015).

At design condition the tip speed ratio is 6 which gives a rotational speed ω=146.18 Hz, $C_P = 0.45$ and $C_T = 0.79$ (Sœtran and Bartl, 2015). Thus, from the power coefficient and the thrust coefficient the thrust, T, and torque, Q, added by the turbine can be calculated as follows.

$$T = \frac{\rho U^2 \pi R^2 C_T}{2} = 43.87N \tag{2.49}$$

$$P = \frac{\rho U^3 \pi R^2 C_P}{2} = 287.40W$$

which gives the torque

$$Q = \frac{P}{\omega} = 1.97Nm \tag{2.50}$$

where $\rho = 1.2kg/m^3$, $U = 11.5m/s$ and $R = 0.472m$ (Sœtran and Bartl, 2015). These values for thrust and torque have been confirmed to be correct by Jan Bartl who wrote the concluding report after Blind Test 4 (Bartl, 2017).

CHAPTER 3

ACTUATOR DISK MODEL WITH IMPERMEABLE HUB

The work done in the project thesis by Hoem (2017), which this model builds on, showed that excluding a hub in the simulations means excluding certain effects at the center of the actuator disk representing the turbine. Therefore, the *ActuatorDiskExplicitForce* model by Hoem (2017), which was first created by Svenning (2010), is now remade with a hub. The original definition of the actuator disk can be seen in figure 3.1. The turbine is represented by an actuator disk which imposes a volume force in the actuator disk region, creating thrust and torque. This definition in *fvSolution* will still be used. However, the mesh in the new version of this model will be different.

$$\text{Real body force} = f_{OF} \cdot \rho \forall_{AD}$$

```
actuatorDisk
   {
     startPoint        (0.02 0 0);
     endPoint          (-0.02 0 0);
     thrust            43.87;
     torque            1.97;
     density           1.2;
     interiorRadius     0.065;
     exteriorRadius     0.472;
   }
```

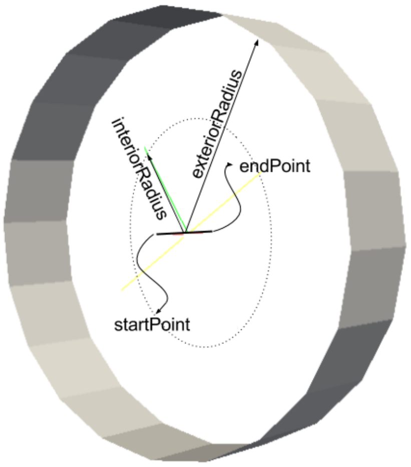


Figure 3.1: The geometry of an Actuator Disk (Hoem, 2017).

Note that the *startPoint* and the *endPoint* can be switched to make a turbine or a fan. The example above, with starting point downstream of end point, creates a turbine.

To create a hub in the existing mesh of the actuator disk one could remake the whole mesh and define a cylinder as a hub in the center of the mesh using `blockMesh`. This means creating the mesh around a cylinder with a lot more blocks and vertices instead of the one-block setup which was used in the project thesis.

Another option is to use the `snappyHexMesh` utility together with a CAD model of the hub in ASCII .stl format. Due to the fact that in later simulations knowledge about `snappyHexMesh` will be very useful, and the amount of work it takes to redefine the existing mesh to a cylindrical mesh in 3D, the approach using `snappyHexMesh` is chosen.

Using data from Blind Test 4 invitation (Sœtran and Bartl, 2015) a hub and nacelle has been created using a free software called FreeCAD. The hub and nacelle has a radius of 65 mm and a height of 639.9 mm in x-direction. The center of the axis are in the center of the hub. The rounding of the edges is arbitrary due to simplicity. See figure 3.2.

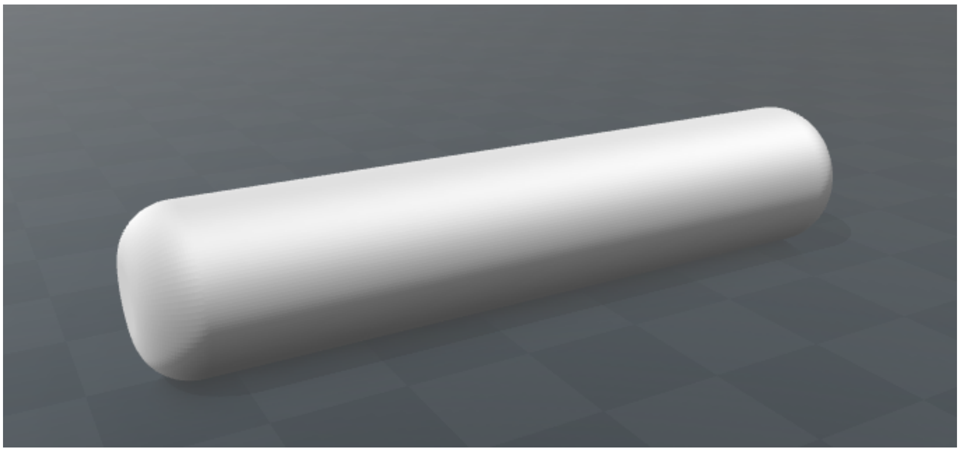


Figure 3.2: Picture of the CAD model made in FreeCAD of the hub for the actuator disk simulation.

First, the axis system in the one-block mesh defined in *blockMeshDict* must be the same as the one for the hub so that the hub can be placed correctly. Thus, the origin of the wind tunnel is also placed in the center of the hub, as sketched below in figure 3.3. This means that the location of the actuator disk in *fvSolution* needs to be changed, and later also the *LocationInMesh* inside *snappyHexMeshDict* must be changed. More on that in section 3.1.

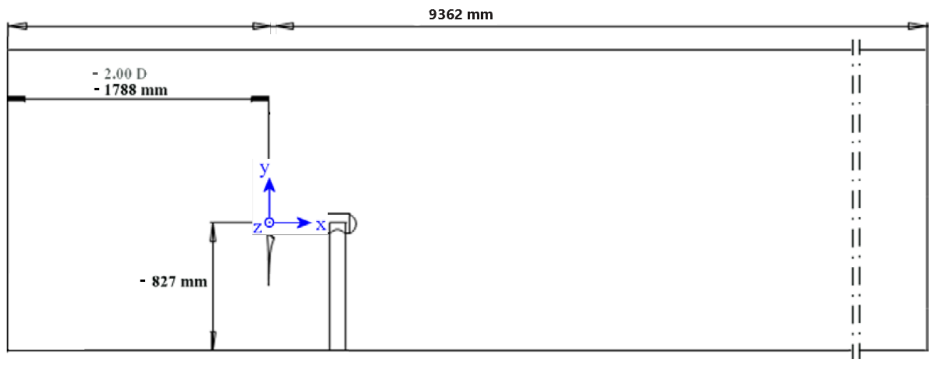


Figure 3.3: Sketch of the wind tunnel setup with correct axis. Modified sketch from Sœtran and Bartl (2015).

3.1 snappyHexMesh

The tool `snappyHexMesh` will refine or sculpture a geometry onto an already existing mesh. Therefore, there must be a mesh already made using `blockMesh`. Here, `blockMesh` will make a rectangular mesh consisting of only hexes, and that is essential. There can be no other types of cells in the background mesh. When using `snappyHexMesh` a defined area of the mesh will be refined, it will cut through the existing mesh at the interface with the defined surface and mesh around the new surface.

There are three main functions called upon by *snappyHexMeshDict*; *autoRefineDriver* which is responsible for the refining and cutting of the mesh, *autoSnapDriver* which snaps the mesh onto the surface, and *autoLayerDriver* which creates the boundary layer (Järpner, 2011). These functions is activated by the following code lines in *snappyHexMeshDict*.

```
castellatedMesh    true;
snap               true;
addLayers          true;
```

Before you know that the mesh is good, write *false;* after *addLayers*. It is better to have a working mesh before adding boundary layer refinements, which may cause problems. Later this will be set to *true*.

In the subdictionary *snappyHexMeshDict.geometry* the user must define what surface should be cut and snapped to by `snappyHexMesh`. It should look similar to this:

```
geometry
{
    ActuatorDiskHub.stl
    {
          type      triSurfaceMesh;
          name      ActuatorDiskHub;
    }
    refinementBox
    {
          type      seachableBox;
          min      (-0.5   -0.7   -0.7);
          max      (9.3    0.7    0.7);
    }
};
```

where the *ActuatorDiskHub.stl* is the file of the surface that the user wants. The file must be located in the */constant/triSurface* folder. Note that FreeCAD, as many other CAD programs, produces the file in mm, while OpenFOAM uses m. Thus, the file must be rescaled. The file from FreeCAD is first named *ActuatorDiskHubmm.stl*, and is converted to m and named *ActuatorDiskHub.stl* by typing the following line in the terminal.

```
> surfaceTransformPoints -scale '(0.001 0.001 0.001)'
ActuatorDiskHubmm.stl ActuatorDiskHub.stl
```

The *refinementBox* is an area around and downstream of the actuator disk to capture the effects in the wake better. It spreads all the way to the back wall of the tunnel.

The *LocationInMesh* must be defined as a point in the region of the mesh that you want to keep. In this case the tunnel part of the mesh should stay and the mesh inside the hub is to be cut out. Thus the point must be located inside the tunnel and outside the hub.

By using the function `surfaceFeatureExtract` the surfaces of the CAD model of the hub are extracted. Thus, they can be used as *features* for edge refinement in *snappyHexMeshDict*. The function creates a .emesh file located in */constant/triSurface*. The *ActuatorDiskHub* must also be defined as one of the *refinementSurfaces* in *snappyHexMeshDict*. In addition, the *refinementBox* is put at the *refinementRegions* with *mode inside*. To see the full snappyHexMeshDict file, see Appendix B.

To create the mesh, first add the new boundary *ActuatorDiskHub* to all the /0 files, then type in the following in the terminal.

```
> blockMesh
> surfaceFeatureExtract
> snappyHexMesh -overwrite
> checkMesh
```

First the background mesh is created. Then the surfaces of the hub is extracted which is used when snapping the new mesh around the new boundary. If this runs successfully the result should be similar to figure 3.4 to 3.7. For large meshes it is nice to run the meshing on several cores using *decomposePar*, which can be investigated in the User Guide (Greenshields, 2016).

There are a few more alterations to the case setup that needs to be done to make the simulations run with k-ω SST turbulence model. In *fvSolution* there must be added a solver for `omega` similar to the solver for `epsilon`. In *fvSchemes* there must be added a differential scheme for the turbulence frequency, ω, and a way of calculating the wall distance; `wallDist` is added as seen below. Lastly, there must be a */0/omega* file.

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

Thus, the case *actuatorDiskExplicitForce* is ready to run.

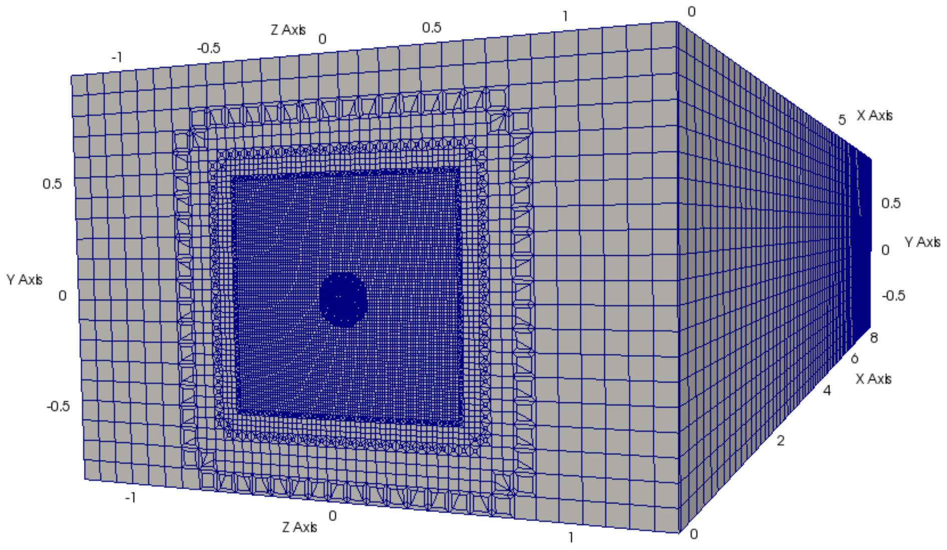


Figure 3.4: Mesh showing the refined region around the actuator disk region which stretches throughout the wake, and the hub refinement in the wind tunnel.

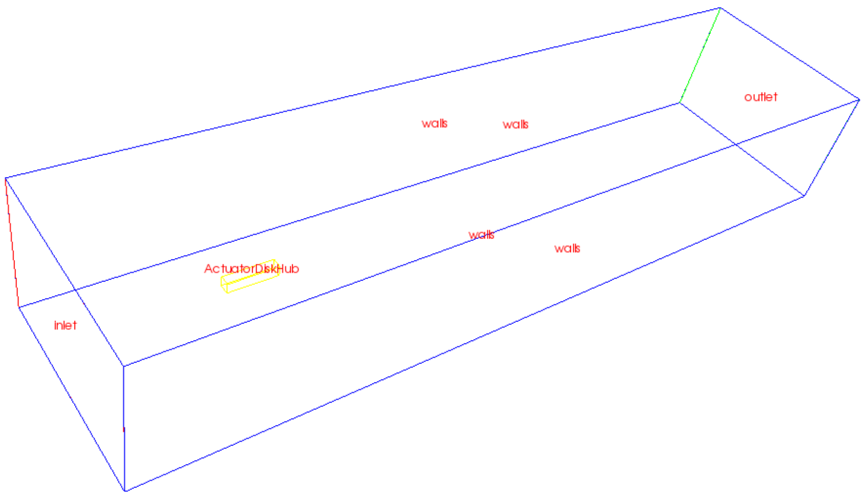


Figure 3.5: Diagram of the hub location in the mesh.

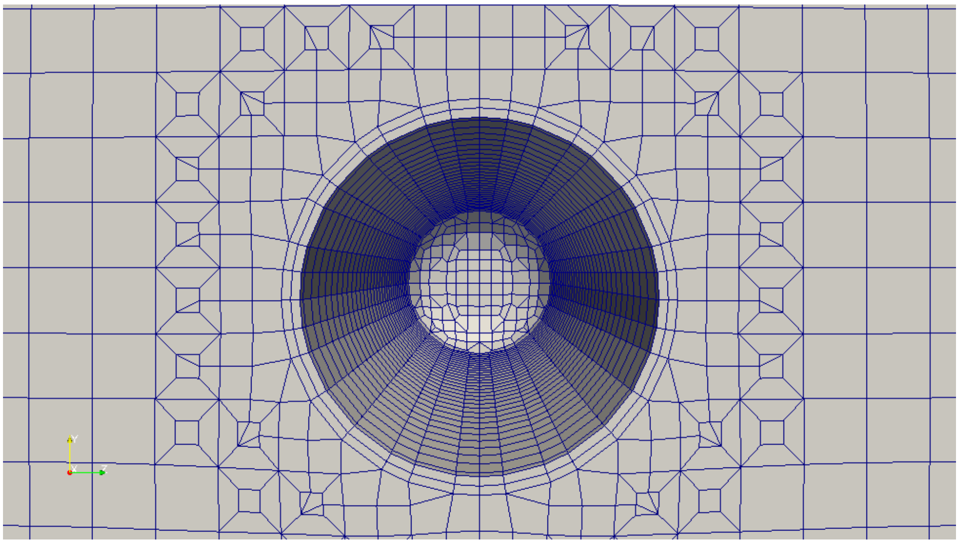


Figure 3.6: Mesh around the hub showing the added layers at the surface.

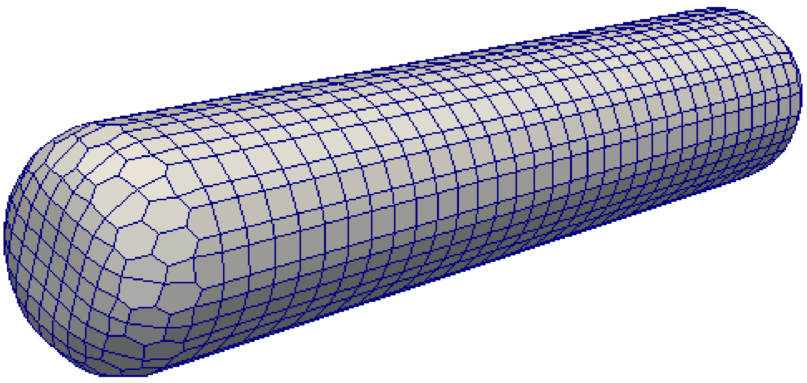


Figure 3.7: Mesh on the surface of the hub. The volume that is cut out from the wind tunnel mesh.

CHAPTER 4 MOVING MESH CASE STUDY

To get familiar with how a moving mesh works the *mixingVesselAMI2D* tutorial from OpenFOAM has been investigated.

4.1 Setup of the mixerVesselAMI2D tutorial

The *mixerVesselAMI2D* tutorial provided by OpenFOAM uses dynamic mesh with AMI patches. AMI stands for Arbitrary Mesh Interface. AMI patches is used to simulate across disconnected, adjacent mesh domains. It is very handy when simulating rotating geometries where you have a rotating part and a stationary part. Each part need to have their separate mesh while at the same time they need to be connected. This is done by using the boundary condition *cyclicAMI*. It requires that the user defines two neighbouring patches in *blockMeshDict*. It looks something like this:

```
AMI1
{
   type            cyclicAMI;
   neighbourPatch  AMI2;
   transform       noOrdering;
}
```

with a similar code for the second patch *AMI2*.

AMI1 corresponds to the rotor side (source) of the interface, and AMI2 corresponds to the stator side (target). The two boundaries communicate through the arbitrary mesh interface with interpolation. The faces at the interface use weighted contributions from the neighbour patch, defining the contribution as a fraction of the intersecting areas. The sum of the weights for each face should sum up to 1 if the geometries are to be well matched, if not there may be conservation errors, but they are usually very small.

The *transform* specification is the operation required to map the *neighbour* patch on to the *source* patch. The different options are (Greenshields, 2016):

- noOrdering: no mapping defined, the operation is determined by the patch.
- coincidentFullMatch: no transform defined, checks if the patch faces are matching.
- rotational
- translational

The dynamic mesh settings are described in */constant/dynamicMeshDict*. For the *mixerVesselAMI2D* case the type of `dynamicFvMesh` is set to `dynamicMotionSolverFvMesh`. It is a mesh that morphs around a set of specified boundaries and calculates the motion based on the pressure on those boundaries. This mesh is used mainly for rigid body motion. It solves a Laplace's equation for the motion displacement, or the motion velocity to calculate the updated point positions using one of the equations below. Which equation in (4.1) that is solved is chosen by the user by selecting a dynamic mesh solver.

$$\nabla \cdot (\gamma \nabla d_m) = 0 \quad \text{or} \quad \nabla \cdot (\gamma \nabla u_m) = 0 \tag{4.1}$$

In this tutorial the dynamic mesh solver used is `solidBody` with `rotatingMotion` function. This function defines the rotation using a defined origin, axis of rotation and an angular speed. The dynamic mesh solver is a displacement solver for solid-body motion.

There are several solvers and functions that can be used for each specific case, and you find them in the folder */src/dynamicMesh/motionSolvers* that comes with OpenFOAM.

Another dynamic mesh type is the `staticFvMesh` which is a mesh without motion. It is very useful when debugging a simulation that involves mesh motion. No parameters required for use. (Nozaki, 2014).

4.2 Testing the *mixingVesselAMI2D* tutorial

The case is built up by a rotor and a stator which is separated by an arbitrary moving interface (AMI). To be able to define all these regions there are several radii defined as seen in figure 4.1. The rotor includes the hub of radius *r* and four rotor blades with a tip radius of *rb*. The stator includes a outer circular casing at radius *R*, and four baffles at radius *Rb*. The radius *ri* defines the middle of the region between the baffle tips of the stator and the rotor, and this is where the arbitrary mesh interface (AMI) is located. The mesh at smaller radii than *ri* is rotation, while the mesh at larger radii is stationary.

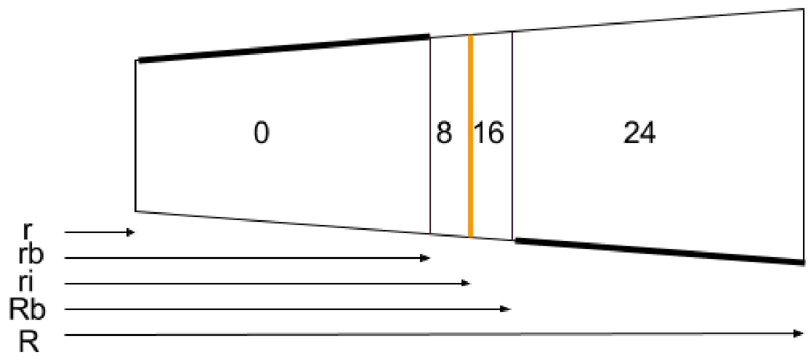


Figure 4.1: The definitions of the different radii used for the *mixingVesselAMI2D* tutorial.

The numbering of the blocks used to set up the mesh using `blockMesh`, and how the stator and rotor is located is shown in figure 4.2. Here, the hub is also visible as a hole in the mesh. The mesh is made up of 8 main blocks where each of these blocks are divided into 4 more blocks to accommodate for the different boundaries defined at different radii. After the blocks are defined correctly arcs can be added to create a smoother circle.

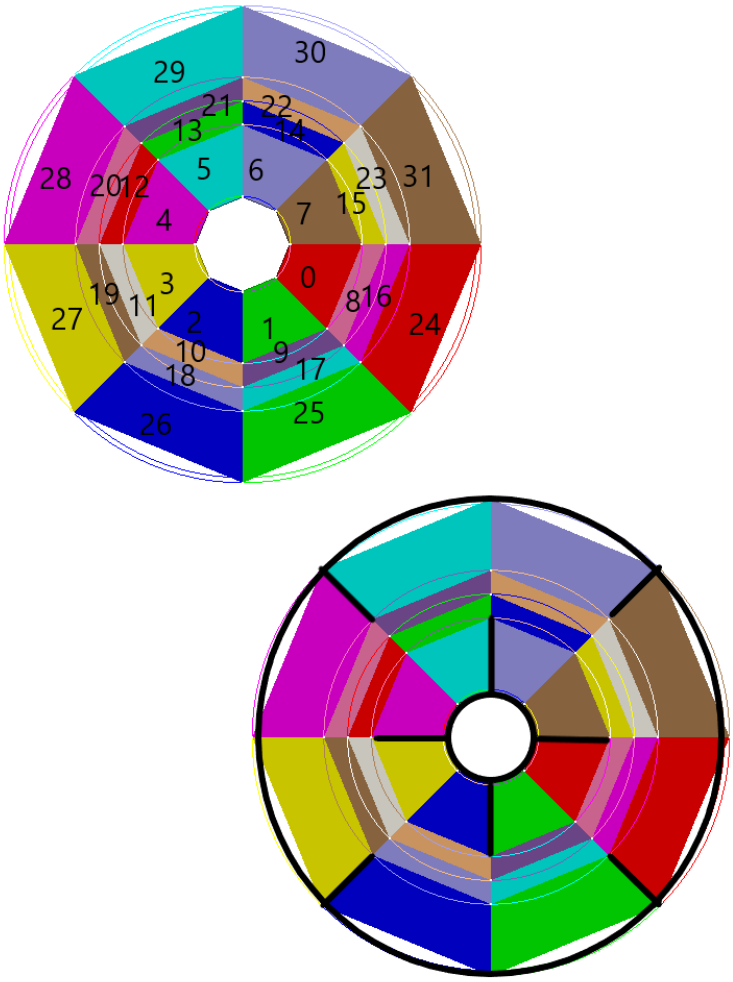


Figure 4.2: The upper figure shows the numbering of the blocks. The lower figure shows the location of the rotor and stator as thick black lines.

The 8 main blocks are symmetric by quadrants, which makes the problem identical in all four quadrants of the mixer vessel. This means that the solution and simulation of this problem should be the same if only one quadrant is solved for and then added to the other quadrants. To check whether this is true for the *mixerVesselAMI2D* tutorial or not, velocity field comparisons have been made.

The results are almost identical, see figure 4.3. The velocity fields looks the same and there are no visible differences between the calculation of the four quadrants. This suggests that the AMI acts the same all around its circumference, as is desirable.

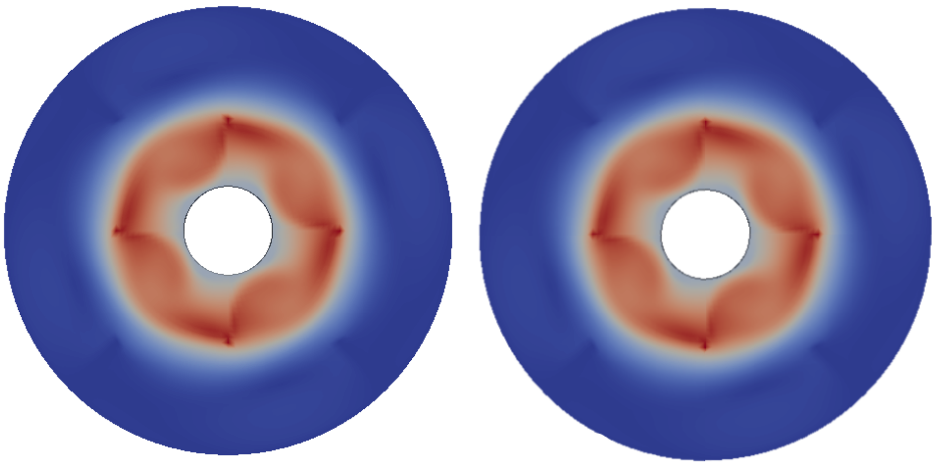


Figure 4.3: Left: Original output from simulation. Right: First quadrant is solved, rotated and added to the other quadrants.

Although this test was satisfactory, there is another test to be conducted; check that the AMI itself does not cause much error or numerical noise simply by being present. To check this, a test case has been made where the mixer vessel is placed in the middle of a channel. The outer casing and its baffles are removed together with the rotor. The moving mesh which used to be around the rotor is now all the way to the center of the circle and of type `staticFvMesh`, see 4.1. Thus the AMI is present, but there is nothing rotating. The upper wall of the channel is moving at steady velocity whilst the lower wall is stationary. In theory, such a case will produce Couette flow.

First, the mesh from the tutorial must be redefined. There are two options; one with wedges at the center and one with squares at the center. Both meshes can be seen in Figure 4.4 and 4.5. The mesh with wedges get very thin cells at the center which is not ideal. This is not a problem with the second mesh where the center has uniform grid of hexes. When running `checkMesh` everything looks fine; the meshes have a low and good value for cell skewness and maximum non-orthogonality, the cell volumes are not too small, and the meshes are deemed OK.

To test that there is nothing wrong with the meshes they are ran with `simpleFoam` with $Re = 1$ and the top wall moving at $U = 1m/s$ to see that they do indeed create Couette flow, which they do. The results can be seen in figure 4.6 and 4.7. Note that there are more points in the plot for the mesh with wedges because there are more cells in the center than the mesh with cubes which makes the plot look more like a full line. However, the mesh with cubes at center has better cell quality.

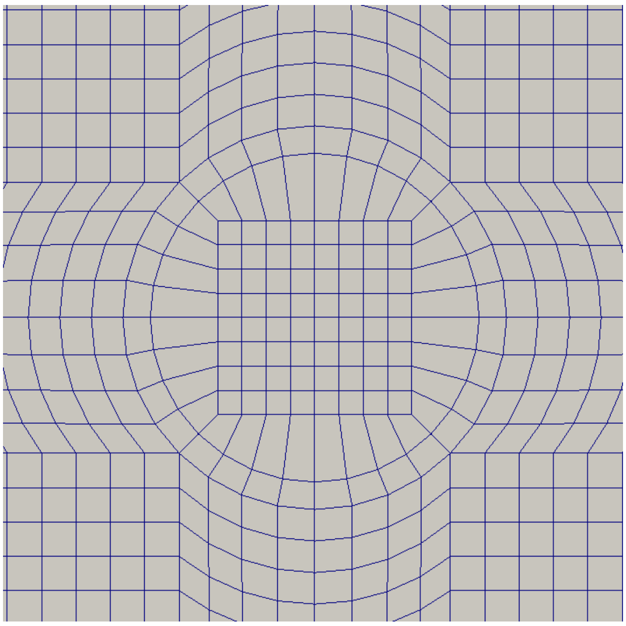


Figure 4.4: Mesh with squares at center.

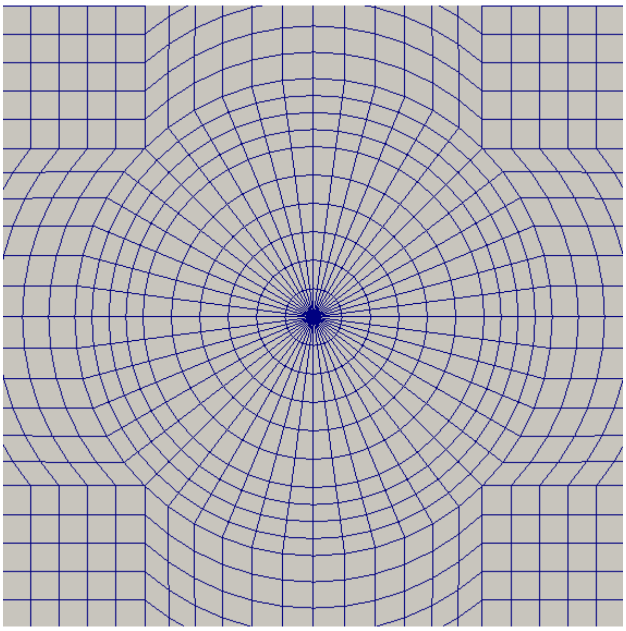


Figure 4.5: Mesh with wedges at center.

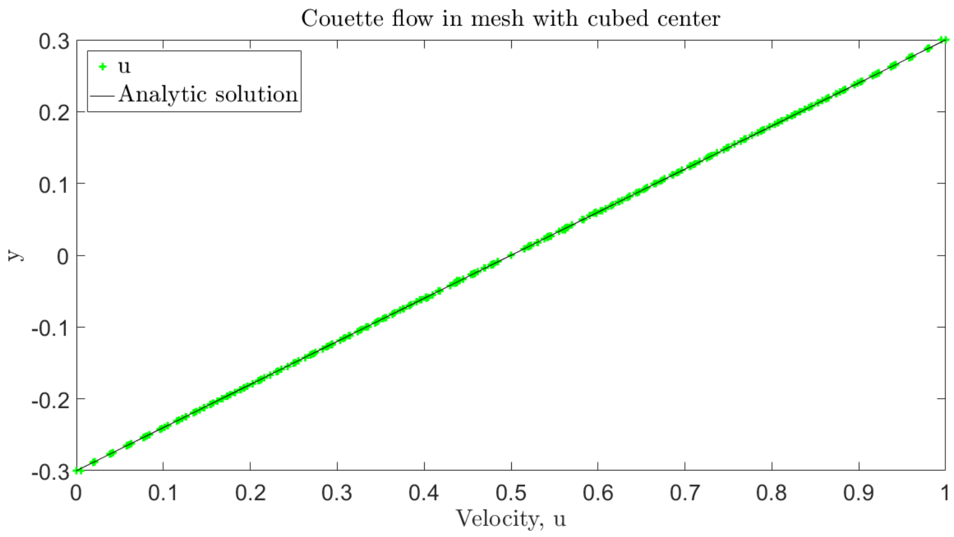


Figure 4.6: Couette flow using the mesh with cubes at center. Solved with simpleFoam.

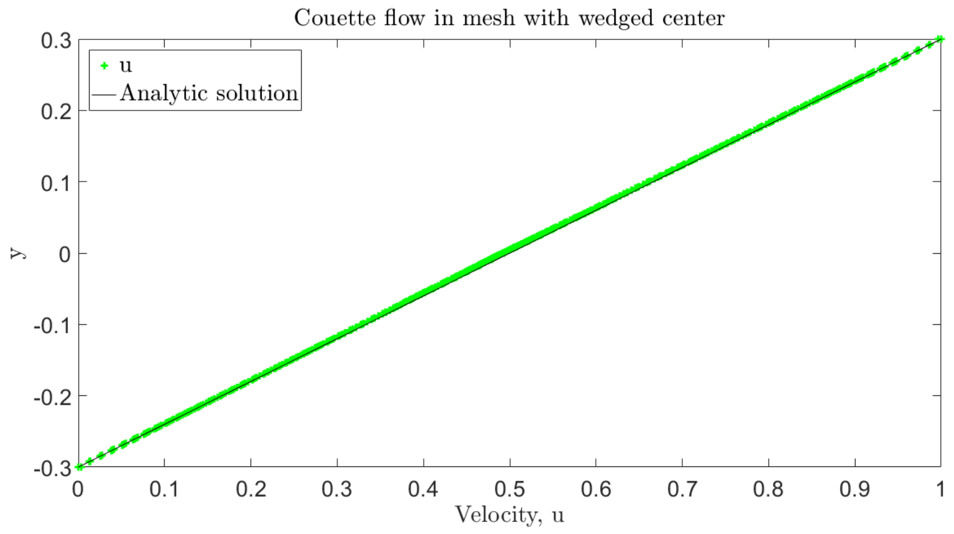


Figure 4.7: Couette flow using the mesh with wedges at center. Solved with simpleFoam.

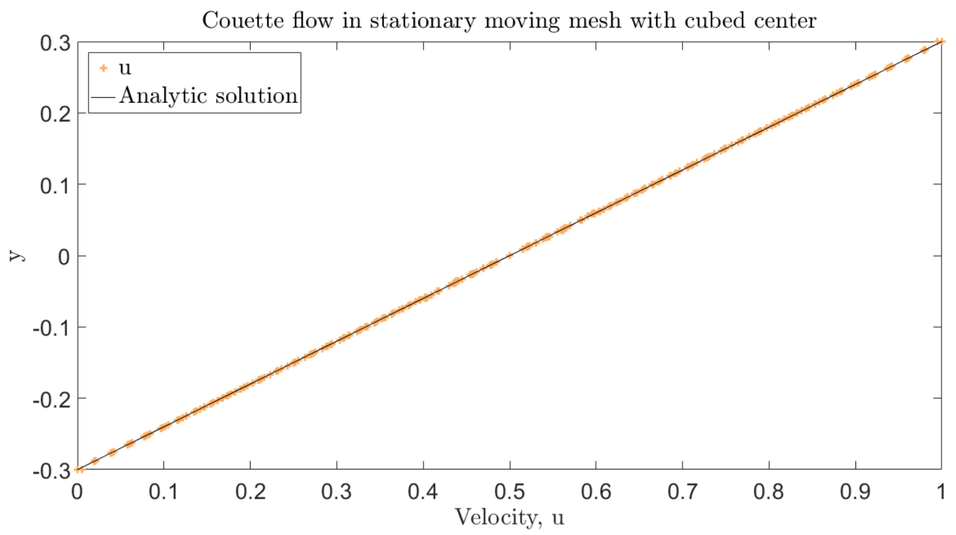


Figure 4.8: Couette flow using cubed mesh and pimpleFoam solver without AMI.

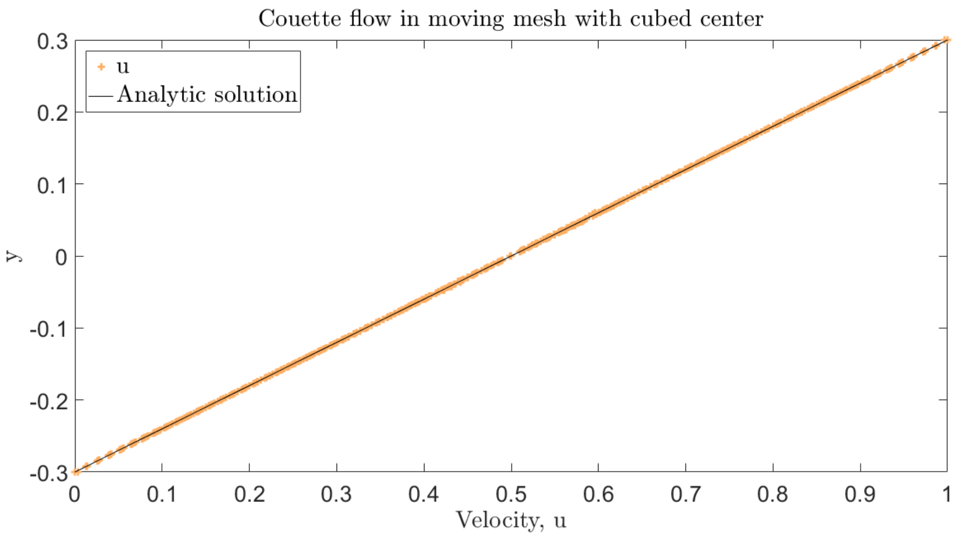


Figure 4.9: Couette flow using the cubed mesh and pimpleDyMFoam solver with AMI.

Next step is to see if the `pimpleDyMFoam` solver can do the same. For this test, only the mesh with cubes at center will be used. The first try is to use the mesh without AMI faces included to check the solver itself. If the solver behaves as it should on the circular mesh, the AMI interface can be introduced and tested. After adjusting the correctors mentioned in 2.6, the two test cases, with and without AMI creates couette flow and thus performs as intended. The results can be seen in 4.8 and 4.9. Note that to define the AMI regions there needed to be more blocks and thus more cells which makes the plot look more like a full line than the plot for mesh without AMI.

The tests in this section have performed well, and the solver `pimpleDyMFoam` seems to perform as intended when using a moving mesh with arbitrary moving interfaces. However, there are some small disturbances in the corners intersecting the circular mesh and the cubes in the center. Since they are present in both the mesh without AMI and the one with AMI, it is concluded that it is not the AMI and the moving mesh that is causing this. The small disturbances are caused by small pressure differences accumulated in the corners mentioned above.

There are some adjustments of the number of correctors, mentioned in section 2.6, that must be done when using the solver on different cases. However, this can easily be done with trial and error, although it may take some time.

CHAPTER 5 ADVANCED TURBINE MODEL

To create the advanced turbine model the OpenFOAM Propeller tutorial is used as a starting point. This chapter describes step by step how the tutorial is changed to create the desired model of the situation described by Blind Test 4 (Sœtran and Bartl, 2015), and some of the challenges along the way. To establish how fine the mesh should be a study have been made of an airfoil with known lift and drag coefficients to see what levels of refinement is needed in *snappyHexMeshDict* and the refinement of the background mesh created by *blockMeshDict*.

5.1 Modifying the Propeller tutorial

The Propeller tutorial is made up by an outer cylinder which is the calculation domain, an inner cylinder witch defines the AMI and a CAD model of a propeller. The steps to create the wind tunnel case from the propeller tutorial are the following.

Start by defining the wind tunnel in *blockMeshDict* with inlet, outlet and fixed walls as before. Then add CAD models of the turbine and the accompanying inner cylinder to */constant/triSurface* as .stl files. Similar to the *actuatorDiskExplicitForce* model with a hub the origin of the tunnel, turbine and the inner-cylinder must all be in the same location. The files *snappyHexMeshDict* and *surfaceFeatureExtractDict* needs to be changed to correspond with the new model which means the new CAD models must be added, and the outer cylinder must be removed. The `surfaceFeatureExtract` collects information about the surfaces that are to be used in *snappyHexMeshDict* to define and refine the right areas of the mesh.

Since the outer domain is defined in *blockMeshDict* instead of *snappyHexMeshDict* the inlet and outlet is already defined and the creation of a new inlet and outlet is not needed anymore. Thus, the file *createInletOutletSets* may be deleted, and its call command in the run file *Allrun* must be removed. The file *dynamicMeshDict* must be changed to have a rotational speed of $146.18 rad/s$ and rotational axis $(-1\ \ 0\ \ 0)$ according to the right hand rule. To enable the use of the k-ω SST turbulence model the same changes as previously described on page 26 for the *actuatorDiskExplicitForce* model must be done to the case files.

Those are roughly the steps needed to create the advanced turbine case from the Propeller tutorial. However, there are several things to be aware of when setting up the case which will be discussed in the following paragraphs.

The CAD model of the turbine used for the advanced turbine model is provided by

Bartl and Sœtran (2017). Some changes are made; only the hub, the three blades and the nacelle is kept for the model, while the tower and tunnel is excluded. See figure 5.1 to see the CAD model used. According to a study done by van der Auweraert (2015) the exclusion of the turbine tower does not effect the far away wake noticeably, only the near wake is effected. Thus, since the actuator disk model neither has a tower it is neglected for the advanced turbine model as well.

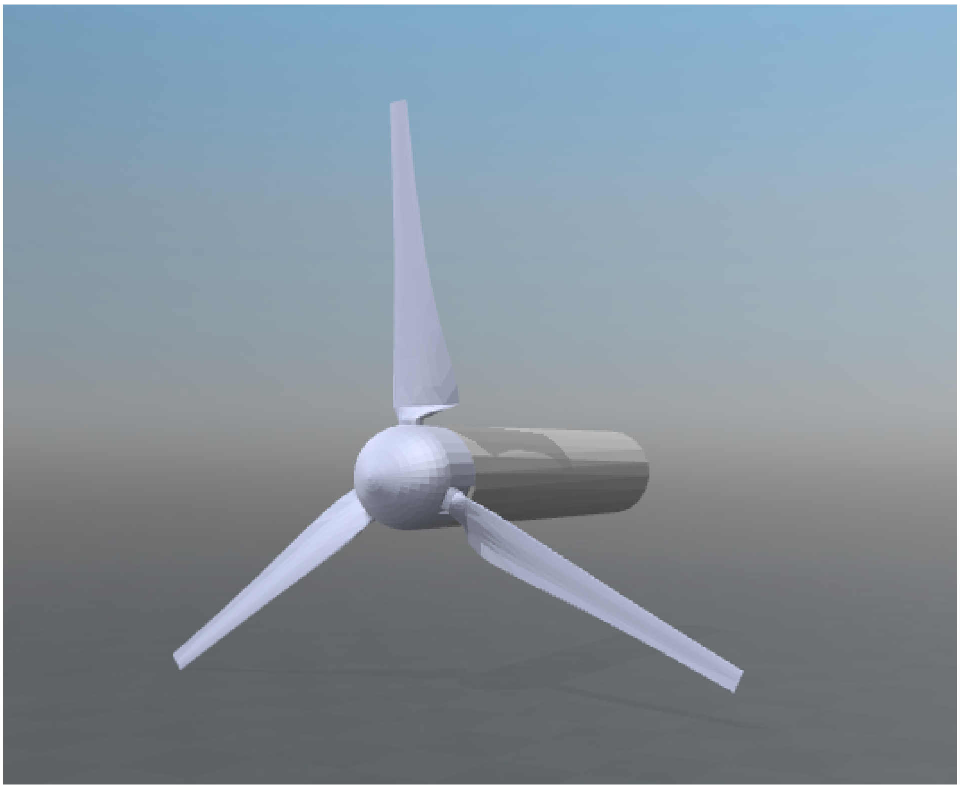


Figure 5.1: CAD model of the turbine used for the measurements in the Wind tunnel at NTNU. CAD model provided by Bartl (2017).

The CAD model for the inner cylinder is large enough to contain all of the turbine blades inside and stretches from a little upstream of the hub to a little downstream of the blades. The larger part of the nacelle is not included in the rotating region since it is to be kept stationary. Again, according to van der Auweraert (2015), the rotating region, the enclosure of the AMI, should be placed quite close to the blades for better results.

There are some important specifications set in *snappyHexMeshDict* to create a good arbitrary mesh interface (AMI). An important part of this case setup is to create a moving mesh where the weights for the AMI information transfers are good. The entry for *snappyHexMeshDict.refinementSurfaces* for the inner cylinder is defined as either `faceType boundary` as in the Propeller tutorial, or `faceType baffles`. The last option is a popular choice by CFD Online OpenFOAM forum users.

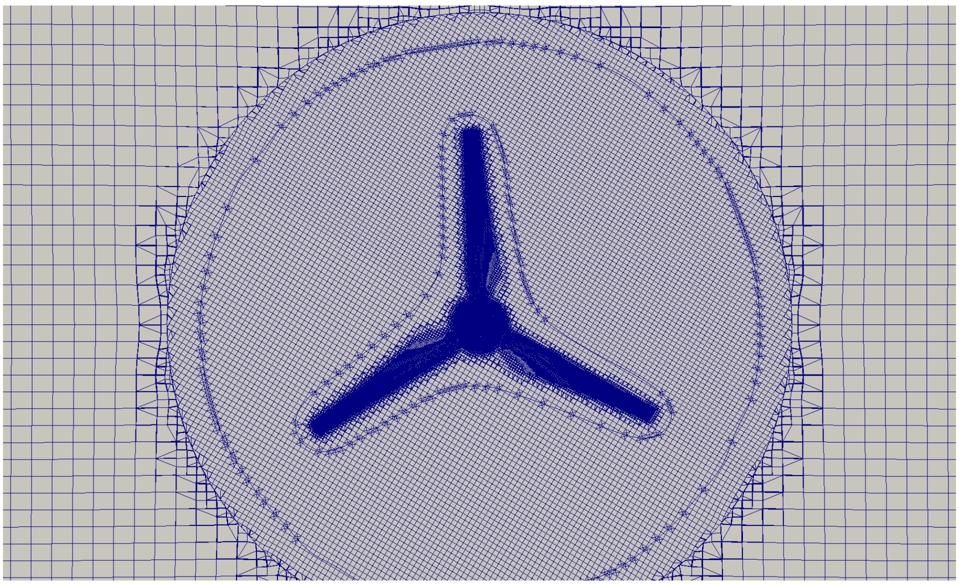


Figure 5.2: Sliding mesh with good AMI allocation during rotation.

The cells inside (rotating) and outside (stationary) of the AMI slides next to each other throughout the rotation of the mesh along a common boundary, see figure 5.2. When using `faceType boundary` the AMI is made as an actual boundary, while the option `faceType baffles` creates a zero-volume boundary. It creates co-located pairs of faces which has the same connection throughout the rotation (Greenshields, 2013).

It is important that the origin stated in *dynamicMeshDict* is exactly the same as the origin of the AMI region, here the origin of the inner cylinder. If not, this creates a distorted mesh which can not be used, see figure 5.3.

As mentioned earlier it is desired that the weights from the contributing cells at the AMI are close to 1 for good results. For complex geometries this is not easy to achieve for all cells. However, since the number of cells where the geometries are not well matched usually are low there is a trick that can be used – introducing the `lowWeightCorrection` option. It is added in *createPatchDict*. This ensures that the cells with a weight under a specified limit, say 0.2, is not included and should not cause the whole case to fail. These cells are given a *zeroGradient* attribute and will not contribute to the calculations, the contributions will only come from the surrounding good cells (Greenshields, 2014). To visualize and inspect the cells with low weights `moveDynamicMesh -checkAMI` creates vtk files located in */postProcessing* showing the weight values for the AMI region which can be opened in *ParaView*.

After running `moveDynamicMesh -checkAMI` with the use of `faceType boundary`, the only fault that is mentioned are some cells with high skewness which are located at the trailing edge of the turbine where the CAD surface is very thin. There should not be much geometric errors other than this after the mesh is rotated. However, the number of faces for the source and target part of the AMI is not the same, and there are cells with low

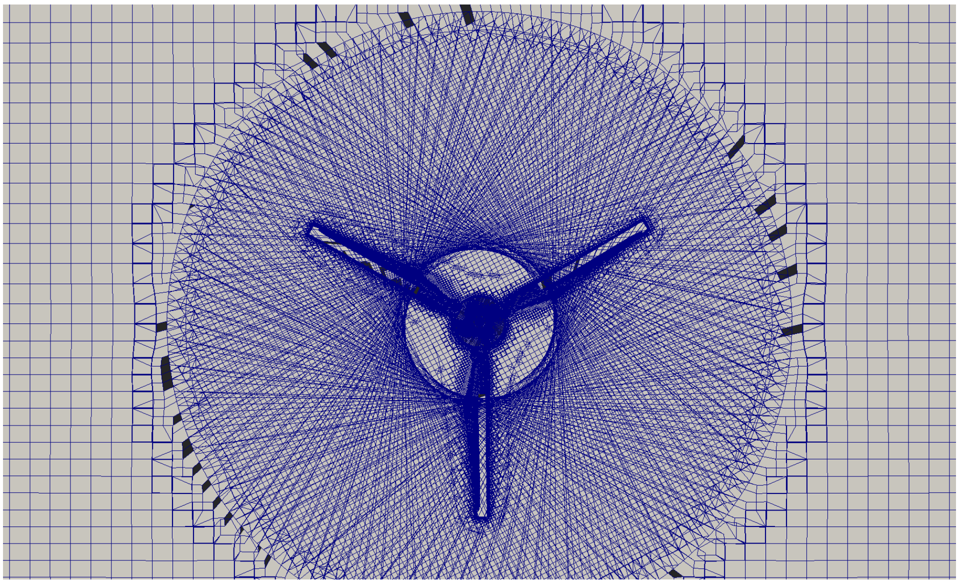


Figure 5.3: Distorted mesh during rotation when AMI region and rotating area defined in *dynamicMeshDict* does not have the same origin or if the baffle is not split.

weights along the edge of the cylinder defining the AMI.

When using the option `faceType baffles` the AMI weights are close to 1, and the number of faces for the source and target part of the AMI is exactly the same. Here, the only low weight cells are at the area where the nacelle points out of the AMI region. This is the reason for changing this option for the advanced turbine model from the Propeller tutorial. However, after running `snappyHexMesh` with `faceType baffles` the baffle for the AMI must be split using `mergeOrSplitBaffles -split -overwrite` (Fcollonv, 2012). This function detects and in this case splits the shared points of the baffle into two parts – the source and target part for the AMI. Forgetting to split the baffle may also cause distorted meshes like in figure 5.3 instead of creating a sliding interface.

To get the important and complex features of the turbine meshed properly the *resolveFeatureAngle* setting in *snappyHexMeshDict* must be set to a small enough value. Here, it is set as 30 degrees, which means that all surface feature angles that are larger than this angle will be refined. Thus, areas that are flatter and smoother can get a refinement level that is lower and have larger cells. Areas where the surface changes a lot, where it is needed more and smaller cells to capture the features well, will get a higher refinement level.

For large meshes it may be beneficial to generate the mesh and run the solver in parallel using several cores and then add the parts together after to have enough memory and for speed. This is done by adding a *decomposeParDict* file in the */system* folder (Greenshields, 2016). The decomposition of the mesh and the solution domain must be equal to the number of cores specified in the job-script for the high performance computer. The hpc job-script is found in Appendix A together with a small tutorial on how to use hpc Vilje.

5.2 Mesh refinement around airfoil

To estimate the mesh refinement needed to simulate as correct forces on the turbine as possible a study of an airfoil has been conducted. The NREL S826 airfoil from the turbine CAD model, seen in figure 5.4, is used to compare different refinement levels and background meshes used with *snappyHexMeshDict*. The tutorial *wingMotion* is used with a CAD model of the NREL S826 airfoil. Using $Re = 100\ 000$, the simulations is compared with data from Airfoiltools (2017) as well as data from Blind Test 4.

Figure 5.4: The NREL S826 airfoil represented by the .stl file from the turbine CAD model.

The mesh must be fine enough to generate the shape of the airfoil. This is accomplished by either refining the area close to the surface or by refining the background mesh as shown in figure 5.6. Due to high computational costs and time consume it is not desired to have a large background mesh, thus the refinement level of the area around the airfoil is investigated further.

The conditions at the tip of a turbine blade is hard to know exactly, therefore the airfoil test case will test the airfoil at 75% length of the blade measured from the center of the hub. Thus, the inlet velocity in the airfoil case is set to $51.75m/s$ which is the relative velocity at that location of the blade, and kinetic viscosity $\nu = 1.794e - 05$ to get a local tip Reynolds number of 100 000. The turbulence models used are k-ω SST described in section 2.2.2, and Spalart-Allmaras described in section 2.2.3. The mesh has a refinement box around the airfoil with level 2, 3 wall layers to refine the boundary layer when not using wall functions, 1 wall layer when using wall functions, and the number of cells between refinement levels are 6 cells. To see how refinement levels work, see figure 5.5. The calculation domain is about 7 chord lengths in length, 5 chord lengths in height and with only 1 cell of 0.05 m depth to create a 2D case.

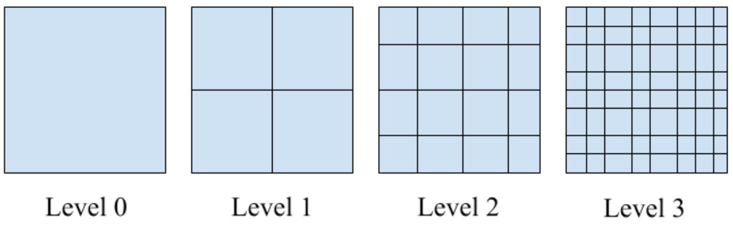
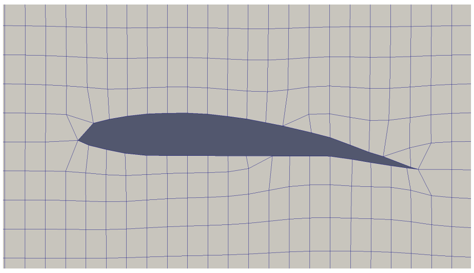
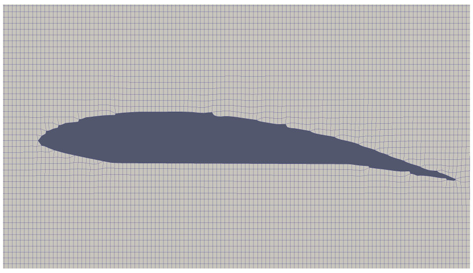
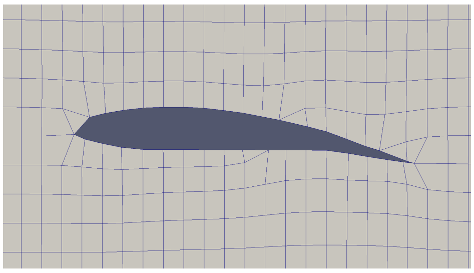


Figure 5.5: Showing how different refinement levels in *snappyHexMesh* works. The background mesh cell is divided once in each direction for level 1, and similar for the other levels.


(a) Level 2, rough background mesh.


(b) Level 2, refined background mesh.


(c) Level 1, rough background mesh.

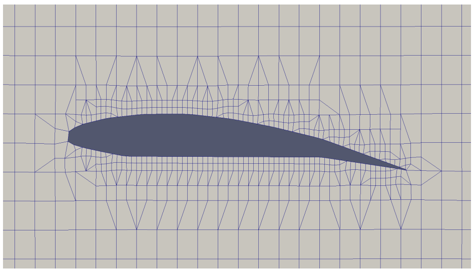

(d) Level 4, rough background mesh.

Figure 5.6: Different mesh refinement around airfoil.

The airfoil in this case is set to an angle of attack of 5 degrees; negative rotation around the z-axis. This is done by using the following command which changes the old.stl file and creates a new.stl file which has been rotated.

```
> surfaceTransformPoints -rollPitchYaw '(0 0 -5)' old.stl new.stl
```

In *controlDict* the functions *forceCoeffs* for C_L and C_D, and *yPlus* has been added. It is important to check the value of y^+ because it reveals whether the model needs wall-functions or not. According to Guerrero (2014) a $y^+ < 6$ is sufficient resolution of the boundary layer and no wall-functions are needed. In some CFD-online forums there have been suggested that the y^+ value should be even smaller to ensure that the coefficients and forces are calculated correctly without wall functions.

The solver `simpleFoam` is used to solve the steady state problem. The results of the airfoil test case without wall functions can be seen in table 5.2, showing lift and drag coefficients for two different turbulence models. Table 5.1 shows the results from the simulations ran with wall-functions. *Level* is the refinement level in *snappyHexMeshDict* and *Cover* shows how many background mesh cells covers the airfoil chord length.

The drag force is usually hard to predict because it is so dependent on the boundary layer. If the mesh is not refined enough in the boundary layer region, the calculations of the drag will be wrong. The lift force is not as dependent on the boundary layer so it is usually easier to predict.

Wall-functions

Level	Cover	Turbulence model					
		Spalart Allmaras			k-ω SST		
		C$_D$	C$_L$	y^+_{min}/y^+_{avg}	C$_D$	C$_L$	y^+_{min}/y^+_{avg}
3*	2	0.07402	0.6873	19.7 / 133.4	0.0661	0.3973	25.1 / 111.5
3	4	0.06351	0.8985	19.9 / 79.5	0.0618	0.5846	8.2 / 47.6
3	8	0.05804	0.9355	3.7 / 31.6	0.0447	0.6985	2.2 / 24.4
3	16	0.06233	0.9185	4.3 / 22.7	0.0422	0.7159	0.7 / 17.5
4*	4	**0.05384**	**0.9443**	9.8 / 43.9	0.0628	0.4495	7.9 / 31.4
4	8	0.05294	0.8972	1.5 / 18.1	**0.0353**	**0.7532**	0.5 / 7.9
4*	8	0.05078	0.9051	6.1 / 23.8	0.0334	0.7444	1.7 / 18.4
		C$_D$			C$_L$		
Airfoiltools		0.02721			1.0938		
Blind Test 4		0.02780			1.0783		

Table 5.1: Calculated force coefficients for a NREL S826 airfoil with 5 degrees angle of attack. Wall-functions are used and the Reynolds number is 100 000.* 0 layers used.

Resolved boundary layer

Level	Cover	Turbulence model					
		Spalart Allmaras			k-ω SST		
		C$_D$	C$_L$	y^+_{min}/y^+_{avg}	C$_D$	C$_L$	y^+_{min}/y^+_{avg}
3	8	0.05769	0.9364	8.6 / 20.9	0.0435	0.9650	7.6 / 20.3
4	4	**0.05816**	**0.9654**	3.4 / 34.8	0.0437	0.9856	4.9 / 33.0
4*	8	0.05294	0.8972	1.5 / 18.0	**0.0406**	**0.9901**	1.1 / 18.3
4	8	0.05477	0.9065	2.3 / 11.7	0.0428	0.9713	2.0 / 12.8
5	8	0.05730	0.8789	1.3 / 6.4	0.0479	0.8715	1.6 / 7.4
5	10	0.05616	0.8831	2.0 / 6.2	0.0429	0.7951	0.2 / 5.9
		C$_D$			C$_L$		
Airfoiltools		0.02721			1.0938		
Blind Test 4		0.02780			1.0783		

Table 5.2: Calculated force coefficients for a NREL S826 airfoil with 5 degrees angle of attack. Boundary layer is fully resolved and the Reynolds number is 100 000. * 1 layer at the wall.

The best results for each turbulence model are marked in bold in table 5.1 and 5.2. Several meshes has been tested. The lift coefficient estimated is lower, best at 0.9901 compared to 1.0783, and the drag coefficient estimated is higher, best at 0.0334 compared to 0.02780. The CL/CD relationship is also a bit off compared to the measurements from Blind Test 4 shown in figure 5.9.

Some of this error seems to comes from when the CAD model of the turbine blade is exported as a .stl file and imported to OpenFOAM it is not as smooth as it should have been. An comparison of the NREL S826 airfoil from Airfoiltools (2017) and the airfoil made with the best result from OpenFOAM is shown in figure 5.7 and 5.8 respectively. The OpenFOAM airfoil is thinner and the bottom side of the airfoil has different curvature.

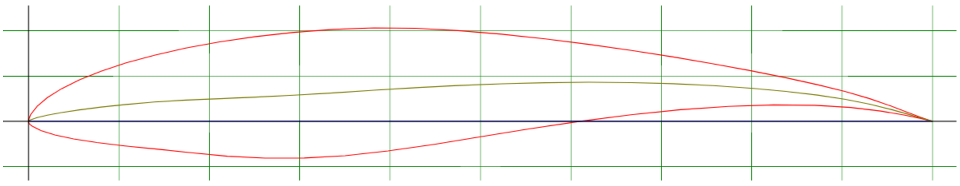


Figure 5.7: Airfoil from AirfoilTools.com of NREL S826. Showing smooth edges.

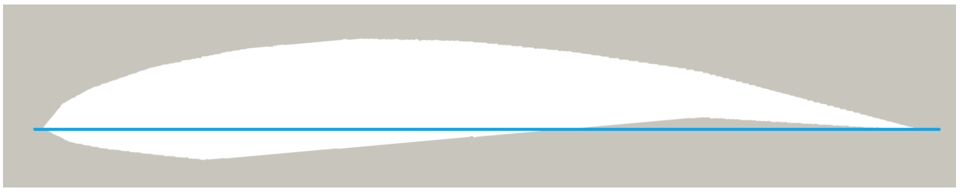


Figure 5.8: Airfoil from best OpenFOAM simulation. Showing slightly different shape.

This is a problem encountered by others before. The geometry is not always implemented as smoothly as desired when using `snappyHexMesh`. This was pointed out by Colley (2012) where he got jagged edges of his turbine blades that he could not get rid of.

However, the results of this airfoil case study are deemed good enough. Thus, the overall best results achieved here with boundary layer resolved with 8 cells covering the chord length, 4 levels of refinement around the airfoil, only 1 layer at the wall and the k-ω SST turbulence model is used for further simulations of the advanced wind turbine model.

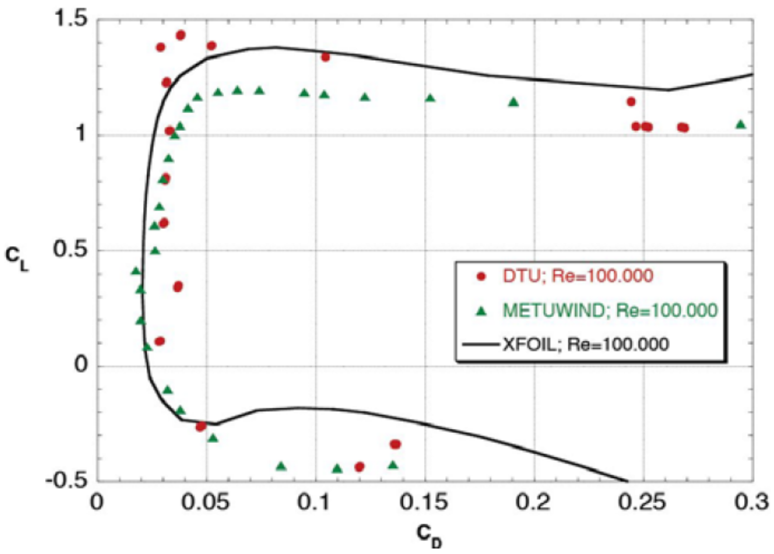


Figure 5.9: CL vs CD measured on the NREL S826 in Blind Test 4 by Sœtran and Bartl (2015)

CHAPTER 6 RESULTS AND DISCUSSION

For each of the two models; *actuatorDiskExplicitForce* and the advanced turbine model, the procedure of the simulations are explained first, then the results are discussed. Finally, the two models are compared with pros and cons.

6.1 *actuatorDiskExplicitForce* simulations

The actuator disk model from the project thesis by Hoem (2017), based on the work of Svenning (2010), is altered as described in Chapter 3. The actuator disk has now an impermeable hub at the center. To compare the new actuator disk model to the old model and the wind tunnel measurements done in Blind Test 4 (Bartl and Sœtran, 2017) plots showing velocity profile and turbulent kinetic energy in the wake are made.

Simulations are done using both k-ϵ turbulence model, described in section 2.2.1, and k-ω SST turbulence model, described in 2.2.2. Three different meshes have been compared before the best mesh is chosen. They are described in table 6.1.

		M1 Medium	M2 Coarse	M3 Fine
Number of cells:		351 346	46 407	3 969 344
kEpsilon y^+	hub	35.87	158.98	25.04
	wall	293.38	279.34	129.64
kOmegaSST y^+	hub	28.95	40.81	5.62
	wall	297.85	282.88	127.89
wallFunction:	hub	on	on	off
	wall	on	on	on
Valid:		no	yes	yes*
Thrust force:		**66.37 N**	**2.11 N**	**45.90 N**

Table 6.1: Details of the three compared meshes. * Valid for k-ω SST model.

First discovery made while testing different meshes is that the thrust and torque added to the actuator disk region depends on the mesh. Theese values are also very sensitive to the meshing. An example is the medium mesh, M1, where a background mesh of about 32 000 cells gives a thrust value of 35N, while a background mesh of about 35 000 cells gives a thrust value of 65N. The value of the added thrust is intended to be 43.87N. The

y^+ values for mesh M1 is indicating that wall-functions should not be used. Thus, a coarser mesh, M2, where the wall-functions can be used is made. The y^+ values are good, however, the added thrust is way too low at only 2.11N. It is clear that the coarse mesh with wall-functions cannot produce an accurate evaluation of wind turbine performance.

Finally, the fine mesh, M3, which does not use wall-functions in the actuator disk region, but is refined enough to generate small y^+ values, gives a thrust force of 45.90 N close to the intended value. Due to having the best added forces by far, mesh M3 is the mesh used in further analysis.

The simulations turbulence models use the numerical values calculated from the equations stated in section 2.2.1 for k-ϵ, and section 2.2.2 for k-ω SST. The numerical values are listed in table 6.2.

		k	epsilon	nut	nuTilda	omega
kEpsilon	internalField	0.4959	4.905	0.0045	-	-
	(-)WallFunction	kqR	epsilon	nutk	-	-
kOmegaSST	internalField	0.4959	-	0.0045	0.0045	60
	(-)WallFunction	kqR	-	nutk	-	omega

Table 6.2: Numerical values and wall-functions used on the walls in the turbulence models used in the actuator disk model. The hub boundary is set to *zeroGradient*.

At the bottom of *controlDict* the function *singleGraph* is added two times to sample data along the z-axis just after the actuator disk at x/D = 0.063, and in the wake at x/D = 2.77. Both velocity and turbulent kinetic energy are sampled.

The velocity and turbulent kinetic energy sampled in the wake are used to create figure 6.1 to 6.4. Here, the wakes of the original actuator disk model made in the project thesis by Hoem (2017) is included to see how the model is improved by including a hub. Two different turbulence models are used because the k-ϵ was used in the previous work and is very commonly used, but k-ω SST is well known for being a better choice when it comes to turbo machinery (Guerrero, 2014). Blind Test 4 measurements are also included.

The velocity profile in the wake is highly improved after introducing the hub in the *actuatorDiskExplicitForce* model. When using k-ϵ turbulence model the velocity towards the walls are in the right range for all three meshes, but the W-shape of the velocity profile after the hub is only reproduced by the medium mesh, M1. For M1 the shape is good, but the velocity deficit is too high, the flow is too slow in the wake. The coarse mesh, M2, with too low thrust force shows no W-shape of the velocity profile. The velocity in the wake is too high and the low added force influences only a small part of the wake flow. The fine mesh, M3, which has the best thrust force assigned recreates the velocity close to the walls well, but not in the wake of the hub. However, this is the wake velocity profile with the closest average velocity to the measurements.

When using k-ω SST turbulence model the velocities towards the walls are again in the right range. The big difference here is that both mesh M1 and M3 are recreating the W-shape in the wake after the hub. They also have a wider velocity peak after the hub than for the k-ϵ case. However, the velocity deficit is a little large. Again, the coarse mesh, M2, fails to reproduce the wake.

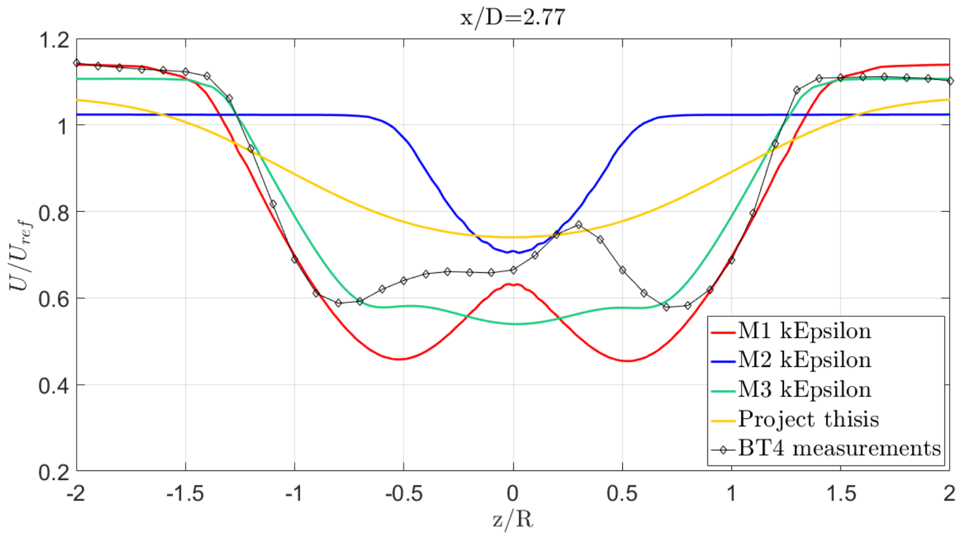


Figure 6.1: Comparing meshes using the k-ϵ turbulence model. Normalized streamwise velocity along normalized depth of wind tunnel in the wake of the actuator disk model using results from project thesis by Hoem (2017), medium mesh M1, coarse mesh M2, fine mesh M3, and measurements from Blind Test 4.

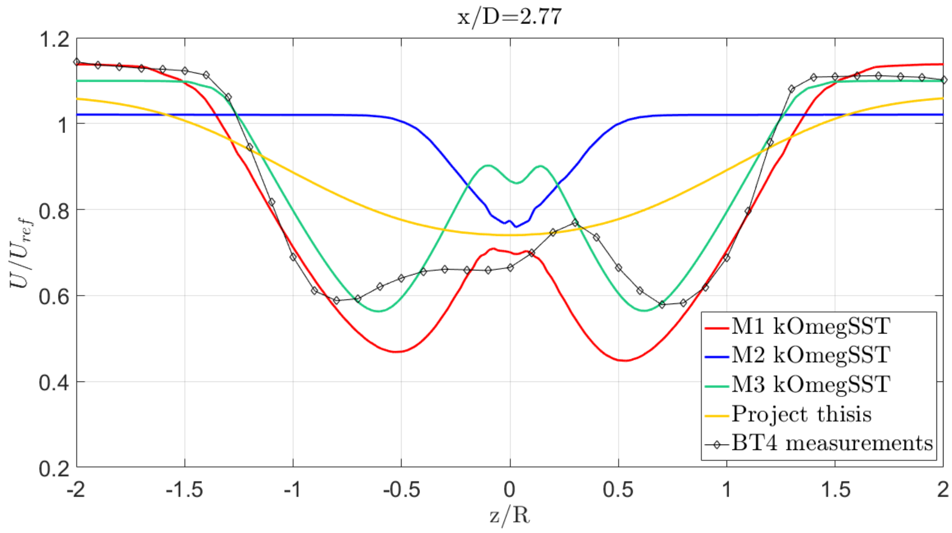


Figure 6.2: Comparing meshes using the k-ω SST turbulence model. Normalized streamwise velocity along normalized depth of wind tunnel in the wake of the actuator disk model using results from project thesis by Hoem (2017), medium mesh M1, coarse mesh M2, fine mesh M3, and measurements from Blind Test 4.

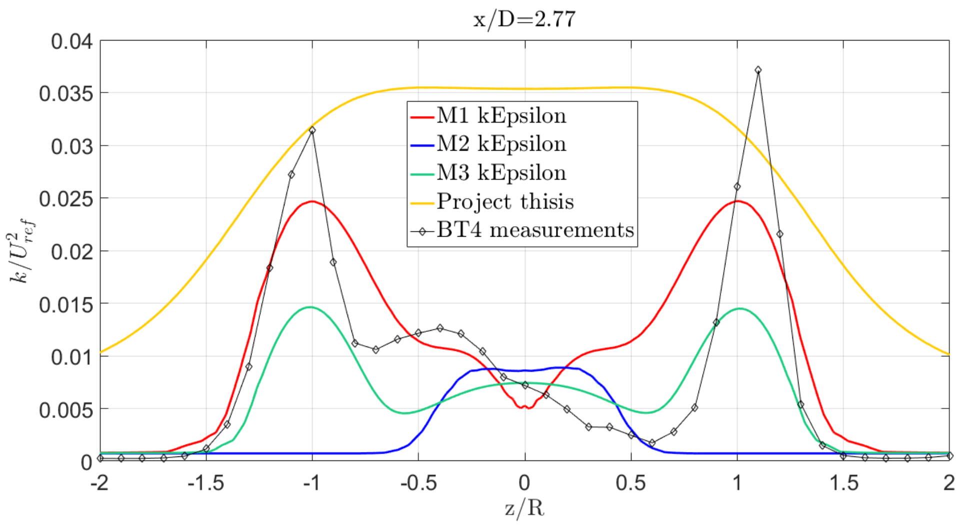


Figure 6.3: Comparing meshes using the k-ϵ turbulence model. Normalized turbulent kinetic energy along normalized depth of wind tunnel in the wake of the actuator disk model using results from project thesis by Hoem (2017), medium mesh M1, coarse mesh M2, fine mesh M3, and measurements from Blind Test 4.

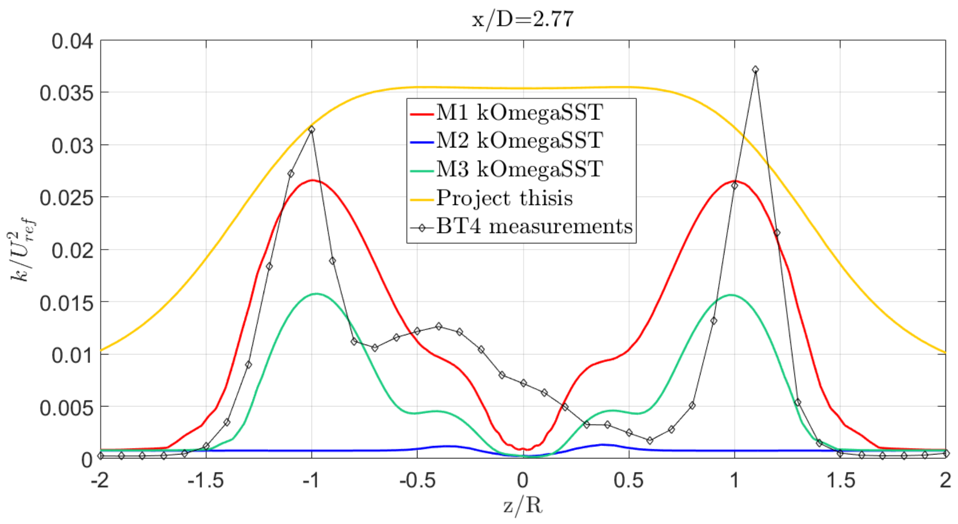


Figure 6.4: Comparing meshes using the k-ω SST turbulence model. Normalized turbulent kinetic energy along normalized depth of wind tunnel in the wake of the actuator disk model using results from project thesis by Hoem (2017), medium mesh M1, coarse mesh M2, fine mesh M3, and measurements from Blind Test 4. .

According to Bartl and Sœtran (2017) the asymmetry in the center of the wake is caused by the wake of the tower. The tower is not included in the simple actuator disk model, and thus the asymmetry fails to be predicted. This contradicts what van der Auweraert (2015) concluded in his study, that the tower would only affect the near wake.

When predicting the velocity profile in the wake the k-ω SST turbulence model performs better than the k-ϵ turbulence model, especially in the wake of the hub. A contributing factor to this is that the y$^+$ values for the k-ϵ turbulence model case is not as low as they should have been. These values are better for the k-ω case. This corresponds with theory suggesting that k-ω SST is a better choice is cases like a wind turbine.

Also, when it comes to the turbulent kinetic energy in the wake the results are a lot better than the previous model. When using k-ϵ turbulence model both the medium and fine mesh recreates the M-shape similar to the measurements, although the peaks are lower. Mesh M1 with too high thrust recreates the wake of the hub slightly better than mesh M3 where the turbulent kinetic energy increases a little in the wake of the hub. The simulations cannot reproduce the same level of turbulent kinetic energy in the wake as the measurements. However, the peaks are located at the same place at the end of the actuator disk regions in the wake. The coarse mesh, M2, fails to predict the turbulent kinetic energy in the wake and has a similar profile shape as the actuator disk model without a hub, but with much lower values. Again, the low levels of added force effects the flow only a little.

When using k-ω SST turbulence model all three meshes somewhat recreates the M-shape of the measured turbulent kinetic energy profile. The values close to the wall are well recreated. The highest peaks are again for mesh M1 where the added force is too high. However, now, the fine mesh, M3, also recreates the wake of the hub fairly well.

Altogether, the level of turbulent kinetic energy is higher and closer to the measured values when using k-ω turbulence model. The profiles have higher highs and lower lows. Again, the coarse mesh, M2, fails to predict the turbulent kinetic energy in the wake, although the shape of the profile is a tiny bit better, the values are way too low.

From the Reynolds averaged Navier Stokes equation the Reynolds stresses are unknown. The Boussinesq's approximation from equation (2.6) is used for both turbulence models, and this estimation of the Reynolds stresses are included in the source code of the turbulence models. To get the data of the estimated $\overline{u_i u_j}$ from the simulations, the following commands are used.

```
> simpleFoam -postProcess -func R -latestTime
> postProcess -func mySampleR -latestTime
```

where *mySampleR* is located in */system* and dictates the sampling of the stresses. The symmetric tensor $\overline{u_i u_j}$ (named *functionObject R* in OpenFOAM) is written to file with 6 elements; $\overline{u_x u_x}$, $\overline{u_x u_y}$, $\overline{u_x u_z}$, $\overline{u_y u_y}$, $\overline{u_y u_z}$ and $\overline{u_z u_z}$. These values can be compared with measured results done by Eriksen (2016).

The low peaks of the turbulent kinetic energy corresponds with the values of the normal stresses in figure 6.5. The simulations are lacking the asymmetric effects in the kinetic turbulent energy field, which again is suggested to be caused by neglecting the tower, but the values towards the walls are very similar. The peaks are quite similar for the two different turbulence models, where the k-ω SST case has slightly higher values. The difference between the two models is larger towards the center of the actuator disk wake.

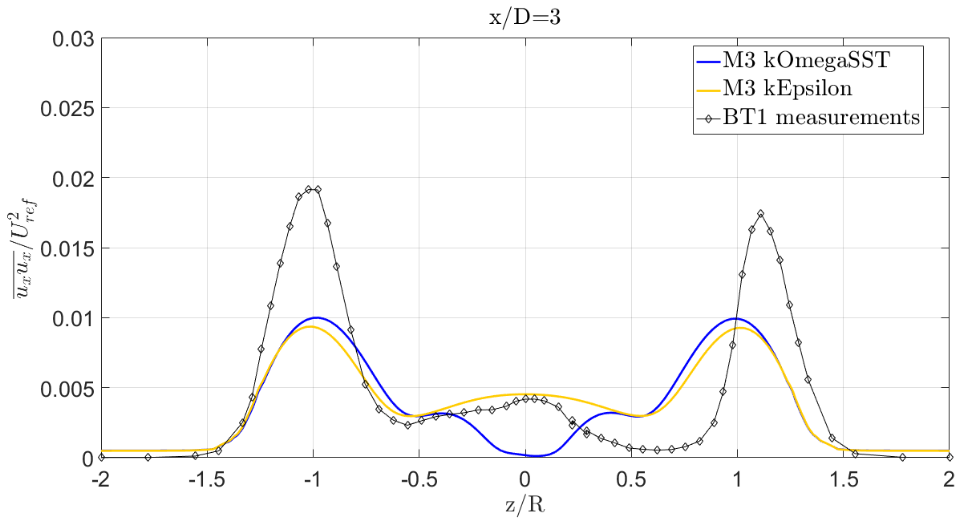


Figure 6.5: Normalized normal stresses along normalized depth of wind tunnel in the wake of the actuator disk explicit force model with hub using mesh M3 with k-ϵ turbulence model, k-ω SST turbulence model and measurements from Blind Test 1 (Eriksen, 2016).

Here, the k-ϵ case has higher levels of normal stresses, while the value in the center of the hub wake for the k-ω SST model is close to zero.

The second sample just behind the actuator disk is used for calculating the thrust coefficient and power coefficient. According to actuator disk theory in section 2.3.3 the thrust and power coefficient can be calculated by a simple expression depending on the axial induction factor a, which depends on the velocities at the disk.

Since the velocity profiles are not uniform across the disk, see figure 6.6, the weighted average is calculated. Velocities further from the hub contributes more to the weighted average due to larger radial distance. Since the forces in the *actuatorDiskExplicitForce* model is added as a volume force the samples are taken after the whole disk volume so that all the forces contributes to the field. Thus, a , C$_T$ and C$_P$ is calculated as following:

$$a = \frac{U_\infty - \bar{U}_d}{U_\infty} \ , \qquad C_T = 4a(1-a) \ , \qquad C_P = 4a(1-a)^2.$$

When calculating the thrust coefficient the weighted averaged velocity, $\bar{U}_d$, is made up of only velocity in x-direction, u, since the thrust only acts in x-direction. When calculating the power coefficient the weighted averaged velocity is made from the total velocity field, U. The resulting coefficients and errors are shown in table 6.3

Table 6.3 shows that the actuator disk model is overestimating the power coefficient and underestimating the thrust coefficient. These results corresponds with the results found by Nodeland (2013) who used a actuator line model which predicted the thrust well, but overestimated the power; Kalvig et al. (2014) did a comparing study with an actuator disk model, an actuator line model and a fully resolved model which all overestimated the power and underestimated the thrust coefficient; Tossas and Leonardi (2012) made an

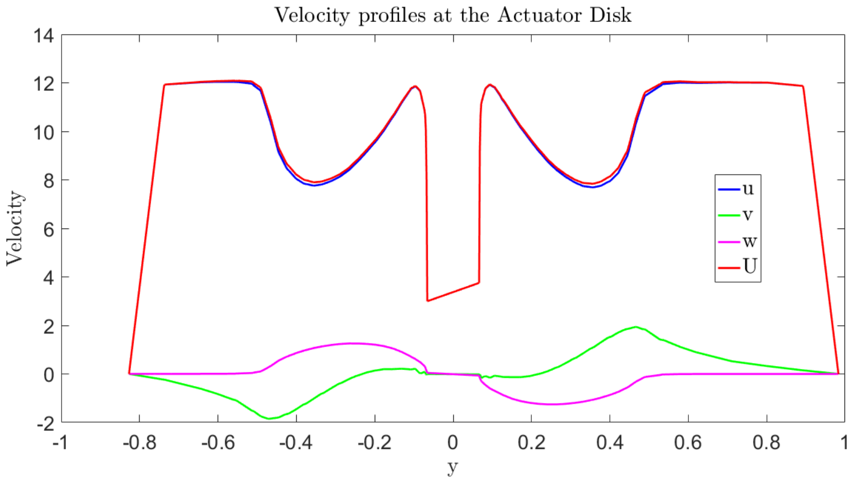


Figure 6.6: Velocity profiles just after the actuator disk. $U = \sqrt{u^2 + v^2 + w^2}$.

	a_T	C_T	Error	a_P	C_P	Error
ADEF kEpsilon	0.2583	0.7664	-5.50 %	0.2356	0.5507	19.20 %
ADEF kOmegaSST	0.2615	0.7725	-4.75 %	0.2377	0.5525	19.59 %
Blind Test 4		0.811			0.462	

Table 6.3: Calculated coefficients and errors of the actuator disk model and the measured values from Blind Test 4 (Bartl and Sœtran, 2017).

actuator disk model and an actuator line model which also overestimated the power and underestimated the thrust coefficient. It seems like this is a common result and might suggest that the actuator disk models does not take into account all the losses of the real turbine. However, most of the models can recreate the streamwise velocity field in the wake quite well. This also applies for the *actuatorDiskExplicitForce* model.

The two turbulence models are giving quite similar results for the coefficients in table 6.3 for the fine mesh, M3. The k-ω SST model performs slightly better, and is the only fully valid case, see table 6.1.

Some of the error of the coefficients may come from the fact that actuator disk theory defines a disk area, whilst the actuator disk modeled here has a thickness of $0.04m$ and thus becomes a volume. Thus, if the mesh is not well resolved, the volume of the actuator disk, i.e. the volume where the force is added is not properly represented. Together with discovering that the forces added varies a lot for meshes with similar sizes, yields the conclusion that the *actuatorDiskExplicitForce* model is mesh dependent. The forces are also applied in a nonuniform way across the disk which makes the calculation of the coefficients dependent on an estimation of the induction factor a. There are no external losses included in the calculations either.

Figure 6.7 shows how the flow is influenced by the actuator disk; how the streamlines are bending and rotating in the wake.

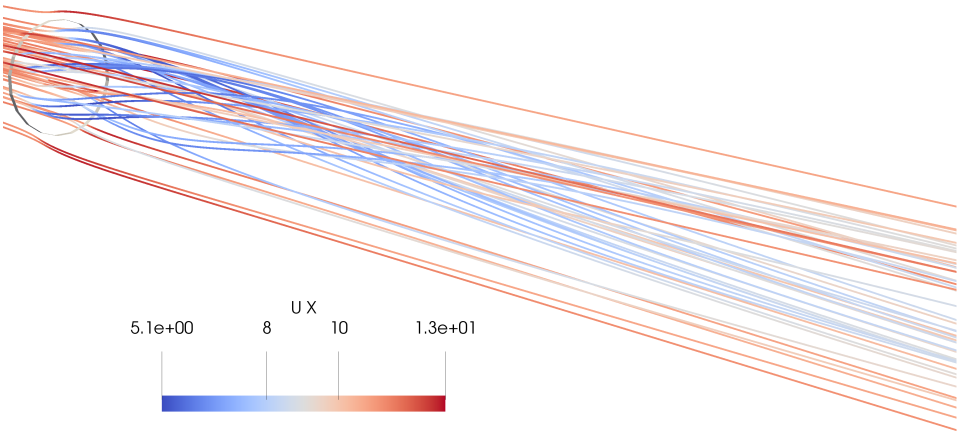


Figure 6.7: Streamlines showing the Actuator disk's influence on the flow. Colored by the stream-wise velocity component u. The thin, grey circle is the outer radius of the Actuator Disk.

6.2 Advanced wind turbine simulations

The advanced wind turbine model is a 3D transient model made up of the CAD model of the turbine and a refined mesh made from *blockMesh* and *snappyHexMesh* using AMI. The ideal level of refinement has been studied in the airfoil test case in section 5.2. However, due to a large amount of time spent on creating the model, there has been limited time left for running large simulations. Thus, the results provided in this discussion is made using a coarser mesh than initially intended, and they are not ran for as long time as desired.

The airfoil test case suggested that a background mesh of 8 cells covering the airfoil chord length was to be used. This creates a very large mesh when it is transferred to case including the whole wind tunnel. Thus, the background mesh is made 8 times smaller in all directions, which is made up for by having 3 higher levels of refinement around the turbine and in the wake. This means that the suggested level 4 around the airfoil becomes level 7 for the turbine, and the suggested level 2 in the wake becomes level 5. The mesh created has some cells with relatively high skewness located at the tail edge of the blades reported by `checkMesh`. The mesh contains about 20 million cells. The coarser mesh used for testing contains about 6 million cells. The big difference is the highest level of refinement for the turbine blades is slightly lower making the refinement of the trailing edge coarser, and the level of refinement of the wake which has been made coarser.

For the advanced turbine model the mesh around the turbine blades are rotating, and the rotational speed of the domain is set in *dynamicMeshDict* to 146.18 rad/s with negative x-axis as rotational axis. The mesh outside the AMI is stationary.

The simulations are computed at *Vilje* which is a high performance computer at NTNU, see `www.hpc.ntnu.no/display/hpc/Vilje` for more information. The simulations are done with $\mathrm{Re}_{tip} = 100\ 000$ and k-ω SST turbulence model as described in 2.2.2, in accordance with the results from the airfoil test case. The values and definitions used for the turbulence model is shown in table 6.4.

		k	nut	omega
kOmegaSST	internalField	0.4959	0.0045	60
	(-)WallFunction	kqR	nutk	omega

Table 6.4: Numerical values and wall-functions used in the k-ω SST turbulence model for the advanced turbine model.

It turns out that there are reported problems with using some function objects such as *singleGraph* together with an AMI case (Lloyd, 2017). However, there is a work around method; creating surfaces at the location in the wake that is of interest. The surfaces are saved as vtk files and can be opened in ParaView. Inside ParaView the data of the surface can be stored in a csv file which can be processed in MATLAB to create the graphs wanted. It takes a lot more time to do, but it works. Thus, the surface used to sample data along the z-axis in the wake is placed at x/D = 2.77 and written to file for every time step of interest covering one whole rotation. Both velocity and turbulent kinetic energy are sampled in this way. The time averaging of the fields are made with samples collected at every 12 degrees rotation throughout one whole rotation resulting in 30 samples. This number of samples is chosen because of the manual work that must be done for post processing, and thus too many samples will take too much time. The velocity and turbulent kinetic energy data collected from the simulations are used for comparing the results of the advanced turbine model to the actuator disk model and the Blind Test 4 measurements. However, the Reynolds stresses $\overline{u_i u_j}$ cannot be sampled as easily from the surfaces, thus this comparing graph has been excluded for now.

After the computations are done at the super computer Vilje the case files are very large and may not be possible to post process on the smaller laptop at hand. Luckily, there is a function named `mapFields`. This function takes a source case and maps the solution of this case on a target case. Thus, the solution of the large case on Vilje may be mapped onto a smaller case for post processing. In the source file *mapFields.C* that comes with OpenFOAM, the process is describes as mapping volume fields between two meshes, reading and interpolating all fields in the time directory chosen. Since the smaller case is made equal to the large case with the same boundaries and geometries and only having different mesh density, the mapping can be done using the option `-consistent`. To map the result of the final time output of the large case onto the small case, the starting time for the small case defined in *controlDict* must be equal to the final time in the large case. When the two case folders are located in the same outer folder, the following code line can be used from inside the target case, i.e. the small case.

```
> mapFields ../largeCase - consistent
```

Thus, the small case gets the corresponding time folder as the result of the large case, and the result can now be viewed as normal in ParaView. Note that the small case do not run further after mapping, so the mapping must be done of the final result. However, the mapping can be done for several time steps and thus a time series for the small case can be created.

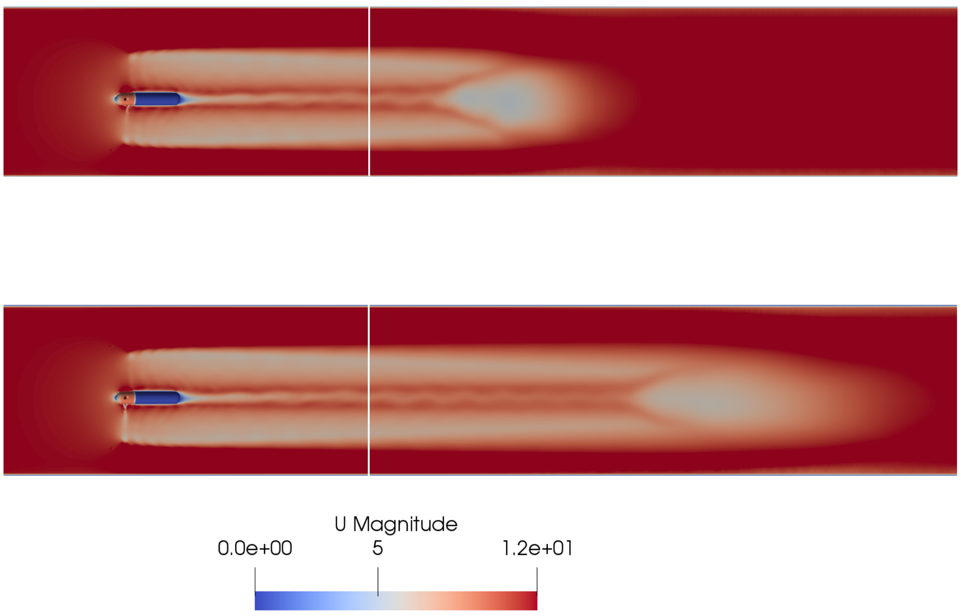


Figure 6.8: Upper: Streamwise velocity field after 0.5s. Lower: Streamwise velocity field after 0.8s. The white line shows where the wake samples are collected parallel to the z-axis.

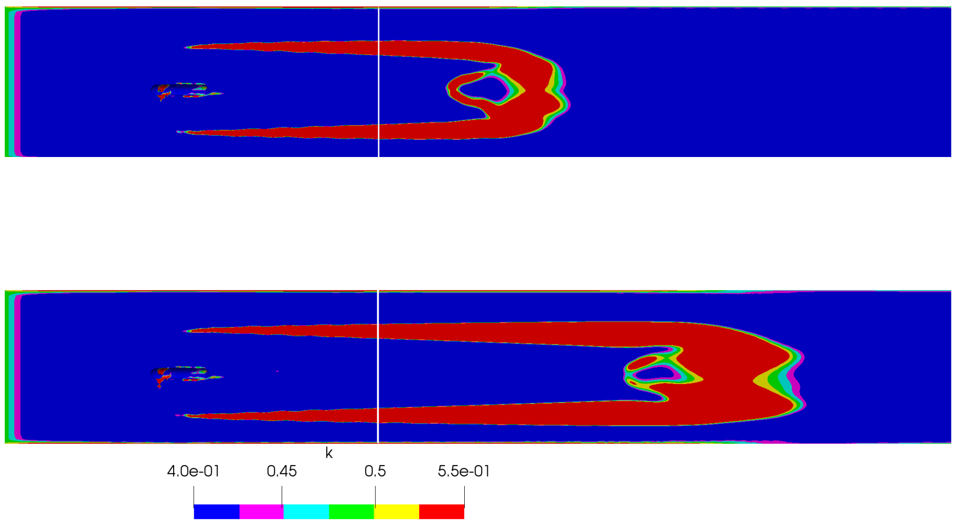


Figure 6.9: Upper: Turbulent kinetic energy field after 0.5s. Lower: Turbulent kinetic energy field after 0.8s. The white line shows where the wake samples are collected parallel to the z-axis.

The velocity field and turbulent kinetic energy field are not fully developed in the simulations discussed here. Figure 6.8 shows the velocity field in streamwise direction at 0.5s and 0.8s. The same is shown for the turbulent kinetic energy in figure 6.9, where it also can be seen that the calculated initial value of k is a little high. For the whole tunnel length to be influenced by velocities from the turbine the simulation time is about 1 second, and it takes longer to fully develop the turbulent field. This takes several days of real computational time. Thus, the values sampled for wake velocity and turbulent kinetic energy fields are estimates of how they will turn out for the model over time.

Even though the simulations are done with a coarser mesh and for shorter time than desired, some tendencies can be seen. First of all, the residuals of calculated fields are nice and low, so there is no indication that the model will crash even for longer simulation times. With the surfaces in the wake at $x/D = 2.77$ giving field samples for every 12th degree rotation, time average fields for one whole rotation can be made. In figure 6.10 the time averaged normalized velocity field for rotation starting at 0.5s and 0.8s are shown together with the measured data from Blind Test 4. Again, the exclusion of the wake results in a symmetric wake profile. The velocities towards the wall and at the outer radius of the area swept by the turbine blades are recreated well for both time intervals. The W-shape of the profile is recreated, and the wideness of the wake is good. However, half way down the turbine blades the recreation of the wake behind yields too high velocities. The two time intervals give almost identical results.

In figure 6.11 the time averaged normalized turbulent kinetic energy field for rotation starting at 0.5s and 0.8s are shown together with measured data from Blind Test 4. The M-shape of the profile is somewhat recreated, but the peaks are too low compared with the measured data. The peaks are at the correct location at the outer edge of the area swept by the turbine blades. There is turbulent kinetic energy present in the wake of the center of the hub as well. Values towards the wall are reproduced well. It looks like the turbulent kinetic energy field takes longer to develop than the velocity field. Again, the two time intervals give almost identical results.

To estimate the performance of the turbine the power coefficient, C_P, is calculated. This is done by adding the function object *forces* to *controlDict* and using the relations in (6.1). The *forces* output file gives values for forces and moments, both divided into pressure, viscous and porous contributions, which again is divided into x-, y- and z-direction. Here, use $CoR = (0.016 \;\; 0 \;\; 0)$ for the moment calculations. This is center of the area where the blades are attached to the hub, where the blades are rotating around.

$$\text{Power} = \sum \text{M}_x \cdot \omega_x \;, \qquad \text{C}_p = \frac{\text{Power}}{\frac{1}{2}\rho A_{swept} U_\infty^3} \tag{6.1}$$

where ω is the rotational speed of the turbine, U_∞ is the inlet velocity and A_{swept} is the swept area of the turbine blades.

The thrust coefficient, C_T, is calculated by adding together the forces in flow direction from *forces* and normalizing, see relations in (6.2).

$$\text{Thrust} = \sum \text{F}_x \;, \qquad \text{C}_T = \frac{\text{Thrust}}{\frac{1}{2}\rho A_{swept} U_\infty^2} \tag{6.2}$$

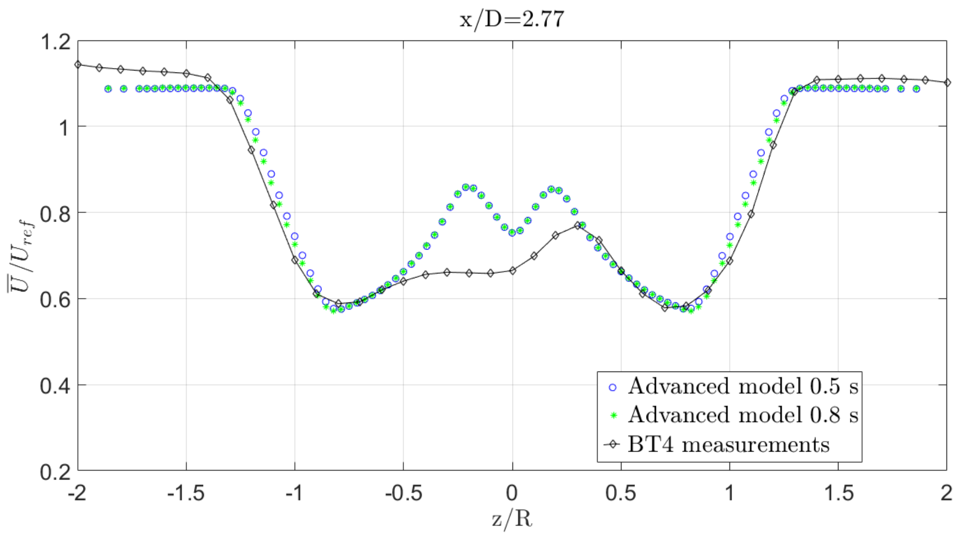


Figure 6.10: Normalized streamwise velocity along normalized depth of wind tunnel in the wake of the coarse advanced turbine model, and measurements from Blind Test 4. The profiles are time averages of one full rotation starting at 0.5 s and 0.8 s for the advanced turbine model.

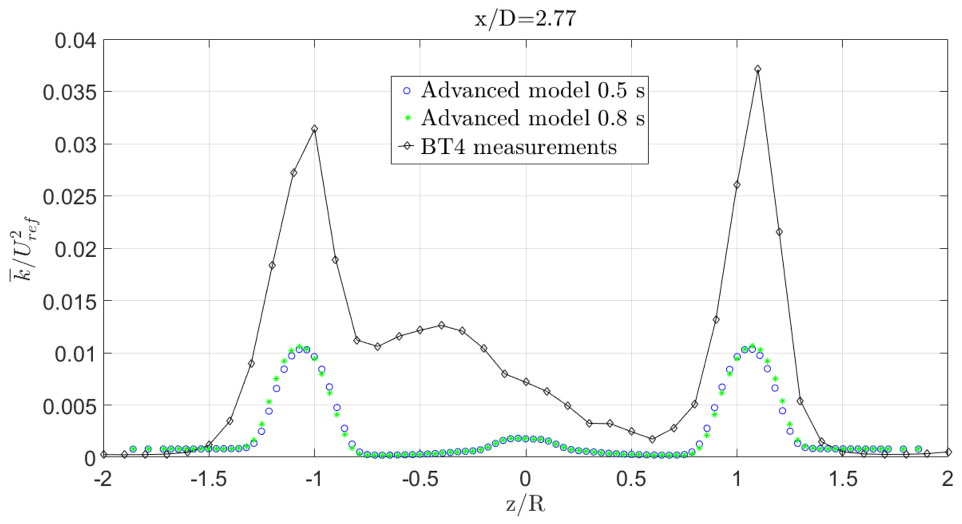


Figure 6.11: Normalized turbulent kinetic energy along normalized depth of wind tunnel in the wake of the coarse advanced turbine model, and measurements from Blind Test 4. The profiles are time averages of one full rotation starting at 0.5 s and 0.8 s for the advanced turbine model.

The results of the coefficients in table 6.5 from the advanced turbulence model simulations look promising. The coefficients are time averages for one rotation starting at 0.8s. The thrust coefficient is slightly underestimated, but an error of -1.18 % is quite satisfactory considering that this is the value for the coarser mesh for the advanced turbine model. The power coefficient is also underestimated with an error of 19.27 %. Since these calculations depend on values from the turbine surface which is not as well resolved as desired, the results are not as good as desired either. The simulations should be done with a finer mesh and run for a much longer time. Again, this was done as a compromise for computational time, too see that the model works as intended.

	C_T	Error	C_P	Error
Advanced Turbulence model	0.801	-1.18 %	0.373	-19.27 %
Blind Test 4	0.811		0.462	

Table 6.5: Calculated coefficients and errors of the advanced turbine model and the measured values from Blind Test 4 (Bartl and Sœtran, 2017).

Figure 6.12 shows how the flow is influenced by the turbine illustrated by streamlines colored with streamwise velocity component. Figure 6.13 - 6.17 shows how the mesh rotates counter-clockwise with a sliding interface and no mesh distortion. And figure 6.18 shows how the velocities increase with the radius of the blades, where the velocities at the tip of the blades are near 70 m/s.

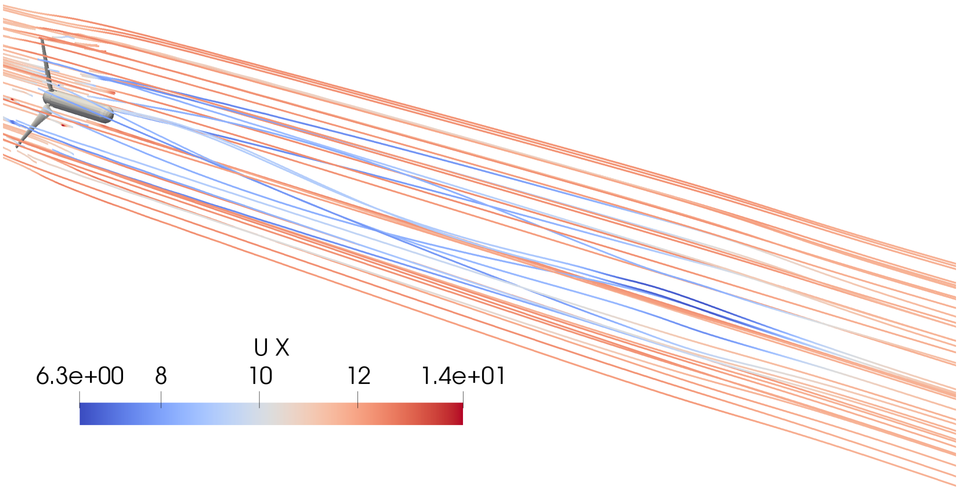


Figure 6.12: Streamlines showing how the turbine influences the flow. Colored by the streamwise velocity component u.

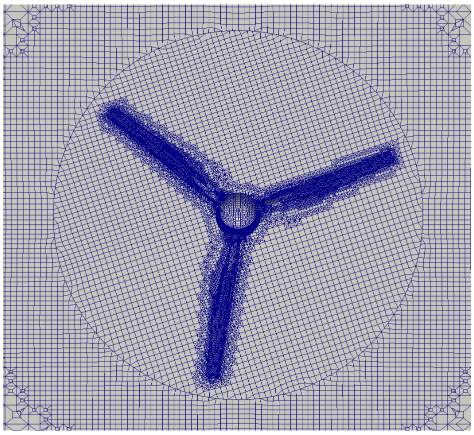


Figure 6.13: Mesh at 0^o rotation.

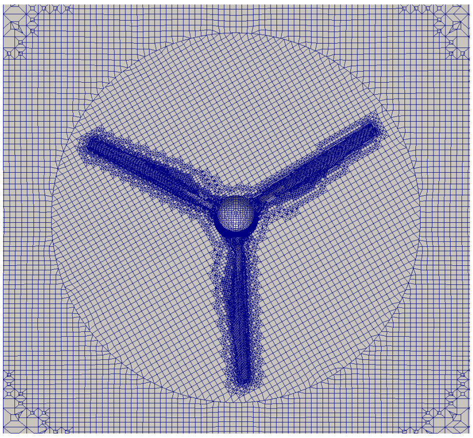


Figure 6.14: Mesh at 12^o rotation.

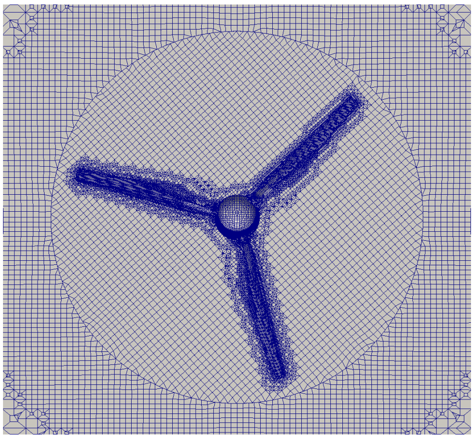


Figure 6.15: Mesh at 24^o rotation.

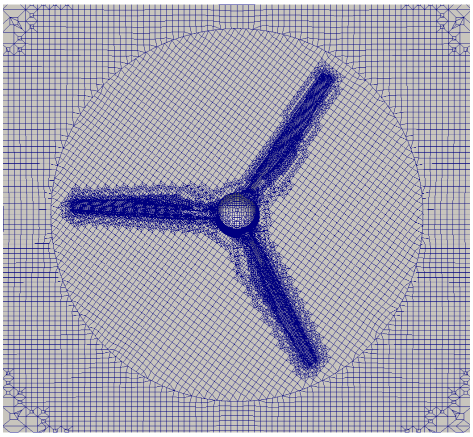


Figure 6.16: Mesh at 36^o rotation.

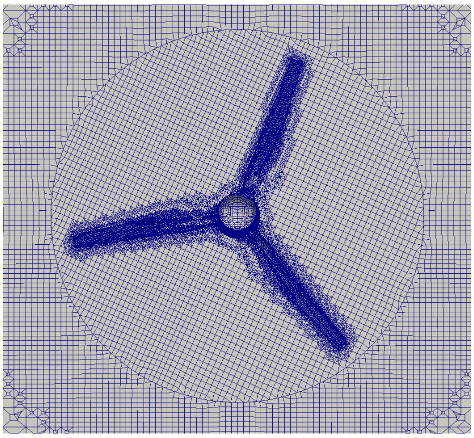


Figure 6.17: Mesh at 48^o rotation.

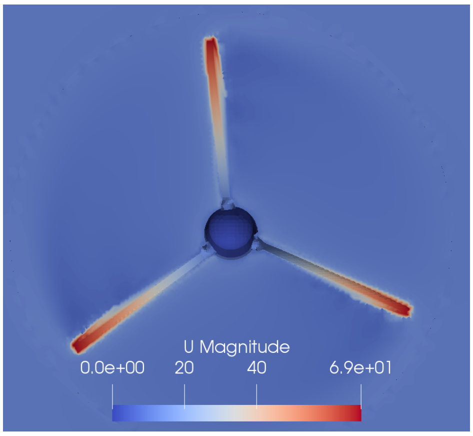


Figure 6.18: Velocity field at the turbine blades.

6.3 Comparing the two wind turbine models

ActuatorDiskExplicitForce model with hub and the advanced turbine model are two very different models for the same wind turbine. The results for the two models are also quite different. However, there are some pros and cons with both the models, see table 6.6.

Model:	Actuator Disk (figure 6.19)	Advanced Turbine (figure 6.20)
	Easy to set up	Good estimation of performance
Pros	Fast	Recreates real wake quite well
	Provides initial estimate of wake	3D transient model
	Inaccurate power estimation	Hard to set up correctly
Cons	Mesh dependent	Demands high computer power
	Steady state model	Slow simulation and post processing

Table 6.6: Pros and cons for the two wind turbine models.

This may be summed up to the fact that if the intention is to estimate the behaviour of a wind turbine and the interest is mostly for qualitative purposes, then the actuator disk model may be used. The set up is easy and it is very quick to get the results. However, with 4.75% error in thrust and 19.59 % error in power estimations, the model is not suited for quantitative results. A large disadvantage with this particular actuator disk model is that it is mesh dependent.

The advanced turbine model is thought to be much better for quantitative studies. However, with 1.18 % error for thrust and 19.27 % error for power estimations for the coarse mesh simulation it is about the same as the actuator disk. Although, the wake is better represented. Further testing should be performed with finer mesh.

The flip side with the advanced model is that it is much harder to set up than the simple actuator disk model. It needs a lot of computer power, and even then it is slow. Thus, it is not suitable for a quick estimate of the influence from a wind turbine on surrounding flow.

Figure 6.7 and 6.12 shows how the actuator disk model influences the flow close to the disk heavily, but the effects dissipates fast downstream. The advanced model acts less at the turbine, but keeps the effects for longer downstream getting a more realistic wake.

It all boils down to the fact that the complex advanced turbine model has the potential to give better and more reliable results than the simple actuator disk model, but it takes time and it is computationally expensive to get them.

Figure 6.19: Actuator disk.

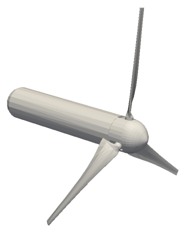


Figure 6.20: Wind turbine.

CHAPTER 7 CONCLUSION

After a case study of a moving mesh using the *mixerVesselAMI2D* tutorial to inspect how a moving mesh is made and works, a wind turbine model for 3D transient simulations has been successively made.

The actuator disk model turns out to be highly mesh dependent. The added volume forces have different values for different mesh refinement, and it is hard to create a mesh which reproduces the correct force. A mesh providing forces close to the intended forces is made by having wall functions only on the tunnel walls, and having the boundary layer around the hub fully resolved for the k-ω SST turbulence model. This mesh is named M3.

Both the velocity field and turbulent kinetic energy field in the wake is estimated quite well for mesh M3, where the k-ω SST turbulence model performs better than the k-ϵ turbulence model – as expected and predicted by literature study.

The estimated power output from the actuator disk model is calculated using weighted average of the velocity field just after the disk. The results are in correspondence to many other similar studies; the thrust is slightly underestimated with an error of -4.75 %, and the power is overestimated with an error of 19.59 % for best mesh and turbulence model. This might be due to the fact that the induction factor is estimated from a velocity field which is induced by a mesh dependent volume force which acts over a volume and not a area such as the theory describes.

A case study of a moving mesh using the *mixingVesselAMI2D* tutorial gives valuable information of how an Arbitrary Mesh Interface (AMI) works and how to set up a correct case. For good information transferal it is important with good weights between the two sides of the AMI. Tests are performed to see whether the AMI influences the calculations in an undesired way. The conclusion is that it works creating a moving mesh using AMI.

An advanced turbine model is created to make 3D transient simulations of the flow. The starting point is the *Propeller* tutorial in OpenFOAM, where changes are made to correspond with the turbine used in Blind Test 4 (Bartl and Sœtran, 2017). One of the hardest parts proved to be the AMI weights and making the interface slide instead of deforming the mesh. It also becomes clear how tricky it can be to define *snappyHexMeshDict* for the meshing correctly. The fact that it is automatic is very handy, but all the different inputs makes it somewhat difficult to keep track of.

A mesh study was performed on a 2D slice of the turbine blade airfoil used in the advanced model, to investigate the effects of the mesh refinement on the aerodynamic forces. The results yields a very dense mesh which demands the computational power of a high performance computer and a lot of time. Due to limited time a coarser mesh is used to test that the advanced model works as intended, and to establish some trends.

The results from the simulations are promising. However, the wake mesh is much coarser than desired and the mesh around the turbine should ideally be a little finer. The advanced model predicts the velocity field in the wake well, and better than the actuator disk. The turbulent kinetic energy field in the wake is not predicted as well with low peaks. The thrust coefficient is estimated to an error of 1.18 % and the power coefficient has an error of 19.27% for the coarse mesh. With some further work the model may be used to test the performance of different turbine designs in the design process.

The final conclusion is that the complex advanced turbine model works, and it has the potential to give good results for the turbine performance as well as wake prediction. However, it takes time and it is computationally expensive, especially when the fine mesh is to be used.

7.1 Further work

The advanced turbine model is now up and running, but there are still improvements to be done and extensions to be explored. One such extension of the model may be testing of different turbine designs by simply changing the CAD models. The advanced turbine model has not yet been tested for different turbine designs, but in theory it should not be a problem. This is a very attractive possible feature for the model, and thus it would be interesting to look into. However, there are improvements to be done to the existing model as well. Some of them are mentioned below.

Implementing turbine tower

One such improvement would be to include the turbine tower. This can be done by simply adding the tower to the CAD turbine model and insert it back into *snappyHexMeshDict*. Hopefully, this would help induce some of the asymmetric effects in the wake of the turbine that is missing at the moment. Since the tower would be sticking out of the AMI region there would likely be a few more cells with low weights for the AMI information transfer, but the model should handle it when the `lowWeightCorrection` option is still being used. The tower should also be added to the actuator disk model.

Longer simulation time and finer mesh

The fields for the advanced model are likely to not be fully developed for the simulated flow after 0.8 seconds, and thus the simulation time should be extended. It would also be interesting to use the intended fine mesh instead of the relatively coarse mesh used in this report. The *snappyHexMeshDict* for the desired fine mesh is added to Appendix C. The results are already promising, but they can be improved.

Easier post processing

The post processing for the advanced turbine model has been time consuming with manually extracting data from each wake surface separately due to *singleGraph* not being compatible with the AMI case. This should be looked into, because it may save a lot of time for the future user. Also, a workaround for getting the Reynolds stresses is missing.

Turbulence model

The next step for the turbulence model is to get rid of the Boussinesq approximation (2.6) used for the two equation turbulence models. Instead, one could try to use the Reynolds Stress Models (RSM) where the Reynolds stress tensor is directly computed. Using transport equations for the Reynolds stresses themselves yields 6 partial differential equations, and then one more equation for the dissipation, leaving 7 equations in total (Larsson, 2006a).

The next level after this will be to try to perform a Large Eddy Simulation (LES). It does not use the Reynolds Average Navier-Stokes (RANS) approach, but computes the large eddies directly and models only the small scale eddies. LES is more accurate than RANS because most of the turbulent kinetic energy is contained in the large eddies and is responsible for most of the momentum transfer, and these large eddies are captured in detail with LES. (Zhiyin, 2015).

BIBLIOGRAPHY

Airfoiltools, Nov. 2017. Nrel's s826 airfoil (s826-nr).
URL: `http://airfoiltools.com/polar/details?polar=xf-s826-nr-100000`.

Amer, R., Dobrev, I., Massouh, F., Jun. 2014. Determination of wind turbine far wake using actuator disk. In: Valencia Global 2014, Escuela Técnica Superior de Ingeniería del Diseño. pp. 254–260.

Bartl, J., 05 2017. Personal correspondance.

Bartl, J., Sœtran, L., 2017. Blind test comparison of the perfomance and wake flow between two in-line wind turbines exposed to different turbulent inflow conditions. Wind Energy Science 2 (1), 55–76.

Bontempo, R., Manna, M., 2017. Actuator disc methods for open propellers: assessments of numerical methods. Engineering Applications of Computational Fluid Mechanics 11 (1), 42–53.

Brennen, C., 2016. Internet book on fluid dynamics. URL: `http://brennen.caltech.edu/fluidbook/basicfluiddynamics/turbulence/lawofthewall.pdf`.

Burton, T., Sharpe, D., Jenkins, N., Bossanyi, E., 2008. Wind Energy Handbook. John Wiley and Sons.

Colley, E., 2012. Analysis of flow around a ship propeller using openfoam. Master's thesis, Curtin University, Perth, Australia.

Conway, J. T., 1995. Analytical solutions for the actuator disk with variable radual distribution of load. Journal of Fluid Mechanics 297, 327–355.

d'Emil, B., Jacobsen, M., Jensen, M. S., Krohn, S., Petersen, K. C., Sandstrm, K., Jul 2003. Size of wind turbines.
URL: `http://xn--drmstrre-64ad.dk/wp-content/wind/miller/windpower%20web/en/tour/wtrb/size.htm`.

Eriksen, P. E., 2016. Rotor wake turbulence - an experimental study of a wind turbine wake. Ph.D. thesis, The Norwegian University of Science and Technology.

Fcollonv, 2012. mergeorsplitbaffles.
URL: `http://openfoamwiki.net/index.php/MergeOrSplitBaffles`.

FINE/Marine, 2009. Note on the body forces propeller implementation. URL: `http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2010/Actuator_Disk.pdf`.

Goldstein, S., Apr. 1929. On the vortex theory of screw propellers. Proceedings of the Royal Society of London Series A 123, 440–465.

Gong, Y., Tanner, F., 2009. Comparison of rans and les models in the laminar limit flow over a backward-facing step using openfoam. Master's thesis, Michigan Technological University, Hougton.

Greenshields, C., 2013. Openfoam 2.2.0: snappyhexmesh feature snapping. URL: `http://openfoam.org/release/2-2-0/snappyhexmesh-features-layers-baffles/`.

Greenshields, C., 2014. Openfoam 2.3.0: Arbitrary mesh interface. URL: `http://openfoam.org/release/2-3-0/non-conforming_ami/`.

Greenshields, C. J., 2016. Openfoam user guide.

Gschaider, B., 2005. Howto setting up dynamic mesh cases. URL: `http://openfoamwiki.net/index.php/HowTo_setting_up_dynamic_mesh_cases`.

Guerrero, J., 2014. A crash introduction to turbulence modelling in openfoam. URL: `http://www.arek.pajak.info.pl/wp-content/uploads/2015/03/15turbulenceOF.pdf`, lecture Notes, University of Genoa.

Hansen, M., Sørensen, J., Voutsinas, S., Sørensen, N., Madsen, H., 2006. State of the art in wind turbine aerodynamics and aeroelasticity. Progress in Aerospace Sciences 42 (4), 285 – 330.

Henningson, D. S., Berggren, M., 2005. Fluid dynamics: Theory and computation. URL: `https://www.mech.kth.se/~henning/CFD/CFD_main.pdf`, lecture notes, Royal Institute of Technology in Stockholm.

Hoem, M. E., Jun. 2017. Implentation and testing of an actuator disk in openfoam. Project work, The Norwegian University of Science and Technology.

Holzmann, T., 12 2016. Mathematics, Numerics, Derivations and OpenFOAM, 4th Edition. Holzmann CFD.

Järpner, C., 2011. Projection of a mesh on a .stl surface. Master's thesis, Chalmers University of Technology.

Jasak, H., 01 2009. Dynamic mesh handling in openfoam. AIAA 341 (47).

Kalvig, S., Manger, E., Hjertager, B., 2014. Comparing different cfd wind turbine modelling approaches with wind tunnel measurements. Journal of Physics: Conference Series 555 (1).

Lam, W., Hamill, G. A., chen Song, Y., Robinson, D., Raghunathan, S., 2011. Experimental investigation of the decay from a ship's propeller. China Ocean Engineering 25 (2), 265–284.

Larsson, J., May 2006a. Reynolds stress model (rsm). URL: `https://www.cfd-online.com/Wiki/Reynolds_stress_model_(RSM)`.

Larsson, J., 2006b. Turbulence free-stream boundary conditions. URL: `https://www.cfd-online.com/Wiki/Turbulence_free-stream_boundary_conditions`.

Lloyd, C., 2017. Sampling for graphs in parallel for ami case. URL: `https://www.cfd-online.com/Forums/openfoam-bugs/193544-sampling-graphs-parallel-ami-case.html`.

Madsen, H. A., Larsen, T. J., Paulsen, U. S., Vita, L., 2013. Implementation of the Actuator Cylinder Flow Model in the HAW2 code for Aeroelastic Simulations on Vertical Axis Wind Turbines. AIAA, Ch. AIAA 2013-0913, p. 12.

Menter, F., Esch, T., 2001. Elements of industrial heat transfer predictions. In: 16th Brazilian Congress of Mechanical Engineering. pp. 117–127.

Menter, F., Kuntz, M., Langtry, R., 01 2003. Ten years of industrial experience with the sst turbulence model.

Mikkelsen, R. F., Sørensen, J. N., 2004. Actuator disc methods applied to wind turbines. Ph.D. thesis, Technical University of Denmark.

Müller, B., 2013. Chapter xi: Fvm for steady state diffusion problems, lecture notes in TEP4165, The Norwegian University of Science and Technology.

Nieto, F., Hargreaves, D., Owen, J., Hernández, S., 2015. On the applicability of 2d urans and sst $k-\omega$ turbulence model to the fluid structure interaction of rectangular cylinders. Engineering applications of computational fluid machanichs 9 (1), 153–173.

Nodeland, A. M., 2013. Wake modelling using an actuator disk model in openfoam. Master's thesis, The Norwegian University of Science and Technology.

Nozaki, F., 2014. Dynamic mesh in openfoam. URL: `https://www.slideshare.net/fumiyanozaki96/openfoam`.

Passalacqua, A., 2014. Openfoam guide/the simple algorithm in openfoam. URL: `https://openfoamwiki.net/index.php/OpenFOAM_guide/The_SIMPLE_algorithm_in_OpenFOAM`.

Russo, F., Basse, N. T., 2016. Scaling of turbulence intensity for low-speed flow in smooth pipes. Flow Measurement and Instrumentation 52, 101 – 114.

Sánchez-Caja, A., Pylkännen, J., Jun. 2007. Prediction of effective wake at model and full scale using a rans code with an actuator disk model. 2nd International Conference on Maritime Research and Transportation.

simScale Documentation, 2017. Rotating zones - SimScale Documentation.
URL `http://www.simscale.com/docs/content/simulation/model/advancedConcepts/rotatingZones/rotatingZones.html`

Sœtran, L., 2017. The closed return wind tunnel, personal correspondance.

Sœtran, L., Bartl, J., 03 2015. Invitation to the 2015 ”blind test 4” workshop, personal correspondance.

Spalart, P., Allmaras, S., 1994. A one-equation turbulence model for aerodynamic flows. La Recherche Aerospatiale 1, 5–21.

Stergiannis, N., Lacor, C., Beeck, J. V., Donnelly, R., 2016. Cfd moddeling approaches against single wind turbine wake measurements using rans. Journal of Physics: Conference Series 753.

Stergiannis, N., van Beeck, J., Runacres, M. C., 2017. Full hawt rotor cfd simulations using different rans turbulence models compared with actuator disk and experimental measurements. Wind Energy Science Discussions, 1–20.

Svenning, E., Oct 2010. Implementation of an actuator disk in openfoam.
URL: `http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2010/erikSvenning/erikSvenningReport.pdf`.

Tande, J. J., 2011. Cfd study of a 10 mw offshore horizontal axis wind turbine blade. Master’s thesis, The Norwegian University of Science and Technology.

Tossas, L. M., Leonardi, S., 2012. Wind turbine modeling for computational fluid dynamics. National Renewable Engery Laboratory.

van der Auweraert, J., 2015. Modelling of wind turbine wake with a sliding mesh. Master’s thesis, Delft University of Technology.

Versteeg, H., Malalasekera, W., 2007. An Introduction to Computational Fluid Dynamics: The Finite Volume Method. Longman Scientific and Technical.

WWEA, Oct 2016. Wwea half-year report: Worldwind wind capacity reached 456 gw.
URL: `http://www.wwindea.org/download/market_reports/Half-year_Report_WWEA_2016.pdf/`.

Yang, Z., Shih, T. H., 07 1993. New time scale based k-epsilon model for near-wall turbulence. AIAA Journal 31 (7), 1191–1198.

Zhiyin, Y., 2015. Large-eddy simulation: Past, present and the future. Chinese Journal of Aeronautics 28 (1), 11–24.

APPENDIX

A Small tutorial for using hpc Vilje

This small tutorial on how to use Vilje will only cover the necessary parts for making the simulations in this thesis run. There are some work-around ways described due to lack of knowledge of how to use the hpc in the best way. The run files/bash file is posted bellow with explanations such that the next user will know how to set it up. Do not use <> when running, just insert what is requested. More information on this is found on the NTNU hpc website: `www.hpc.ntnu.no/display/hpc/Vilje`.

Bash file basics

```
# !/bin/bash
# PBS -N <name of the job>
# PBS -A <account number on Vilje>
# PBS -m e                   // notification when job is ended.
# PBS -M <email@address.no>     // email for notification.
# PBS -l select=10:ncpus=32:mpiprocs=8    //requested resources.
# PBS -l walltime=20:00:00   //time limit for the job.

module load gcc/6.2.0
module load mpt/2.14         //versions of programs to run
module load openfoam/5.0

# Create a unique working directory
workdir=/work/$USER/$PBS_JOBID
mkdir -p $workdir

# Copy input files and move to the working directory
cp -r $PBS_O_WORKDIR $workdir
cd $workdir/<case name>

Insert OpenFOAM operations here!

# Create a unique directory in the directory where the job
# was submitted and copy the result files to that directory.
mkdir -p $PBS_O_WORKDIR/$PBS_JOBID
cp -r constant system postProcessing [0-9]* $PBS_O_WORKDIR/$PBS_JOBID
```

OpenFOAM operations for meshing

```
> blockMesh
> surfaceFeatureExtract
> decomposePar
> mpiexec_mpt snappyHexMesh -overwrite -parallel
> mpiexec_mpt mergeOrSplitBaffles -split -overwrite -parallel
> reconstructParMesh -constant
> renumberMesh -overwrite
> createPatch -overwrite
> rm -rf 0
> cp -rf 0.orig 0
> checkMesh
```

OpenFOAM operations for solving

Use the mesh case folders */system, /constant* and */0* only, and create a new bash file.

```
> decomposePar
> mpiexec_mpt pimpleDyMFoam -parallel
> reconstructPar
```

Useful commands for using Vilje

Copy case folder from local computer to Vilje (execute from local computer):
```
    > rsync -zav .  username@vilje.hpc.ntnu.no:./CaseFolder
```
Log onto Vilje:
```
    > ssh -l username vilje.hpc.ntnu.no -X
```
Go into case folder on Vilje:
```
    > cd ~/CaseFolder
```
Submit the job bash file to the que at Vilje:
```
    > qsub job.sh
```
To view the que statistics:
```
    > qstat -u username
```
To delete job from que or kill the job:
```
    > qdel jobID
```
To get to the work folder where the simulation results are located:
```
    > cd /work/username
```
To open graphical folder structure:
```
    > nautilus .
```
To open and edit file:
```
    > gnome-open file_name
```
To copy result folder back to local computer:
```
    > rsync -zavr "folder" localUserName@ipadress:~/targetLocation/
```
To get the ipadress of the local computer (executed on local computer):
```
    > hostname -I
```

B snappyHexMeshDict Actuator Disk model

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  4.1                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

castellatedMesh true;
snap            true;
addLayers       true;

geometry
{
    ActuatorDiskHub.stl
    {
        type    triSurfaceMesh;
        name    ActuatorDiskHub;
    }

    refinementBox       // for the wake, a little shorter than the tunnel
    {
        type searchableBox;
        min (-0.3 -0.6 -0.6);
        max (5 0.6 0.6);
    }
};

castellatedMeshControls
{
    maxLocalCells 200000;
    maxGlobalCells 3000000;
    minRefinementCells 0;
    maxLoadUnbalance 0.10;
    nCellsBetweenLevels 1;

    features
    (
        {
            file        "ActuatorDiskHub.eMesh";
            level       3;
        }
    );

    refinementSurfaces
    {
        ActuatorDiskHub
        {
            level       (3 3);
        }
    }

    resolveFeatureAngle 30;

    refinementRegions
    {
        refinementBox
        {
            mode inside;
            levels ((1E15 2));
        }
    }

    locationInMesh (-1.4 0 0);
    allowFreeStandingZoneFaces false;
}

snapControls
{
    nSmoothPatch 3;
    tolerance 4.0;
    nSolveIter 50;
    nRelaxIter 5;
}

addLayersControls
{
    relativeSizes true;

    layers
    {
        "(ActuatorDiskHub).*"
        {
            nSurfaceLayers 1;
        }
    }

    expansionRatio 1.0;
    finalLayerThickness 0.3;
    minThickness 0.25;
    nGrow 0;
    featureAngle 30;
    nRelaxIter 5;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;
    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 0.3;
    minMedianAxisAngle 90;
    nBufferCellsNoExtrude 0;
    nLayerIter 50;
    nRelaxedIter 20;
}

meshQualityControls
{
    maxNonOrtho 65;
    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
    minVol 1e-13;
    minTetQuality 1e-30;
    minArea -1;
    minTwist 0.05;
    minDeterminant 0.001;
    minFaceWeight 0.05;
    minVolRatio 0.01;
    minTriangleTwist -1;
    nSmoothScale 4;
    errorReduction 0.75;

    relaxed
    {
        maxNonOrtho 75;
    }
}

debug 0;    // 0: only the last mesh, 1: all the meshes for debug
mergeTolerance 1e-6;
// ************************************************************************* //
```

C snappyHexMeshDict Advanced Turbine model

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  5                                     |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

castellatedMesh true;
snap            true;
addLayers       false;

geometry
{
    refinementCylinder
    {
        type searchableCylinder;
        point1 (-0.5 0 0);
        point2 (9 0 0);
        radius 0.7;
    }
    refinementBox
    {
        type searchableBox;
        min (-1.738 -0.777 -1.300);
        max (9.312 0.933 1.300);
    }
    T1.stlb
    {
        type        triSurfaceMesh;
        name        T1;
    }
    AMI.stl
    {
        type        triSurfaceMesh;
        name        innerCylinder;
    }
};

castellatedMeshControls
{
    maxLocalCells 200000;
    maxGlobalCells 5000000;
    minRefinementCells 0;
    maxLoadUnbalance 0.10;
    nCellsBetweenLevels 1;

    features
    (
        {
            file        "AMI.eMesh";
            level       0;
        }
        {
            file        "T1.eMesh";
            level       2;
        }
    );

    refinementSurfaces
    {
        innerCylinder
        {
            level       (0 0);
            faceType    baffle;
            cellZone    innerCylinder;
            faceZone    innerCylinder;
            cellZoneInside  inside;
        }
        T1
        {
            level       (3 7);
        }
    }

    resolveFeatureAngle 30;

    refinementRegions
    {
        refinementCylinder
        {
            mode        inside;
            levels      ((1E15 5));
        }
        refinementBox
        {
           mode         outside;
           levels       ((0.5 1));
        }
        innerCylinder
        {
            mode        inside;
            levels      ((1E15 0));
        }
    }

    locationInMesh (-1.4 0 0);
    allowFreeStandingZoneFaces false;
}

snapControls
{
    nSmoothPatch 3;
    tolerance 2.0;
    nSolveIter 200;
    nRelaxIter 10;
        nFeatureSnapIter 10;
        implicitFeatureSnap false;
        explicitFeatureSnap true;
        multiRegionFeatureSnap false;
}

addLayersControls
{
    relativeSizes false;

    layers
    {
        "(T1)"
        {
            nSurfaceLayers 1;
        }
    }

    expansionRatio 1.0;
    finalLayerThickness 0.3;
    minThickness 0.1;
    nGrow 0;
    featureAngle 30;
    nRelaxIter 3;
    nSmoothSurfaceNormals 1;
    nSmoothNormals 3;
    nSmoothThickness 10;
    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 0.3;
    minMedianAxisAngle 90;
    nBufferCellsNoExtrude 0;
    nLayerIter 50;
    nRelaxedIter 20;
}

meshQualityControls
{
    maxNonOrtho 75;
    maxBoundarySkewness 20;
    maxInternalSkewness 4;
    maxConcave 80;
    minVol 1e-13;
    minTetQuality 1e-15;
    minArea -1; //1e-13;
    minTwist 0.01;
    minDeterminant 0.001;
    minFaceWeight 0.05;
    minVolRatio 0.01;
    minTriangleTwist -1;
    nSmoothScale 4;
    errorReduction 0.75;

    relaxed
    {
        maxNonOrtho 75;
    }
}

mergeTolerance 1E-6;

// ************************************************************************* //
```