Computational Fluid Dynamics Analysis of two Savonius-type Ocean Current Turbines with Augmentation Techniques

A two-dimensional, computational fluid dynamics approach investigating two Savonius-type blade profiles, with- and without a deflector and a barrier.

TOBIAS HAUKELI SKRETTING

SUPERVISOR
Joao Leal

University of Agder, 2022
Faculty of Engineering and Science
Department of Engineering and Sciences

Mandatory Declaration

1.	I/We hereby declare that my/our report is my/our own work and that I/We have not used any other sources or have received any other help than mentioned in the report.	[✓ YES]
2.	**I/we further declare that this report:** • has not been used for another exam at another department/university/university college in Norway or abroad; • does not refer to the work of others without it being stated; • does not refer to own previous work without it being stated; • have all the references given in the literature list; • is not a copy, duplicate or copy of another's work or manuscript.	[✓ YES]
3.	I/we am/are aware that violation of the above is regarded as cheating and may result in cancellation of exams and exclusion from universities and colleges in Norway, see Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§ 31.	[✓ YES]
4.	I/we am/are aware that all submitted reports may be checked for plagiarism.	[✓ YES]
5.	I/we am/are aware that the University of Agder will deal with all cases where there is suspicion of cheating according to the university's guidelines for dealing with cases of cheating.	[✓ YES]
6.	I/we have incorporated the rules and guidelines in the use of sources and references on the library's web pages.	[✓ YES]

Publishing Agreement

Authorization for electronic publishing of the report.	
I hereby give the University of Agder a free right to make the task available for electronic publishing:	[✓ YES]
Is the report confidential?	[✓ NO]
Is the task except for public disclosure?	[✓ NO]

Acknowledgements

This master's thesis built upon the knowledge I gained through my research project [1] in the previous year, which acted as a warm up for the current thesis. Prior to my research project I had only taken a course in fluid dynamics, with little to no experience within computational fluid dynamics (CFD). Most of what I have learned for CFD is self taught, however, I gained a lot of knowledge talking with my supervisor, Joao Leal, and also from Moisés Brito, who introduced me to OpenFOAM, and Ghali Raja Yakoub.
During my research project, I struggled to make the simulations run properly in OpenFOAM for almost half a year, until I finally managed to crack the code, thinking I would be ready for my thesis. This was not the case. Once again, countless of hours went into troubleshooting, despite following the methodology that had previously worked. Eventually I figured out the problem. Having fought through all the hardships and setbacks during this study, I believe I can now say, confidently, that I am able to perform turbine CFD simulations. Nevertheless, there is still much to learn, and I am looking forward to expand my knowledge in the field.

Abstract

Ocean current turbines are one of many environmentally friendly, prospective energy sources out there, however, it is still at an embryonic stage in development. This thesis aims to build on the existing knowledge in the field, by investigating the design of two Savonius type ocean current turbines, both with- and without surrounding structures that augment their performances. The two profiles evaluated were the semi circular- and the elliptic blade profile. For this purpose, the computational fluid dynamics software, OpenFOAM, was utilised, with the geometry and mesh created in Solidworks and Gmsh, respectively. The Reynolds Average Navier-Stokes equations were used, employing the $k - \omega$ SST turbulence closure model, together with the PIMPLE pressure-velocity coupling algorithm. Wall functions were implemented to estimate the flow parameters in the wall boundaries, using an average y^+ value of 300. However, results show that this approach provided inaccurate results, most likely due to poor estimations of flow separation. Augmentations increased the power coefficient of the semi circular turbine by 50.78%, from 0.258 to 0.389, whereas the elliptic profile saw a 79.71% increase in power coefficient, from 0.276 to 0.496. Future work regarding this thesis should look at further enhancement of the elliptic profile, optimizing the augmentation around it. Moreover, a finer grid should be implemented.

Contents

List of Figures

List of Tables

Abbreviations and Nomenclature

Abbreviations

AMI	Arbitrary Mesh Interface
AR	Aspect Ratio
CFD	Computational Fluid Dynamics
DNS	Direct Numerical Simulation
GAMG	Geometric-Algebraic Multi-Grid
LES	Large Eddy Simulation
OpenFOAM	Open Source Field Operation and Manipulation
OR	Overlap Ratio
PIMPLE	Not an abbreviation, but a combination of PISO and PIMPLE
PISO	Pressure-Implisit with Splitting of Operator
RANS	Reynolds Average Navier-Stokes
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations Extrapolation
SIMPLE	Semi-Implicit Method for Pressure-Linked Equations
SIMPLEC	Semi-Implicit Method for Pressure-Linked Equations Consistent
SIMPLER	Semi-Implicit Method for Pressure-Linked Equations Revised
SST	Shear Stress Transport
TSR	Tip Speed Ratio

Greek

α	Turbulence closure coefficient blended between α_1 and α_2	[-]
α_1	Turbulence closure coefficient	5/9 [−]
α_2	Turbulence closure coefficient	0.44 [−]
α_r	Linear relaxation coefficient	[−]

β	Turbulence closure coefficient blended between β_1 and β_2	[-]
β^*	Turbulence closure coefficient	0.09 [-]
β_1	Turbulence closure coefficient	3/40 [-]
β_2	Turbulence closure coefficient	0.0828 [-]
δ	Boundary layer thickness	[m]
κ	von Kármán constant	[0.41 –]
μ	Dynamic viscosity	[$\mathrm{kg\,m^{-1}\,s^{-1}}$]
μ_t	Turbulent, dynamic eddy viscosity	[$\mathrm{kg\,m^{-1}\,s^{-1}}$]
ν	Kinematic viscosity	[$\mathrm{m^2\,s^{-1}}$]
ω	Specific turbulence dissipation rate	[$\mathrm{s^{-1}}$]
Ω_R	Angular velocity of the rotor in RPM	[RPM]
ω_r	Angular velocity of the rotor	[$\mathrm{s^{-1}}$]
ω_{BC}	Specific turbulence dissipation rate inlet and outlet boundary condition	[$\mathrm{s^{-1}}$]
ω_{log}	Specific turbulence dissipation rate in the inertial region for wall boundaries	[$\mathrm{s^{-1}}$]
ω_{vis}	Specific turbulence dissipation rate in the viscous region for wall boundaries	[$\mathrm{s^{-1}}$]
ω_{WBC}	Specific turbulence dissipation rate wall boundary condition	[$\mathrm{s^{-1}}$]
ρ	Density	[$\mathrm{kg\,m^{-3}}$]
σ_k	Turbulence closure coefficient blended between σ_{k1} and σ_{k2}	[-]
$\sigma_{\omega 1}$	Turbulence closure coefficient	0.5 [-]
$\sigma_{\omega 2}$	Turbulence closure coefficient	0.856 [-]
σ_ω	Turbulence closure coefficient blended between $\sigma_{\omega 1}$ and $\sigma_{\omega 2}$	[-]
σ_{k1}	Turbulence closure coefficient	0.85 [-]
σ_{k2}	Turbulence closure coefficient	1 [-]
$\sigma_{k\varepsilon}$	Turbulence closure coefficient	1.00 [-]
τ_w	Wall shear stress	[$\mathrm{kg\,m^{-1}\,s^{-2}}$]
ε	Turbulent dissipation rate	[$\mathrm{m^2\,s^{-3}}$]
ζ	Model coefficient of small value to prevent floating point exception	[–]

Roman

A	Swept area of the rotor	[m^2]
a_1	Turbulence closure coefficient	0.31 [−]
B_k	Model coefficient	[8.366 −]
B_r	Rotational friction coefficient	[kg m^2 s^{-1}]
C	Constant for estimating eddy length scale	[-]
C_1	Turbulence closure coefficient	[-]
C_2	Turbulence closure coefficient	[-]
C_p	Power coefficient of a turbine	[-]
C_{eps2}	Model coefficient	[1.9 −]
$C_{f,x}$	Schlichting and Gersten's local skin-factor correlation	[-]
C_{kBC}	Model coefficient	[−0.416 −]
$CD_{k\omega}$	Cross-diffusion term	[kg m^{-3} s^{-2}]
e	Overlap length	[m]
F_1	Blending function, auxiliary turbulence relation	[-]
F_2	Blending function, auxiliary turbulence relation	[-]
G	Body accelerations acting on the fluid	[meter/s^2]
g	Gravitational acceleration	9.81 [m/s^2]
k	Turbulent kinetic energy	[m^2 s^{-2}]
k_{BC}	Turbulent kinetic energy inlet and outlet boundary condition	[m^2 s^{-2}]
k_{log}	Turbulent kinetic energy in the inertial region for wall boundaries	[m^2 s^{-2}]
$k_{vis/log}$	Turbulent kinetic energy placeholder for k_{vis} and k_{log}	[m^2 s^{-2}]
k_{vis}	Turbulent kinetic energy in the viscous region for wall boundaries	[m^2 s^{-2}]
k_{WBC}	Turbulent kinetic energy wall boundary condition	[m^2 s^{-2}]
L	Length of the boundary layer	[m]
l	Reference length scale of energy-containing eddies	[m]
L_D	Equivalent pipe diameter	[m]
L_t	Length, or height, of the turbine	[m]

P_c	The power captured by the turbine	[W]
P_D	Perimeter	[m]
P_k	Production term for turbulent kinetic energy	[$\mathrm{kg\,m^{-1}\,s^{-3}}$]
P_{KE}	Kinetic energy available in the fluid flow	[W]
R	Radius of the turbine	[$\mathrm{m^2}$]
Re	Reynolds number	[-]
S	Invariant measure of the strain rate	[$\mathrm{s^{-1}}$]
T_r	Torque from shaft and generator	[$\mathrm{kg\,m^2\,s^{-2}}$]
T_t	Turbine torque	[$\mathrm{kg\,m^2\,s^{-2}}$]
U	Average velocity	[$\mathrm{m\,s^{-1}}$]
u	Velocity	[$\mathrm{m\,s^{-1}}$]
u^*	Friction velocity	[$\mathrm{m\,s^{-1}}$]
U^+	Dimensionless velocity	[-]
y	Normal distance to the nearest wall	[m]
y^+	Dimensionless normal distance to the nearest wall	[-]
y_c	Distance from the nearest wall to the cell centroid	[m]
$\tilde{P}_k$	Production term limiter	[$\mathrm{kg\,m^{-1}\,s^{-3}}$]
$\vec{U}$	Average velocity vector with x, y, z components (U, V, W)	[$\mathrm{m\,s^{-1}}$]
$\vec{u}$	Velocity vector with x, y, z components (u, v, w)	[$\mathrm{m\,s^{-1}}$]

Chapter 1

Introduction

During the last decade, extensive research has targeted a variety of renewable energy sources to prevent further degradation of the climate, while fossil fuels are gradually being phased out. One of the branches of research focus on capturing energy from tides and ocean currents, which could provide stable, clean energy, especially for remote areas close to the ocean that are currently dependant on fossil fuels. While tidal- and ocean current turbines bear heavy resemblance to wind turbines, underwater conditions are more demanding, thus this technology is still in its embryonic stages. When only considering Canada, China, and Norway, an estimated 9078 MW of tidal turbines can be installed, accounting for grid connectivity, sustainability, shipping routes, and more [2]. Learning to harness this energy could prove to be a valuable asset to the energy mix. Doing this as efficiently as possible requires a well designed turbine.

1.1 Objective

To aid propel the progression of drag based ocean current turbines, this thesis will utilise computational fluid dynamics (CFD) to evaluate their design. The initial idea was spurred by OceanEnvi, who look to harvest the energy of the ocean. For the simulations, OpenFOAM version 9 [3] will be used, along with SolidWorks 2021-2022 and Gmsh v4.8.4 [4] for geometry design- and meshing. This thesis aims at investigating blade profiles and surrounding structures that augment the performance of the turbine.

1.2 Report Structure

Following is a comprehensive description of the relevant computational fluid dynamics theory in Chapter 2. This includes the Navier-Stokes equations, turbulence modeling, different types of meshes and indicators of what constitutes a good mesh, and finally the pressure-velocity coupling algorithms, where large parts of text is taken from the previous work made by the author in [1]. Some alterations and improvements were made where fit, however, for the most part it is the same. When the foundation of CFD has been laid out, a literature review of existing ocean current turbines and CFD simulations of turbines will be presented in Chapter 3, which also includes parts from the previous research project [1]. Subsequently, the research questions of the current thesis will be stated in Chapter 4, discussing how this thesis builds on the previous literature. After the research questions have been defined, the methodology used to answer them is presented in Chapter 5, with the results and discussion thereof given in Chapter 6. Finally, the work is concluded in Chapter 7, ending with recommendations for future work in Chapter 8.

Chapter 2

Computational Fluid Dynamics and Modeling Theory

In the design process of a system containing fluid flow, the use of computational fluid dynamics (CFD) has proven very beneficial. Prior to its formation, around the 1960s [5], engineers had to perform experiments and calculations by hand, if they wanted to study the behaviour of fluids. Now, CFD can be used to analyse, test, and optimise designs on a computer, reducing the time and financial assets required. In this chapter, the most important governing equations used in CFD programs will be presented. Additionally, the meshed cells where these equations are applied, and the solvers and turbulence models used to calculate them, are also briefly introduced.

2.1 Navier-Stokes Equations

Fluid flow is governed by the laws of physics. The equations used to describe its behaviour are called the Navier-Stokes equations, shown in Eqs. 2.1 and 2.2. They were initially derived by Claude Louis Marie Henri Navier, a French engineer, in the 19th century, and then later on George Stokes, a British mathematical physicist, arrived at the same derivation independently [5], [6]. For incompressible fluids, as is the case of water in our application, the first equation is the continuity equation,

$$\nabla \cdot \vec{u} = 0, \tag{2.1}$$

which is when the divergence of the velocity vector, $\vec{u}$ [m s^{-1}], is zero. That is, when the sum of the rate of change in the x, y, and z direction is zero. The momentum equation, Eq. 2.2, is in essence Newton's second law of motion, $\vec{F} = m\vec{a}$,

$$\rho \left[\frac{\partial \vec{u}}{\partial t} + \left(\vec{u} \cdot \nabla \right) \vec{u} \right] = -\nabla p + \mu \nabla^2 \vec{u} + \rho \vec{G}, \tag{2.2}$$

where the left hand side describes mass times acceleration per volume of fluid, and the right hand side encapsulates all the forces acting on this fluid volume. Here, ρ [kg m^{-3}] is the density, p [kg m^{-1} s^{-2}] is the pressure, μ [kg m^{-1} s^{-1}] is the dynamic viscosity, and $\vec{G}$ [m s^{-2}] is the body acceleration, often replaced by the gravitational acceleration $\vec{g}$, 9.81 [m s^{-2}] in the downwards direction. To build on that, on the left side, mass is described by the density, which is mass divided by volume, and the acceleration is simply the derivative of velocity. Next, the pressure forces are described by the divergence of pressure, being negative since the pressure force is acting from the surrounding fluid onto the control volume. Lastly, the viscous forces are the viscosity multiplied by the Laplacian of velocity. By inserting the full form of the velocity vector and the del (or nabla) operator ∇,

$$\vec{u} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (2.3) \qquad \nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right] \quad (2.4)$$

into Eq. 2.2, the complete expression for the Navier-Stokes momentum equation is obtained,

$$\begin{aligned} \rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}\right) &= -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) + \rho G_x \\ \rho\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z}\right) &= -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}\right) + \rho G_y \ . \\ \rho\left(\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z}\right) &= -\frac{\partial p}{\partial z} + \mu\left(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}\right) + \rho G_z \end{aligned} \quad (2.5)$$

Solving these equations, however, is no easy task as the instantaneous velocity field can vary a lot in both space and time for turbulent flows, underlined by being one of the seven Millennium Prize Problems [7]. Consequently, several models and solvers that estimate the solution accurately have been developed. An essential part of these methods is the turbulence modeling.

2.2 Turbulence Modeling

Turbulence can be described as the volatile component of motion in flows with high Reynolds numbers [6]. It is often visually characterised by eddies of different sizes, shapes, and orientations, swirling around in the fluid, such as the well known von Kármán vortex street. Modeling this swirling, turbulent behaviour is important, as some applications, such as vortex generators on an airfoil [8] or in heat exchangers [9], take advantage of it, while others want to avoid turbulence, such as minimizing stall cells in pump-turbines [10]. Several methods modelling turbulence exist, all with their advantages and disadvantages, characterised by the Reynolds Average Navier-Stokes (RANS) equations and partly by the Large Eddy Simulation (LES). Another option is to use Direct Numerical Simulation (DNS), although this calculates all the flow properties numerically, thus does not model the turbulence. A comparison between the three is shown in Figure 2.2.1, which can help indicate the difference in accuracy [11]. For RANS, the main focus is directed towards the mean flow, and how the turbulence affects its properties [12], where the entirety of the Navier-Stokes equations are modeled [5]. Compared to LES and DNS, the computational resources necessary are small, making it the dominant method for complex engineering purposes [12]. Variations of RANS, such as unsteady RANS, are also common, and research has been made using machine learning techniques in combination with RANS [13]–[15]. Additionally hybrid RANS-LES models are often used [5], [16], [17]. In LES, one ideally wants to resolve about 90% of Navier-Stokes equations numerically, while the remaining 10% are modelled. This will allow the large eddies to be resolved, while the small eddies are modelled, providing a better picture of the true behaviour of the flow. Direct numerical simulation takes this up a notch, resolving 100% of the Navier-Stokes equations numerically, which is mostly used for specific, low Reynolds research and experiments [18], [19], or to aid the validation of lower cost models [20]. [1], [5]

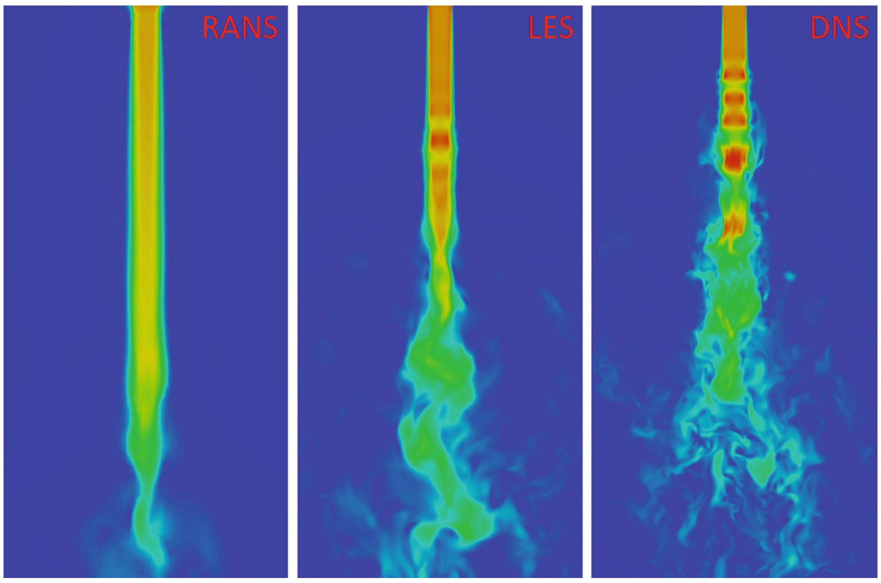


Figure 2.2.1: A comparison of RANS, LES, and DNS [11].

Since both LES and DNS are computationally heavy, RANS will be the main focus of this paper. The RANS equations resemble the original Navier-Stokes equations in Eq. 2.1 and 2.2, although the velocity vector is now the time-averaged velocity $\vec{U}$, and a turbulent eddy viscosity μ_t [$\mathrm{kg\,m^{-1}\,s^{-1}}$] appears due to this time-averaging procedure [5], [6],

$$\nabla \cdot \vec{U} = 0, \tag{2.6}$$

$$\rho \left[\frac{\partial \vec{U}}{\partial t} + \left(\vec{U} \cdot \nabla \right) \vec{U} \right] = -\nabla P + (\mu + \mu_t) \nabla^2 \vec{U} + \rho \vec{G}. \tag{2.7}$$

To solve the RANS equations, several turbulence closure models can be utilised, such as $k-\varepsilon$ [21], $k-\omega$ [22], and $k-\omega$ SST (shear stress transport) [23]. From literature the latter method is the one that performs the best, [24]–[29], although the realizable $k-\varepsilon$ (one of the many variants of $k-\varepsilon$) model has also proven to perform well [30]–[32]. In the OpenFOAM software, the Menter *2003* version of $k-\omega$ SST, [23], [33] is used. Following is a description of this model, with its equations, based on Menter *2003* [23] and the OpenFOAM documentation [33]. Some adjustments to the notation used in the equations were made to stay consistent with the notation used in this report.

The $k-\omega$ SST model is a combination of the $k-\varepsilon$ and $k-\omega$ model, made to mitigate the shortcomings of the two models working individually [23], [34]. Due to a lack of sensitivity to adverse pressure-gradients, the $k-\varepsilon$ model predicts notably higher shear-stress levels than what would actually occur, resulting in a delay in, or even a prevention of, separation [35]. Adverse pressure gradients often arise in curved geometries, such as for an airfoil, making the $k-\varepsilon$ unfavourable for simulations of such geometries. Additionally, it implements highly nonlinear damping functions in the sublayer, which could impede the convergence properties of the scheme. Conversely, the

$k-\omega$ model does not require damping functions in the sublayer, increasing the numerical stability, moreover, it also has a superior performance with adverse pressure-gradients in the boundary layer, i.e. near walls. Despite these advantages, $k-\omega$ is heavily dependant on the freestream values of the specific turbulence dissipation rate, ω [s^{-1}], such that even a minor difference in this value could drastically affect the eddy-viscosity, μ_t [kg m^{-1} s^{-1}], and skin friction coefficient, $C_{f,x}$ [−] [23], [36], [37]. In this regard, $k-\varepsilon$ is superior. Following is a description of the $k-\omega$ SST model, where first an explanation of what k, ε, and ω are will be presented, followed by their respective transport equations. Then, the relationship between the two models, $k-\varepsilon$ and $k-\omega$, including the blending between them in the $k-\omega$ SST model, will be explained.
The turbulent kinetic energy, k [m^2 s^{-2}], describes the kinetic energy of the eddies in turbulent flow [12]. This energy is proportional to the size of the eddies, such that large eddies contain more energy than small eddies. Figure 2.2.2 represents this, where the spectral energy (kinetic energy per unit mass and per unit wavenumber) is plotted as a function of the inverse of eddy size (wavenumber). Larger eddies are formed by the shear forces of the mean flow, and in turn their energy generate smaller eddies [5]. As the energy progressively propagates into lesser and lesser eddies (larger wavenumbers), there comes a point where the viscous forces and the shear forces are equal. At this point the eddies dissipate into heat, which occurs at a rate ε [m^2 s^{-3}], known as the turbulent dissipation rate. To obtain ω [s^{-1}], the following relation can be used [1]

$$\omega = \frac{\varepsilon}{\beta^* k} \iff \varepsilon = \omega\beta^* k, \tag{2.8}$$

where ω represents the specific turbulence dissipation rate, with $\beta^* = 0.09$ being an empirical constant.
As k, ε, and ω have been defined, their transport equations for the turbulence models will be presented. Each transport equation follows the same pattern: rate of change of the variable + transport by convection = transport by diffusion + rate of production - rate of destruction [12]. The transport equations for $k-\varepsilon$ are as follows

$$\rho\left[\frac{\partial k}{\partial t} + \nabla\left(\vec{U}\cdot k\right)\right] = \nabla\left[\left(\mu + \frac{\mu_t}{\sigma_{k\varepsilon}}\right)\nabla k\right] + P_k - \rho\varepsilon \tag{2.9}$$

and

$$\rho\left[\frac{\partial \varepsilon}{\partial t} + \nabla\left(\vec{U}\cdot \varepsilon\right)\right] = \nabla\left[\left(\mu + \frac{\mu_t}{\sigma_{\varepsilon}}\right)\nabla \varepsilon\right] + C_1\frac{\varepsilon}{k}P_k - C_2\rho\frac{\varepsilon^2}{k}, \tag{2.10}$$

where P_k [kg m^{-1} s^{-3}] is a production term representing the conversion of the mean kinetic energy from the flow to the turbulent kinetic energy of the eddies [12]. $\sigma_{k\varepsilon}$, σ_ε, C_1, and C_2 are empirical constants with values 1.44, 1.92, 1.00, and 1.30, respectively.
The transport equations used in the $k-\omega$ model are [23]

$$\rho\left[\frac{\partial k}{\partial t} + \nabla\left(\vec{U}\cdot k\right)\right] = \nabla\left[(\mu + \sigma_{k\omega}\mu_t)\nabla k\right] + P_k - \rho\beta^* k\omega \tag{2.11}$$

and

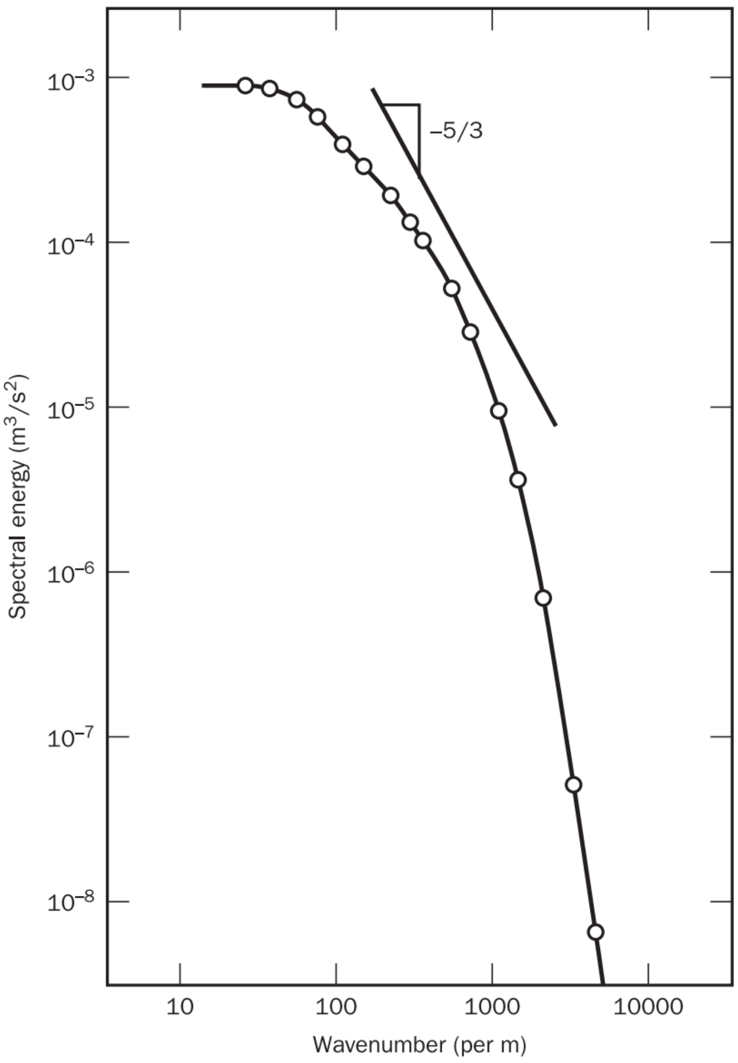


Figure 2.2.2: Typical turbulence energy spectrum as a function of the wave number [12].

$$\rho\left[\frac{\partial\omega}{\partial t}+\nabla\left(\vec{U}\cdot\omega\right)\right]=\nabla\left[\left(\mu+\sigma_{\omega'}\mu_t\right)\nabla\omega\right]+\frac{\rho\alpha_1}{\nu_t}P_k-\rho\beta_1\omega^2, \tag{2.12}$$

where $\sigma_{k\omega}$, $\sigma_{\omega'}$, α_1, and β_1 are empirical constants with values 0.5, 0.5, $\frac{5}{9}$, and $\frac{3}{40}$, respectively, and ν_t [m^2 s^{-1}] is the turbulent kinematic viscosity.
To construct the transport equations used in the $k-\omega$ SST model, it is helpful to look at the relationship between the $k-\varepsilon$ and $k-\omega$ transport equations by inserting Eq. 2.8 into Eqs. 2.9 and 2.10, resulting in

$$\rho\left[\frac{\partial k}{\partial t}+\nabla\left(\vec{U}\cdot k\right)\right]=\nabla\left[\left(\mu+\sigma_{k\omega}\mu_t\right)\nabla k\right]+P_k-\rho\beta^* k\omega, \tag{2.13}$$

and

$$\rho\left[\frac{\partial\omega}{\partial t}+\nabla\left(\vec{U}\cdot\omega\right)\right]=\nabla\left[\left(\mu+\sigma_{\omega'}\mu_t\right)\nabla\omega\right]+\frac{\rho\alpha_2}{\nu_t}P_k-\rho\beta_2\omega^2+2\rho\sigma_{\omega 2}\frac{1}{\omega}\nabla k : \nabla\omega. \tag{2.14}$$

Comparing these transport equations for $k-\varepsilon$, written in terms of ω, with the transport equations for $k-\omega$, one can see that for k they are both equal, whereas for ω the only difference is the term $2\rho\sigma_{\omega 2}\frac{1}{\omega}\nabla k : \nabla\omega$. This term is known as the cross-diffusion term. Now, by implementing a blending function, F_1, in the cross-diffusion term, such as

$$\rho\left[\frac{\partial\omega}{\partial t}+\nabla\left(\vec{U}\cdot\omega\right)\right]=\nabla\left[\left(\mu+\sigma_\omega\mu_t\right)\nabla\omega\right]+\frac{\rho\alpha}{\nu_t}P_k-\rho\beta\omega^2+2\left(1-F_1\right)\rho\sigma_{\omega 2}\frac{1}{\omega}\nabla k : \nabla\omega, \tag{2.15}$$

where F_1 varies from 0 to 1 depending on the distance to the nearest wall, y [m],

$$F_1 = tanh\left\{\left\{min\left[max\left(\frac{\sqrt{k}}{\beta^*\omega y},\frac{500\nu}{y^2\omega}\right),\frac{4\rho\sigma_{\omega 2}k}{CD_{k\omega}y^2}\right]\right\}^4\right\}, \tag{2.16}$$

the $k-\omega$ SST will blend smoothly between its two turbulence closure models. Here, $CD_{k\omega}$, [kg m^{-3} s^{-2}] is simply a limitation of the cross-diffusion term, such that it never reaches zero to ensure Eq. 2.16 is not divided by zero, resulting in a floating point exception, defined as

$$CD_{k\omega} = max\left(2\rho\sigma_{\omega 2}\frac{1}{\omega}\nabla k : \nabla\omega, 10^{-10}\right). \tag{2.17}$$

For cells close to a wall, F_1 is 1 and the cross-diffusion term disappears from Eq. 2.15, resulting in the transport equation for the standard $k-\omega$, Eq. 2.12. Conversely, for cells far away from a wall, F_1 is 0, such that the transport equation corresponds to the one found in $k-\varepsilon$, Eq. 2.14. Since F_1 varies between 0 and 1, as seen in Appendix B, the $k-\omega$ SST model can blend smoothly between ε and ω. This now reflects the advantages of the two turbulence models, where $k-\omega$ is used close to the walls, where adverse pressure gradients are likely to occur, whereas $k-\varepsilon$ is used in the freestream, since it is not overly sensitive in this region.
Furthermore, with $k-\omega$ SST, the turbulent eddy viscosity, μ_t, is calculated using Eq. 2.18,

$$\mu_t = \frac{\rho a_1 k}{max\left(a_1\omega, SF_2\right)}, \tag{2.18}$$

where $S=\sqrt{2S_{ij}S_{ij}}$ [s^{-1}] is the invariant measure of the strain rate, where S_{ij} is the mean stress tensor [s^{-1}]) and F_2 [-] is another blending function,

$$F_2 = tanh\left[\left[max\left(\frac{2\sqrt{k}}{\beta^*\omega y},\frac{500\nu}{y^2\omega}\right)\right]^2\right]. \tag{2.19}$$

For CFD codes, the production term, P_k, is replaced by a production limiter, $\tilde{P}_k$ [kg m^{-1} s^{-3}], that prevents accumulation of turbulence in stagnation regions, defined as

$$P_k = \mu_t\frac{\partial U_i}{\partial x_j}\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right) \longrightarrow \tilde{P}_k = min\left(P_k, 10\cdot\beta^*\rho k\omega\right). \tag{2.20}$$

Turbulence closure coefficients, such as α, β, σ_k, and σ_ω, are also subject to the blending between $k-\varepsilon$ and $k-\omega$, using Eq. 2.21. Here, they replace the place holder variable ϕ. Additionally, the place holders ϕ_ω and ϕ_ε are replaced by the constants corresponding to the variable that is being calculated, specified in Table 2.2.1. This linear interpolation method is done to improve the transition between the two turbulence models.

$$\phi = \phi_\omega F_1 + \phi_\varepsilon (1 - F_1) \tag{2.21}$$

Table 2.2.1: Empirical, dimensionless turbulence closure coefficients for Menter *2003* $k - \omega$ SST [23], [33].

			ϕ_ω				ϕ_ε			
Constant	a_1	β^*	α_1	β_1	σ_{k1}	$\sigma_{\omega 1}$	α_2	β_2	σ_{k2}	$\sigma_{\omega 2}$
Value	0.31	0.09	5/9	3/40	0.85	0.5	0.44	0.0828	1	0.856

2.3 Mesh

When applying the Navier-Stokes equations and the turbulence models in numerical simulations, the computational domain has to be discretized into small cells, or elements [38]. These cells comprise what is known as a mesh, often called grid. Every single cell in the mesh represents a discrete space of the fluid domain, moreover, the governing equations are solved in each individual cell. In this chapter an overview of the two main types of mesh, unstructured- and structured mesh, will be presented. Additionally, what constitutes a good mesh will be discussed. Large parts of this is gathered from the previous work in [1].

2.3.1 Unstructured Mesh

An unstructured mesh can be characterized by its irregular connectivity, and is often applied in complex, irregular geometries [38], as seen in Fig. 2.3.1. This type of meshing is simpler to automate than the structured mesh, due to its ability to take on a variety of shapes, including triangular-, and polygon cells for 2D, and tetrahedral-, pyramid-, or polyhedral shaped for 3D. The process of generating this type of mesh is quite basic, where points need to be created, and the connectivity between these points must be defined. Due to its flexibility, this method is often preferred, however, its irregularities can often yield skewed elements that require more computational power in numerical simulations compared to its structured counterpart. This is especially noticeable close to boundary layers or other sensitive regions. Numerical computations are easier solved with regular, structured cells. By combining unstructured and structured grids a hybrid mesh is made [5]. [39]

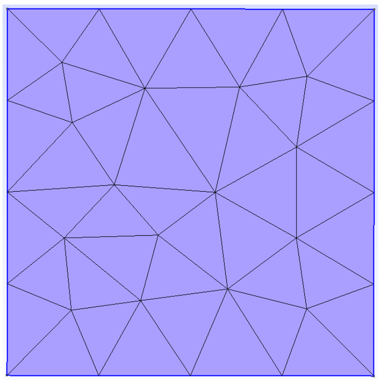


Figure 2.3.1: Unstructured grid with triangles made in Gmsh.

2.3.2 Structured Mesh

The structured mesh can broadly be divided into Cartesian and curvilinear grids [5], with examples of these being shown in Fig. 2.3.2 together with a body-fitted curvilinear grid. In curved geometries the Cartesian method would generate staircase-like steps, shown in Fig. 2.3.3a [38].

In Fig. 2.3.3b, however, the grid is first made in a curvilinear domain, and subsequently transformed into the physical domain using some transform function. In modern meshing software, these transformations occur behind the scenes, where the user usually only has to specify the resolution of the grid [5], [6], [38]. Common for the structured grids, is that they consist of quadrilateral- or hexahedral cells for 2D and 3D, respectively, with a regular connectivity [38], [39], contrary to the irregular connectivity of the unstructured grid. If the fluid domain is complex, it is possible to divide it into several areas that are meshed individually, with multiblock grids. If all these sub-domains have a structured mesh, the grid as a whole will be termed as a block structured grid [6]. When making block structured grids, it is preferable to have matching cells in the interface between the multiblocks. That is, the nodes of the cells on either side of the interface should be connected to each other, otherwise the computational time will increase [40]–[43].

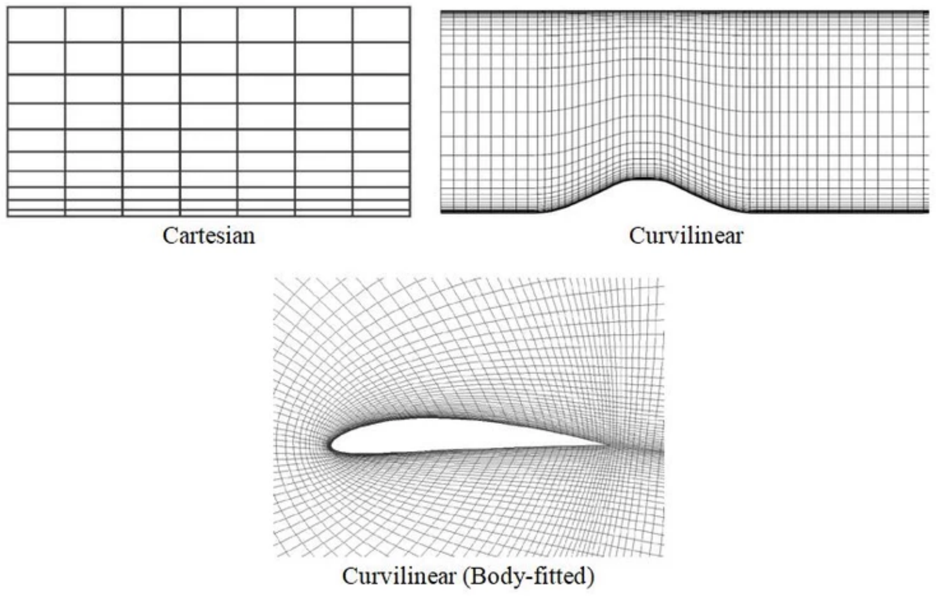
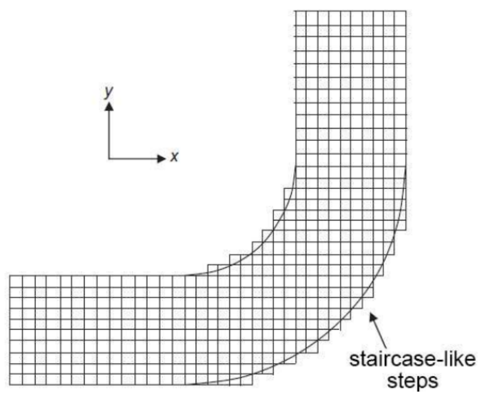
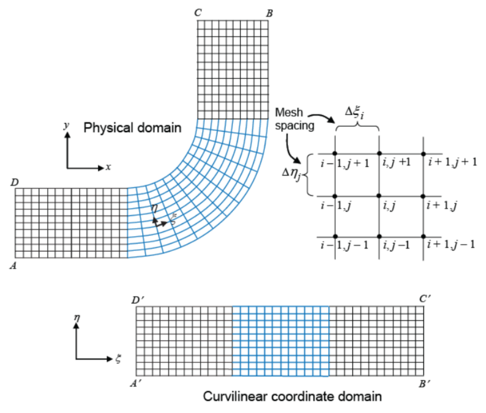


Figure 2.3.2: Example of a Cartesian, curvilinear, and body-fitted curvilinear grid [43].



(a) Cartesian grid.



(b) Body-fitted grid with both the physical- and the curvilinear domain.

Figure 2.3.3: Cartesian- and body-fitted curvilinear grid on a 90° bend [38].

2.3.3 Rotating Mesh

When implementing rotating meshes to for example marine current turbines, it is difficult to achieve matching grids. One method of generating the rotating mesh is the overset method, also known as a Chimera mesh, which is a variant of multiblock grid [5], [6]. An example of the Chimera mesh is shown in Fig. 2.3.4 [44]. Here, blocks that are individually meshed overlap with each other. The unnecessary cells will subsequently be removed in a process called hole cutting, which often results non-matching grids at the interface [6], [44]. To alleviate this issue, interpolation methods are implemented.

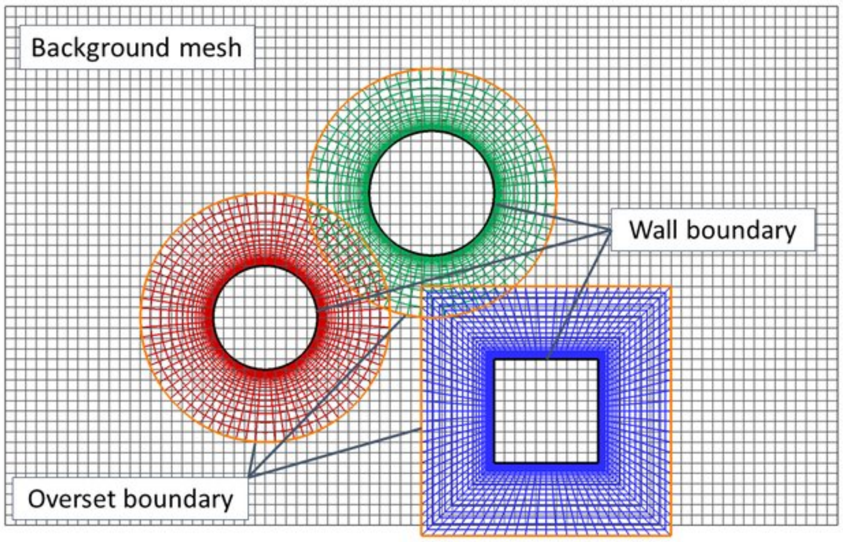


Figure 2.3.4: Example of a Chimera mesh [44].

Another method used for moving topologies is the Arbitrary Mesh Interface (AMI) [46]. Fig. 2.3.5 depicts how this could look [45], [46]. This method also requires interpolations between the cells on either side of the sliding interface to determine the flow properties (pressure, velocity, specific turbulence dissipation rate, etc.). For this procedure, a weighting system is implemented to determine the contribution of each cell fraction on the face of the neighbouring domain/patch [46], [47]. The AMI method performs well for the rotating behaviour of a turbine [46], although deviations could occur at the interface due to poor matching of cells. A good mesh should define this AMI region some distance away from sensitive regions to safeguard the accuracy of the solution.

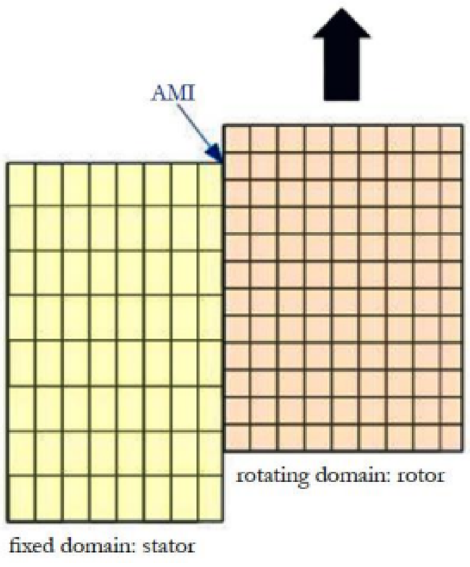
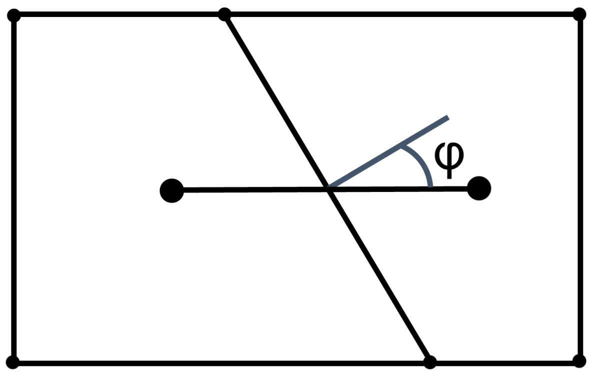


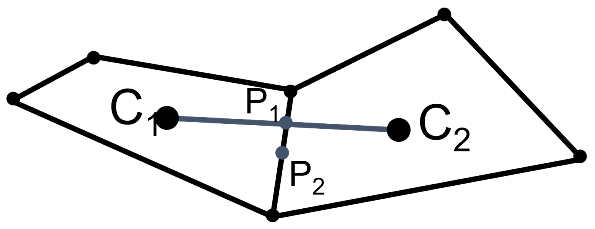
Figure 2.3.5: Example of the Arbitrary Mesh Interface (AMI) between a fixed domain and a moving domain [45], [46].

2.3.4 Mesh Quality

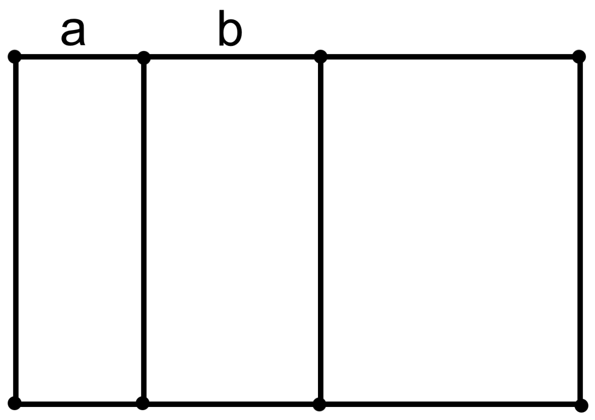
While a straightforward definition of what constitutes a good mesh can be difficult to describe, parameters such as orthogonality, skewness, smoothness, and aspect ratio can be used to give an indication of the quality. It is worth noting that OpenFOAM has a command, "checkMesh", that will evaluate these parameters [48]. The sketches in Figure 2.3.6 can be used to identify what each of the parameters represent. Orthogonality, Figure 2.3.6a, is defined as the angle, phi, between the line connecting two cell centroids and the normal of the face between them. The desired orthogonality is zero degrees, although OpenFOAM allows angles up to 65°before the "checkMesh" command issues a warning, according to the source code [49]. Furthermore, the source code defines a maximum internal skewness of 4, where skewness, Figure 2.3.6b, describes the relationship between the line C_1C_2 and P_1P_2. While OpenFOAM describes smoothness by the difference in volume between connected cells, it could be helpful to imagine the growth ratio between the length a and the length b, Figure 2.3.6c. Too large growth ratios will impede the computational speed, moreover, the difference between the largest and smallest volume should not be too big. Lastly, the aspect ratio, Figure 2.3.6d, is the ratio between the longest, c, and shortest, d, face of the cell, where the optimal value is one.



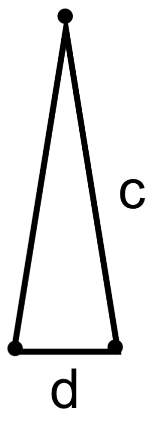
(a) Orthogonality



(b) Skewness



(c) Smoothness



(d) Aspect ratio

Figure 2.3.6: Different parameters that help describe the quality of the mesh [48], [50], [51]

Ultimately, a good mesh should be able to provide an accurate solution, representative of the true-, analytical solution, while also enabling the required computational time to be small. To safeguard this, the mesh should be refined in regions of importance and areas where large velocity gradients are likely to occur, while regions of less importance can be coarser [6]. To deduce which regions will be affected by large velocity gradients, a preliminary simulation can be made with a coarse mesh, giving a rough overview of affected areas. Additionally, logic and experience can come in handy, as regions such as the tip of a blade or at a pipe intersection are bound to have larger gradients than in the freestream. A poor mesh can increase simulation time, or in worst case induce numerical diffusions that crash the simulations entirely [52]. This becomes increasingly important close to walls, as the behaviour of fluids within the so called "boundary layer" differs from that seen in the freestream.

2.3.5 Boundary Layers

In boundary layers, care must be taken when modeling the change of velocity between the no-slip condition at the wall, where the velocity is zero, to the freestream value [53], shown in Fig. 2.3.7. The y^+ and U^+ are dimensionless values for the normal distance to the nearest wall and the velocity, respectively defined as,

$$y^+ = \frac{\rho \cdot y_c \cdot u^*}{\mu}, \tag{2.22}$$

and

$$U^+ = \frac{U}{u^*}. \tag{2.23}$$

Here, y_c [m] is the distance from the wall to the cell centroid and u^* [m s^{-1}] is the friction velocity defined as

$$u^* = \sqrt{\frac{\tau_w}{\rho}}, \tag{2.24}$$

where τ_w [kg m^{-1} s^{-2}] is the wall shear stress in Eq. 2.25.

$$\tau_w = \frac{1}{2} \cdot \rho \cdot U^2 \cdot C_{f,x}, \tag{2.25}$$

where $C_{f,x}$ is the local skin friction, which Schlichting and Gersten [54] fitted into an empirical equation from values found in their computations for Reynolds numbers below 10^9,

$$C_{f,x} = (2 \cdot log_{10}(Re) - 0.65)^{-2.3}. \tag{2.26}$$

Note that due to the uncertainty of turbulent flows, there are several possible ways to calculate the skin friction coefficient [55], many of which can be found in [56]. The Reynolds number can be calculated as,

$$Re = \frac{\rho \cdot U \cdot L}{\mu}, \tag{2.27}$$

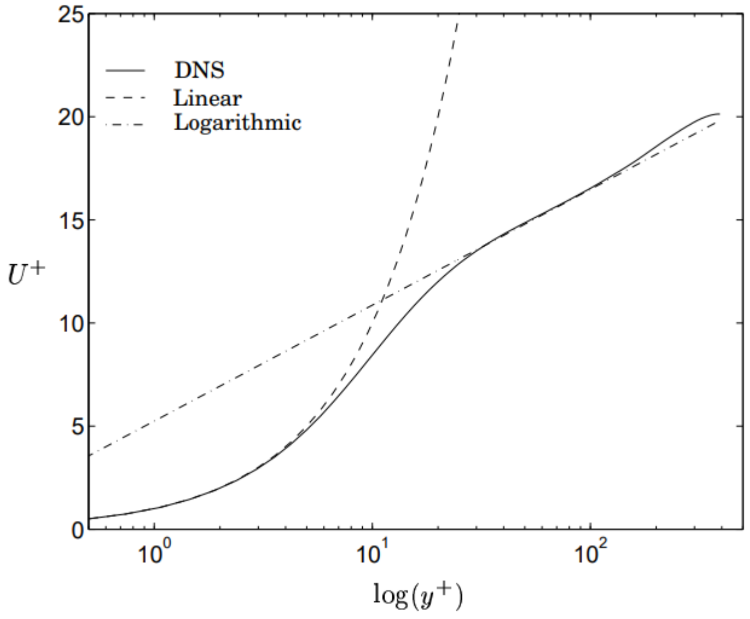


Figure 2.3.7: Dimensionless velocity profile in the boundary layer as a function of y^+ [53].

where L [m] is the length of the boundary layer.

The region with $0 < y^+ < 5$ can be termed as the viscous (or laminar) sublayer, $5 < y^+ < 30$ as the buffer (or transient) layer, and $30 < y^+ < 200$ as the inertial sublayer. While the roughness of a surface can play a part in the boundary layer, if it is smaller than the thickness of the viscous sublayer, one can assume a hydrodynamically smooth surface [54], [57], [58]. Surfaces such as glass and plastic can often be regarded as hydrodynamically smooth, therefore the effect this roughness has on the velocity profile is negligible [55]. In this boundary layer, the mesh should be very fine, with the first cell height being two times the cell centroid height in Eq. 2.22. Thereafter, a number of layers, growing in height, are added perpendicular to the wall, atop of the first cell, until the boundary layer is resolved. These layers are known as inflation layers. The growth rate depends on the amount of layers desired, as well as the boundary layer thickness. This boundary layer thickness, δ [m], can be estimated based on an equation for turbulent flow found in [55],

$$\delta = \frac{0.38 \cdot L}{(Re)^{\frac{1}{5}}}. \tag{2.28}$$

To create a good mesh, the estimated values for y and δ can be used in an iterative meshing process, where the first mesh uses these values, and is used for simulations. Then, in the post-processing, the actual values of y^+ are noted down, and a new mesh is created if the target y^+ value has not been reached. It is important to mention that several of these equations are based on experiments of a moving flat plate, thus estimations for complex geometries are less likely to be accurate.

To accurately simulate the near-wall behaviour of the fluid, y^+ should be less than 1, with the first grid points being well within the viscous sublayer. In this scenario, the solution obtained will be good, however, at the cost of increasing the computational power required, thus also the time spent simulating. Another option is to use empirical wall functions, where y^+ should be larger than 11.63, preferably between 30 and 500 [12]. This will provide similar results as with $y^+ < 1$, with less time spent. The reasoning for a preferred $y^+ \in [30,500]$ can be illustrated using Fig. 2.3.7, as this is the region where the logarithmic assumption fits the best, moreover, it is apparent that it does not describe the buffer layer sufficiently.

2.4 Boundary Conditions

These empirical wall functions are used in the wall boundaries, and must be defined together with all the other boundaries (inlet, outlet, etc.) to define the system. To achieve the greatest accuracy, measurements of the system parameters should be taken beforehand, although this is only possible if the object being simulated already exists. Often this is not the case, however, therefore estimations of the initial parameters must be made. In this chapter, a brief explanation of the wall functions and estimations is presented for velocity, turbulent kinetic energy, specific turbulence dissipation rate, pressure, and turbulent viscosity.

2.4.1 Velocity

Previously, when talking about the mesh quality in Chapter 2.3.4, the no-slip condition was briefly introduced. This is a well known concept, where viscous fluids immediately adjacent to a solid surface will stick to it, such that it does not slip [6], [55]. This happens due to the shear stress of the fluid being equal to the shear stress of the wall, resulting in a velocity of zero relative to the wall [12]. Most fluids abide by this condition, including water. Therefore, when simulating a hydrokinetic turbine, wall-surfaces can be defined as no-slip boundaries. A rotating turbine will act as a moving wall, where the boundary condition has to be defined as moving to ensure that the velocity is zero relative to the wall, and not relative to the starting coordinates. Free surfaces can be approximated as wall boundaries with slip condition, such that there are zero shear stresses [55]. Regarding the inlet boundary, the velocity can be defined as a Dirichlet condition, meaning it has a fixed value of choice, such that the turbine behaviour can be studied at whichever velocity desired. For the outlet, as long as the it is placed sufficiently far away from geometrical disturbances, such that the flow becomes fully developed, and assuming the area of the inlet and outlet are equal, the velocity can be defined with the same direction and magnitude as in the inlet. In cases where the flow does not fully develop, the outlet should allow for backward flow, although this is not recommended. Versteeg and Malalasekera [12] state that the distance between the last physical obstacle and the outlet should be ten times the height of the last obstacle to ensure a fully developed flow, with accurate results. Similarly, the cross sectional area of the domain, should be ten times larger than the area of the turbine to avoid confinement effects [59]. If the confinement is too large, the performance of the turbine will be overestimated, where both upstream- and downstream conditions can be altered, and the forces on the rotor will change.

2.4.2 Turbulent Kinetic Energy

The wall boundary condition for turbulent kinetic energy, k, can be found using a wall function. In OpenFOAM, there are two of these for k: the "kLowReWallFunction" and the "kqRWallFunction". The "kLowReWallFunction" can be used for both low- and high Reynolds numbers, switching between two equations for k depending on whether the cell is in the viscous or inertial sublayer. OpenFOAM uses

$$k_{WBC} = max\left(k_{vis/log}u^{*2}, \zeta\right), \tag{2.29}$$

for this, where ζ is a small value to prevent a floating point exception. $k_{vis/log}$ represents k_{vis} if the cell is in the viscous sublayer and k_{log} in the inertial sublayer, defined as

$$k_{vis} = \frac{2400C_{f,x}}{C_{eps2}^2} \tag{2.30}$$

$$k_{log} = \frac{ln(y^+)\,C_{kBC}}{\kappa} + B_k \tag{2.31}$$

Here, $\kappa = 0.41$ [−] is the von Kármán constant, and $B_k = 8.366$, $C_{eps2} = 1.9$, and $C_{kBC} = -0.416$ are dimensionless model coefficients [60]. This is a discontinuous form of blending between the two sublayers, however, as long as $y^+ \in [30,500]$, Eq. 2.31 should be the only one used.
The "kqRWallFunction" can be used for high Reynolds number flows, simply defining the turbulent kinetic energy at the wall as a zero-gradient [61].
As for the non-wall boundaries an estimation of k can be made as

$$k_{BC} = \frac{3}{2}(UI)^2 \tag{2.32}$$

where I [−] is the turbulence intensity. This equation is embedded in OpenFOAM's inlet condition turbulentIntensityKineticEnergyInlet. In [12], a typical turbulence intensity between 1% and 6% was noted, while in [62], which focuses on river turbines, a value of 15% was used for a turbine deployed at the bottom of a river. In [63], values between 3.2% and 24% across five different locations were noted in their literature review regarding the effect turbulence intensity has on ocean current turbines. Four of these locations were noted at 5 m above the seabed, with velocities between 1.3 and 3.5 m s^{-1}.

2.4.3 Specific Turbulence Dissipation Rate

For ω, OpenFOAM has a wall function adequately named omegaWallFunction is used. By default it uses a step wise switch between ω_{vis} and ω_{log}, similar to the one for kLowReWallFunction, switching between the following to equations

$$\omega_{vis} = \frac{6\nu}{\beta_1 y^2} \tag{2.33}$$

$$\omega_{log} = \frac{\sqrt{k}}{\sqrt[4]{\beta^*}\kappa y} \tag{2.34}$$

using ω_{vis} in the viscous sublayer and ω_{log} in the inertial sublayer. Again, as long as $y^+ \in [30,500]$, Eq.2.34 will be used. In the event the mesh resolves the viscous sublayer, and a continuous blending is desired, there are several options available for the omegaWallFunction, which can be found in the user guide for omegaWallFunction [64]. It is appreciable that both of these equations are inversely proportional to the nearest-wall-distance y, such that ω tends to infinity as $y \longrightarrow 0$, which makes sense considering energy should dissipate quickly near walls with the no-slip condition.
For the inlet and outlet boundaries, as well as in the freestream, the initial value of ω can be estimated as

$$\omega_{BC} = \frac{\sqrt{k_{BC}}}{\sqrt[4]{\beta^*} l} \qquad (2.35) \qquad\qquad l = CL_D \qquad (2.36)$$

where l is a reference length scale of the energy-containing turbulent eddies, C is a constant, and L_D is the hydraulic diameter of the inlet [12], [55]. In [12], this constant, C, has a value of 0.07, whereas in [5] it is 0.1. In OpenFOAM, the inlet can be set to turbulentMixingLengthFrequencyInlet, which incorporates Eq. 2.35 directly. The reference length scale will not be calculated automatically, thus this needs to be specified, where the hydraulic diameter is defined as

$$L_D = \frac{4A}{P_D}, \tag{2.37}$$

with A being the area of the inlet, and P_D [m] representing the inlet perimeter.

2.4.4 Pressure

When dealing with pressure in boundary conditions, it is common practice to prescribe a Dirichlet condition at either the inlet or the outlet, serving as a reference value for the rest of the domain [12], [55]. Note, that both the pressure and the velocity can not be specified simultaneously at one boundary, as they are coupled in the Navier-Stokes equations, causing the problem to be over-specified. If one of them are defined, then the other will accommodate to its value. When an inlet velocity is specified, then the pressure at the inlet can be set to the zero gradient condition. This zero gradient can also be used on walls.

2.4.5 Turbulent Eddy Viscosity

As discussed in Chapter 2.2, the turbulent eddy viscosity will be calculated using Eq. 2.18. This is not the case for walls where the boundary layer is not fully resolved, thus, once again, wall functions are required. For this, OpenFOAM has several solutions which can be found in the user guide for wall conditions in [65]. Most of these define ν_t as zero in the viscous sublayer, while having different functions for calculating it in the inertial sublayer.

2.5 The PIMPLE Pressure-Velocity Coupling Algorithm

When the mesh has been constructed and the necessary boundary conditions have been specified, the governing equations can be solved. This is done iteratively using algorithms like the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE), the Pressure-Implicit with Splitting of Operator (PISO), or the combination of the two: PIMPLE [5], [6], [12]. These algorithms will perform the pressure-velocity coupling in each cell of the mesh, following a procedure as shown in Figure 2.5.1, which illustrates a simplified flowchart of the main principles of the PIMPLE algorithm.
Together with the boundary conditions, an initial guess of the flow parameters is made, where the pressure is used to solve the momentum equations, Eq. 2.2, for the velocity vector. However, in most cases, the resulting velocity field does not satisfy the continuity equation, Eq. 2.1, at this

point, and needs to be corrected by utilising a Poisson equation for pressure. This pressure corrector is derived, in essence, by rearranging the momentum equation for velocity, then inserting it into the continuity equation. Thereafter, the resulting values for velocity and pressure need to be under-relaxed, as to ensure a more stable solution,

$$X_t = \alpha_r X_t^{'} + (1 - \alpha_r)\, X_{t-1} \tag{2.38}$$

where X_t is the resulting value to be used, $X_t^{'}$ is the calculated value to be relaxed, X_{t-1} is the value for the previous time step, and α_r is the relaxation coefficient. If $\alpha_r = 1$, there is no relaxation, and the model accepts $X_t^{'}$ as the new value. For under-relaxation, values between zero and one are chosen, with the optimal one often being found through trial and error [66]. The resulting residuals can be used as an indication of which value of α_r is most suited, employing the one giving the best convergence. Similar to pressure and velocity, the transport equations for the turbulence modeling are solved and under-relaxed as well. Subsequently, the turbulent eddy viscosity will be updated in accordance to Eq. 2.18 in Chapter 2.2. When a possible solution has been acquired, the convergence is evaluated. If the criteria are met, the simulation continues to the next time step. Conversely, if convergence has not been reached, all the parameters are updated and a new cycle begins. Up until this point, for the first time step, all three algorithms (SIMPLE, PISO, and PIMPLE) would go through the same procedure. This is not the case going forward. Since PIMPLE is a combination of SIMPLE and PISO, this is the point where a choice of whether it should proceed as the SIMPLE- or the PISO algorithm is made. The difference between SIMPLE and PISO is the amount of times they calculate the momentum equations. The SIMPLE algorithm will solve the momentum equations for every single iteration, whereas the PISO algorithm only performs this calculation once, before jumping straight into the pressure corrector, utilising the updated velocity parameter. For SIMPLE, the iterative loop is often called the outer loop. Similarly, the PISO loop can be termed as the inner loop. When using the PIMPLE algorithm, the amount of times it runs in either of the loops can be specified in the software. The SIMPLE algorithm has the advantage of being very stable, and is often used for steady-state calculations. Variations of SIMPLE, such as SIMPLE Consistent (SIMPLEC), SIMPLE Revised (SIMPLER) and SIMPLE Extrapolation (SIMPLEX) have been shown to be more robust than their unmodified version, with SIMPLEC also proving to be the optimal choice [67]. They are all similar to SIMPLE, albeit with minor modifications. PISO is efficient, but it can suffer from instabilities, and is more adept in transient simulations. Moreover, it requires a low Courant number to provide stable results, resulting in low time steps. [5], [6], [12], [68]–[74]
The Courant number, Co [−], is defined as the ratio between the time step Δt and the time a fluid particle spends to convect through a mesh cell Δn while moving at a velocity U [75],

$$Co = U\frac{\Delta t}{\Delta n}. \tag{2.39}$$

Being a combination of SIMPLE and PISO, another key aspect of the PIMPLE algorithm is its ability to use Courant numbers higher than 1, while still providing accurate results [76]. For viscous turbomachinery flow, this Courant number has been found to give adequate results up to a value of 10 when employing an implicit scheme [77].

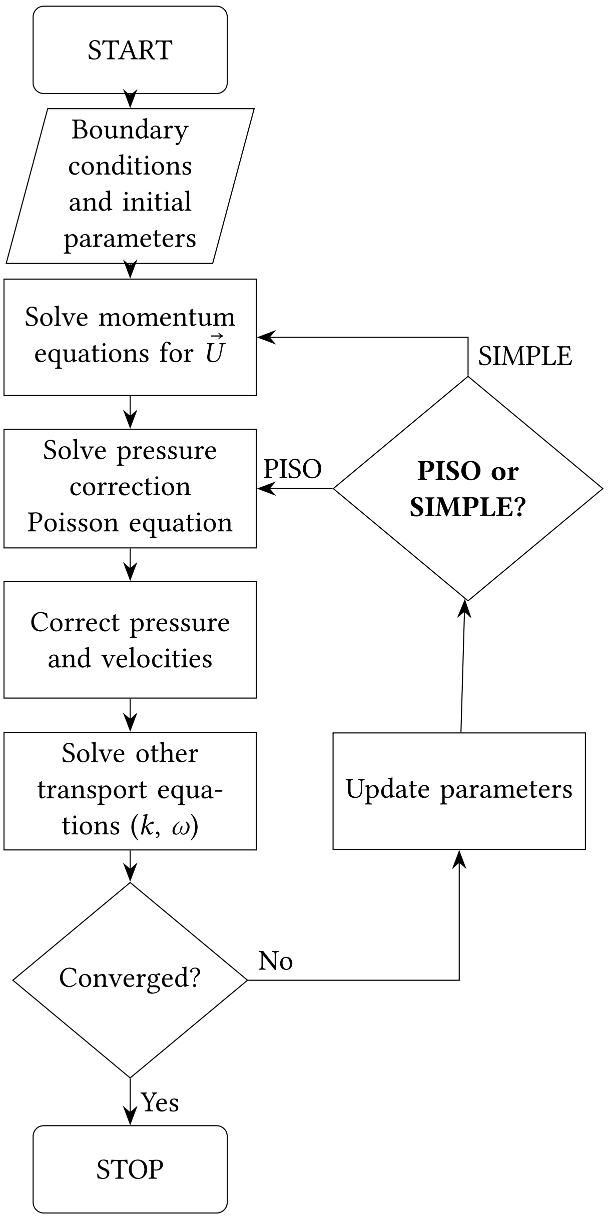

START
Boundary conditions and initial parameters
Solve momentum equations for $\vec{U}$
SIMPLE
Solve pressure correction Poisson equation
PISO
PISO or SIMPLE?
Correct pressure and velocities
Solve other transport equations (k, ω)
Update parameters
Converged?
No
Yes
STOP

Figure 2.5.1: Flowchart of the PIMPLE Algorithm. Based on [5], [6], [12] and the OpenFOAM guides and source codes for SIMPLE [68]–[70], PISO [71], [72], and PIMPLE [73], [74].

Chapter 3

Literature Review

While the previous chapters aimed at providing a brief overview of the different aspects and equations of CFD, this chapter dives further into different types of turbines in literature, as well as how authors have conducted CFD simulations of them. The aim of this chapter is therefore to provide a better foundation for the choices made in the current thesis, including which geometry will be studied and the setup of the simulation.

3.1 Rudimentary Concepts

Compared to wind, the average velocity found in ocean currents is quite small. Nevertheless, due to the density of water being more than 800 times that of air, an ample amount of power is still available. Equation 3.1 indicates the potential kinetic power, P_{KE} [W], available in fluid flow, with a plot comparing the power density for hydrokinetic- and wind turbines in Appendix A,

$$P_{KE} = \frac{1}{2} \cdot \rho \cdot A \cdot U^3, \tag{3.1}$$

where ρ [kg m^{-3}] is the density, A [m^2] is the swept area of the turbine, and U [m s^{-1}] is the velocity of the flow. It is important to note that the marine currents seldom surpass velocities of 4 m s^{-1}, which is not the case for wind. While this equation describes how much power is available in the flow, it does not actually describe how much power is captured. For hydrokinetic turbines, the Betz limit is known as the theoretical maximum amount of power that can be extracted from the flow, with a value of approximately 59.3%. Most turbines, however, are not close to this limit. Fig. 3.1.1 illustrates how the power coefficient can vary for typical wind turbines at different tip speed ratios, TSR [-], which can give an idea of how the corresponding hydrokinetic turbines would behave [78]. The power coefficient [-] is defined as

$$C_p = \frac{P_c}{P_{KE}} = \frac{P_c}{\frac{1}{2} \cdot \rho \cdot A \cdot U^3}, \tag{3.2}$$

where

$$P_c = T_t \cdot \omega_r, \tag{3.3}$$

is the captured power (power take-off) [W], where T_t [kg m^2 s^{-2}] is the turbine torque, and ω_r

[s^{-1}] is the angular velocity. For angular velocity in RPM, Ω_r will be used as notation, where the conversion $\Omega_r = \frac{\omega_r \cdot 60}{2\pi}$ is used. The TSR describes the relationship between the angular velocity of the turbine, multiplied by its radius, R [m], and the incoming flow velocity, defined as

$$TSR = \frac{\omega_r \cdot R}{U}. \tag{3.4}$$

The power coefficient is one of the key parameters that are used to describe turbines, together with the torque coefficient,

$$C_T = \frac{T_t}{T_a} = \frac{T_t}{\frac{1}{2} \cdot \rho \cdot A \cdot U^2 \cdot R} \xrightarrow{Eqs.\ 3.3\ and\ 3.4} \frac{C_p}{TSR}, \tag{3.5}$$

where T_a is the available torque.
When conducting CFD analysis of a rotating turbine there are two ways to study its behaviour. The first method uses a predetermined angular velocity, forcing the rotation of the turbine to be constant. As the torque reaches quasi-stable conditions, the simulation is finished. This method enables a straightforward way to study the turbine at specific TSR, as long as the radius and incoming velocity are known. Conversely, the second method uses a flow-induced motion, where instead of changing the angular velocity directly, a damping coefficient, B_r [kg m s^{-1}], representing the friction of the shaft and generator, is adjusted. The induced torque from this friction can be calculated as

$$T_r = B_r \cdot \omega_r, \tag{3.6}$$

which will act in the opposite direction of the rotation, and can be used to calculate the power take-off of the turbine, such that $T_t = T_r$.

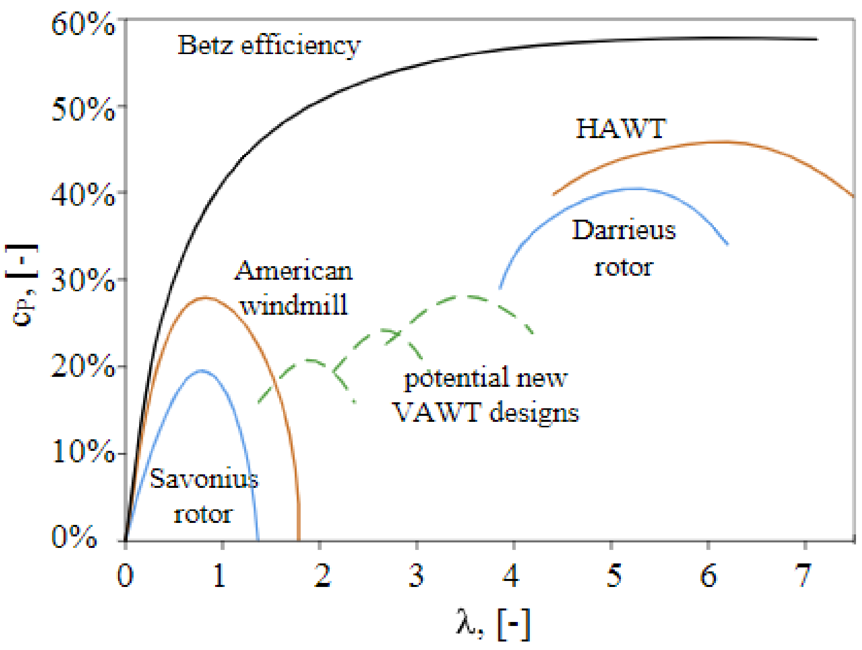
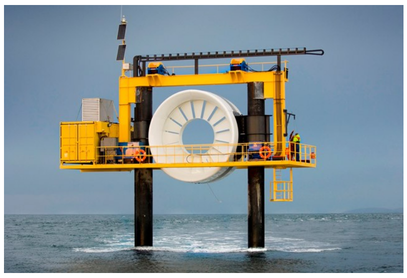


Figure 3.1.1: The power coefficient (C_p) of conventional wind turbines based on the TSR (in this article, λ was used for TSR) [78].

(a) Alstom Oceade [81]


(b) OpenHydro [82]

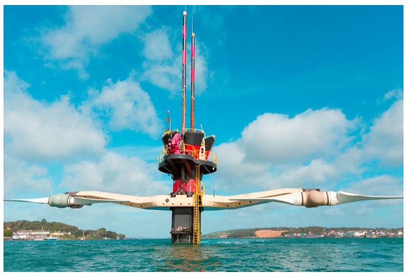

(c) SeaGen [83]

Figure 3.2.1: Three different tidal current turbines.

3.2 Overview of Turbines

Zhou et al. [79] review marine tidal current turbines rated above 500 kW that were up-to-date in 2014, with an update available in [80] from 2017. Most of these bore heavy resemblance to horizontal axis wind turbines, several of which had megawatt levels of production. Figure 3.2.1 illustrates three of these turbines, namely the Alstom Oceade, OpenHydro, and the SeaGen.
The European Marine Energy Center Ltd. created a list of all the tidal current turbine concepts known to them, with Table 3.2.1 showing the distribution of these turbines by category [84]. Their list was last updated on the 19. February 2020 when the author last visited their page. Based on this, it becomes even more apparent the horizontal axis turbines are the most popular.

Table 3.2.1: Categorisation of different tidal current turbines [84].

Total	**97**
Horizontal Axis	43
Vertical Axis	16
Tidal Kite	2
Other	12
Oscillating Hydrofoil	4
Enclosed Tips (Venturi)	5
Archimedes Screw	2
Unclassified	13

In [85], a review of cross-flow hydrokinetic turbines was made, investigating its technology, configurations, and performance. The two main types of configurations for cross-flow turbines are lift-based and drag-based turbines, with a hybrid of the two also being used. Examples of lift-based turbines are the Darrieus and Gorlov Helical Darrieus turbines. Common for these is the use of airfoils to generate the lift force, while minimizing the drag force. Drag-based configurations, as the name implies, rely on the drag forces to drive the turbine, with a popular example being the Savonius rotor shown in Fig. 3.2.2. Compared to lift-based configurations they are typically 20% less efficient [86], however, they have better self-starting abilities [87] and have a noticeably lower TSR. A lower TSR makes it less likely to kill marine animals swimming nearby. Additionally, the risk of cavitation can be decreased, which is a more frequent issue for large TSR [88]–[90]. The cavitation phenomena occurs when the pressure decreases below the vapour pressure, causing bubbles to appear which can damage the structure [91]. By employing different augmentation techniques, the efficiency and self-starting capabilities of the drag-based turbines can be improved. Deflector blades can be used to deflect the flow away from the returning blade, decreasing the negative drag forces [92]. Moreover, curtaining can be applied to direct the water into the leading blade [93]. Similarly, the Savonius can be ducted, which led to a theoretical power coefficient of 53% at a TSR of 3.5, and a theoretical cut-in velocity of 0.5 m s^{-1} in [94]. This is a quite significant improvement compared to the general power coefficient of about 20% at TSR below 1, recalling Figure 3.1.1. The ducted turbine requires a large construction to achieve this, however. Designs with hinged blades that close to reduce the negative drag forces were also reviewed, with the Hunter turbine [95] and the CU turbine [96] being examples of this.

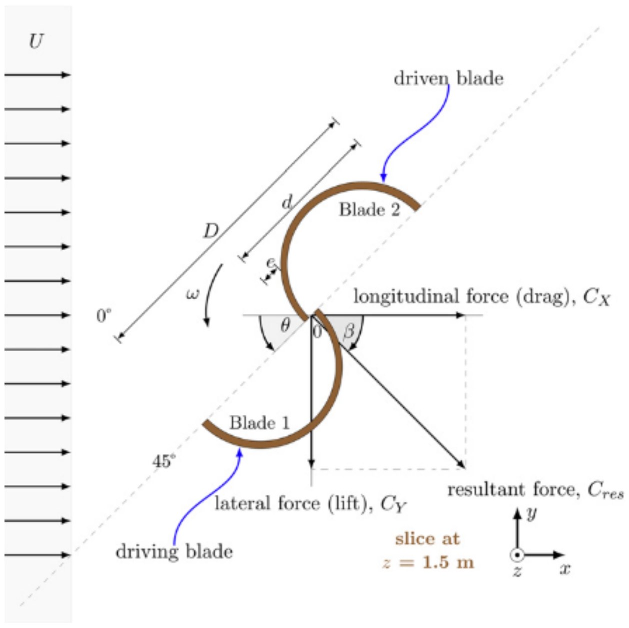


Figure 3.2.2: The geometric parameters of a typical Savonius rotor [97]. Drag-based configuration.

In [98] a number of small scale hydrokinetic turbines, below 5 kW, were investigated. Some of the turbine manufacturers claimed to be able to produce power from current velocities as low as 0.5 m s^{-1}, however, several of these designs seem to be obsolete, indicating that this is not economically feasible. Both axis flow propeller- and cross-flow designs were investigated, with- and without ducts. The ducted designs were able to achieve better performances, which aids reducing the cost of energy. Although the increase in material usage for these designs may counter-balance this cost reduction. Due in part to their small size, most of the turbines were easily installed, using either pontoon boats, floating buoys with anchors, mooring it to the ground, or a weighted base with cables to ensure they stayed put.

Bachant and Wosnik [99] performed experiments on two helical cross-flow turbines — a Gorolov Helical Turbine and a Lucid Spherical Turbine — using a towing tank with velocities between 0.5 m s^{-1} and 1.5 m s^{-1}. Results showed that the Helical turbine had superior performance.

Mohammadi et al. [100] optimized a hydrokinetic turbine for low-speed flow using particle swarm optimisation and Xfoil for velocities between 0.5 m s^{-1} and 3 m s^{-1}. Their turbine was a 3-bladed horizontal axis turbine, achieving a power coefficient of approximately 0.45 between 0.5 m s^{-1} and 2.5 m s^{-1}, before it decreased to 0.28 at 3 m s^{-1}. They found that thicker hydrofoils are more impervious to cavitation. The probability of cavitation was low even at the blade tip.

A horizontal axis cross-flow turbine, designed for 100 W at a velocity of 1.2 m s^{-1}, was constructed

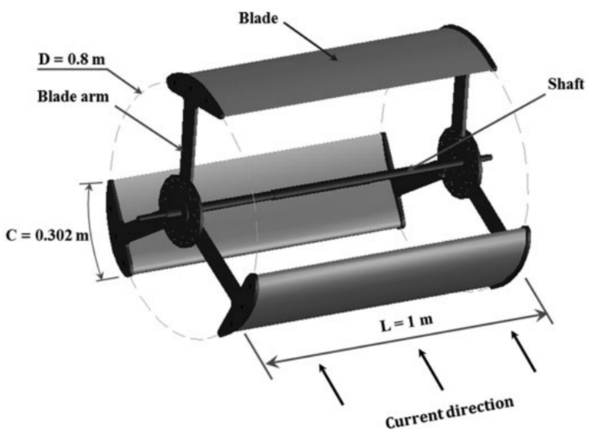


Figure 3.2.3: Straight bladed cross-flow hydrokinetic turbine designed for 100 W at 1.2 $\mathrm{m\,s^{-1}}$ [101]. Lift-based configuration.

and tested in both a saltwater channel and the sea in [101]. When creating this design, they made a number of assumptions, including among other things a transmission efficiency of 0.7, mechanical efficiency of 0.7, three blades, a tip speed ratio at peak power of 1.5, and a length of 1 m. Moreover, they used previous literature to create the basis of calculation for their turbine. The resulting design is showed in Figure 3.2.3. During their experiments, they exceeded the design power, reaching a maximum output of 304 W at 1.8 $\mathrm{m\,s^{-1}}$. Experiments showed a cut-in velocity of 0.8 $\mathrm{m\,s^{-1}}$.

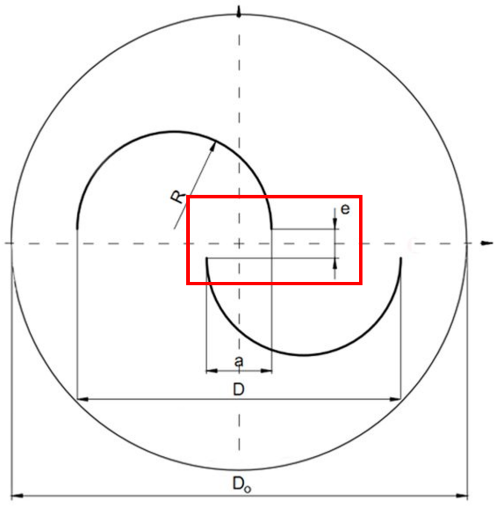
Alom and Saha [102] reviewed the progress and evolution of the Savonius rotor blade profile and shape, comparing several profiles. It is important to note that this study focused only on wind turbines. Nevertheless, there are similarities. In [103], the authors compared the performance of a Savonius hydrokinetic turbine with an identical Savonius wind turbine using experiments and CFD. They used a simple, three bladed, semi-circular rotor, and found that the power coefficient of the hydrokinetic turbine exceeded that of the wind turbine by 61.32%, from 0.24 to 0.39. The lower performance of the wind turbine was caused in part by flow circulation at the blade tip, as well as separation in the concave side of returning blades. This would impose a force acting against the motion of the turbine. Among the blade profiles evaluated by Alom and Saha, the Roy profile [104], [105] and the new elliptic profile [106] were concluded as the most promising ones, having C_p values of 0.3 and 0.33, respectively, with Banerjee et al. [107] achieving a C_p of 0.31 for a similar elliptical profile. Furthermore, they underlined the importance and impact of aspect ratio (AR)

$$AR = \frac{L_t}{D_t}, \tag{3.7}$$

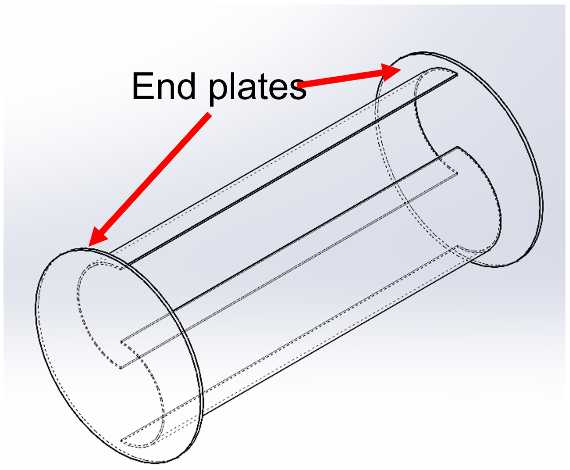
overlap ratio (OR)

$$OR = \frac{e}{D_t}, \tag{3.8}$$

number of blades, and end plates. Figure 3.2.4 illustrates what is meant by overlap and end plates for the Savonius turbine. In Eqs. 3.7 and 3.8, L_t [m] is the length, or height, of the turbine, D_t [m] is the turbine diameter, and e [m] is the overlap distance. While rotor moment and inertia decreases with an increase in AR, angular acceleration increases. They determined that for higher incoming velocities, a higher AR is desired. For the OR, values between 0.15 and 0.20 seem to give the highest power coefficients for single-stage rotors. An overlap between the blades will allow the fluid to flow through the gap, such that it can act on the concave side of the returning blade, aiding the rotation of the rotor. Interestingly, for helical blades they found that an OR of zero was favourable. Regarding the number of blades, most designs use two to four, however C_p values have proven to be higher when using only two blades. While end plates increase the inertia of the turbine, they also prevent fluid from leaking from the concave side of the rotor. This will ensure a more uniform pressure difference along the blades. Regarding the end plates, usually the diameter should be 1.1 times the diameter of the turbine. Too large end plates could impact the C_p negatively, as the inertia becomes too large.



(a) Overlap [108]



(b) End plates

Figure 3.2.4: Illustration of overlap and end plates.

3.3 Computational Fluid Dynamics of Turbines

Marsh et al. [109] investigated the influence two different turbulence models, $k-\omega$ SST and Baseline-Reynolds Stress Model, have on the results, using both a 3D and 2D domain. Moreover, they compared the performance of using wall functions as opposed to fully resolving the boundary layer in a 3D case using $k-\omega$ SST. Two types of straight bladed Darrieus turbines were used. They found that a 3D case with $k-\omega$ SST and a fully resolved boundary layer gave the most accurate results compared to experimental results, while simulating for 20 hours per revolution. Although being computationally efficient, 1.5 hours simulation time per revolution, the two-dimensional $k-\omega$ SST cases with fully resolved boundary layers showed poor correspondence to the experimental data, mainly because it could not capture the effect of the struts and ends of the turbine. Here, for most TSR, the power coefficient was significantly overestimated. Simulations using wall functions were only made in 3D for $k-\omega$. As a result of a lower mesh resolution requirement, simulation times were lowered to 6 hours and 40 minutes, however, this came at the cost of inaccurate predictions of C_p. This inaccuracy was caused by poor prediction

of separation at low angular velocities.
In [110], Kacprzak et al. performed 2D simulations on three different Savonius wind turbines: the classical Savonius, a Bach-type, and an elliptical Savonius. Both the classical and the elliptical profiles had an overlap ratio of 0.15, whereas the Bach type had no overlap. While they state using the $k - \omega$ SST turbulence model, they do not specify which pressure-velocity coupling algorithm they use. Regarding the turbulence, a key point of their project was to investigate the likelihood of the flow transitioning between laminar and turbulent in the blade boundary layer. If it was high then a substantial mesh refinement would be necessary, as wall functions are inaccurate in the viscous sublayer. To avoid the need of interpolation at the sliding mesh interface, the time step was chosen such that it rotated exactly one mesh each step. This is only possible since they are studying the turbines with a forced rotation, where the angular velocity is constant at all times. After conducting a grid independence test, they ended up with an average y^+ value of less than three. All three of the turbines achieved a maximum power coefficient at a TSR of 0.8, giving approximately 0.15, 0.18, and 0.17 for the classical-, Bach-, and elliptical Savonius turbines. Without resolving the viscous sublayer, the simulations overestimated the power output.
In [24] the authors attempted to improve the performance of a low cut-in speed, vertical axis, hybrid Savonius-Darrieus tidal turbine using OpenFOAM in a 2D case. Tests were also made on the stand-alone Savonius and Darrieus turbines. For the Savonius rotor, a two-stage setup was used, with an overlap ratio of 0.298, and an aspect ratio of 2. The inlet velocity was set to 0.5 m s^{-1}. They utilised the Unsteady RANS SST $k - \omega$ turbulence closure model, solved using the SIMPLE algorithm. For their mesh, an unstructured, low density, triangular mesh was created upstream of the turbine, whereas a denser, progressive mesh was used in the blade vicinity and wake region. For the boundary layer, a y^+ value of 1 was used. Moreover, a mesh refinement was also made in the sliding interface between the stationary fluid domain and the rotating domain. Comparisons were also made with existing literature, where they were able to achieve a marginally higher power coefficient, due to the 2D numerical solutions not taking into account the lifted vortex structures that are present in 3D cases.
Mosbahi et al. [32] used ANSYS FLUENT 17.0 to conduct a performance study of a hydrokinetic three bladed helical Savonius turbine, with- and without a deflector. Experiments were also conducted in an irrigation channel with a 3D printed model. Four different RANS turbulence coupling models were used, namely the RNG $k - \varepsilon$, Realizable $k - \varepsilon$, SST $k - \omega$, and the transition SST. Pressure-velocity coupling was done with the SIMPLE algorithm. Contrary to the literature of the current report, in the literature review of Moshabi et al. They found that the Realizable $k - \varepsilon$ model was best suited for this kind of simulation. Due to the complexity of the geometry, an unstructured mesh was created, with a fine mesh in the rotating zone. A prismatic mesh was applied for the rotor blade boundary layer to better model the behaviour in this region. For this, $y^+ < 1$ was used to determine the height of the first layer, using 20 prismatic layers and a growth rate of 1.2. The deflector setup was composed by a NACA 0020 airfoil in conjunction with a straight ramp. Different combinations of distance between the two deflector parts and the angle relative to the incoming flow were tested. The angle of the airfoil was always equal to that of the ramp, although in the opposite direction. Without the deflector, a maximum power coefficient of 0.125 was found at a TSR of 0.7, whereas with the deflector at an angle of 30°and a distance

between the two parts of 204 mm (approximately 1.121 times the diameter of the turbine), the maximum power coefficient increased to 0.14. These results were validated with experimental data. Both the experiments and simulations were conducted using a velocity of 0.86 m s^{-1}. A higher C_p might be achieved by using an overlap between the blades, using a different number of blades, or a different profile for the blades.
Khanjanpour and Javadi [111] used ANSYS-Fluent in combination with the Taguchi approach to perform the CFD analysis and optimization of a horizontal axis tidal turbine. The mesh around the turbine itself was unstructured, whereas the computational area had a structured mesh. Meshing was done using ICEM 2019. A sliding mesh was used to simulate the turbine movement. the mesh was defined as sliding to allow simulation of movement. Furthermore, the PISO algorithm was used for pressure-velocity coupling, in combination with the $k - \omega$ SST turbulence model. The Taguchi method was used to optimize blade size, number of blades, and hub radius- and shape, resulting in a 10% increase in the power coefficient. They found that the number of blades had the most influence on C_p, followed by blade size, hub radius, then hub shape.
The optimized position of a barrier for the hydrokinetic, semi circular Savonius rotor was investigated in [112]. They studied a two bladed turbine with an overlap ratio of 0.15 and blade thickness of 1 mm in a two-dimensional simulation, investigating several barrier configurations. Figure 3.3.1 illustrates the different configurations, where a shield is attached between the end plates, at a diameter of 1.1 times the turbine diameter. For turbulence modeling they used $k - \omega$ SST, with a fully resolved boundary layer, $y^+ < 1$. Furthermore, the inlet was specified as a velocity inlet, with a constant flow of 7 m s^{-1}, and a pressure outlet with a reference value of zero. Moreover, the free surface slip condition was used on the top and bottom boundaries. Through their simulations, the best combination was that of section 4, 5, and 6, increasing the maximum generated power by 18%. This type of configuration seems to be easily installed, while also being area efficient.

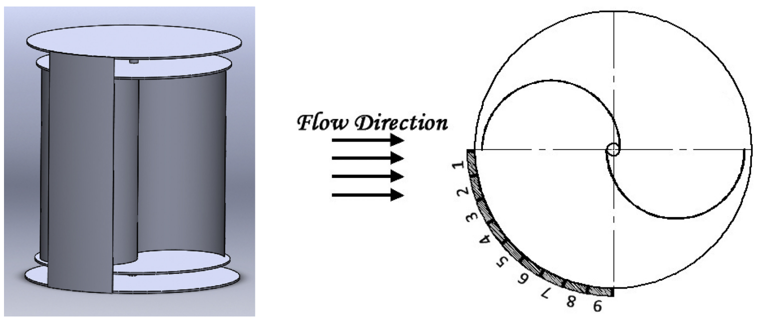


Figure 3.3.1: The barrier utilised in [112], tested with different configuration combinations (1-9) to optimize its positioning.

A three dimensional CFD study was performed on a three bladed horizontal axis ocean current turbine both with- and without a deflector by Maldar et al. [113]. This deflector was positioned in front of the turbine, acting like a ramp. Both a flat- and a curved ramp was simulated, at different angles, with the optimal configuration being a flat ramp at 25°. With the deflector, an optimal C_p and C_T of 0.28 and 0.238, respectively, was found, compared to the turbine without augmentation at 0.195 for both C_p and C_T.
Salleh et al. [114] also investigated the Savonius turbine, employing an augmentation technique

consisting of two flat deflector plates. One positioned in front of the returning blade, and one above the turbine, as shown in Figure 3.3.2. This Savonius turbine used an overlap ratio of 0.109, end plates with a diameter of 1.1 times the turbine diameter, and both the deflectors had lengths 0.556 times the diameter of the turbine. Without the augmentation setup, the maximum power- and torque coefficient was 0.13 and 0.16, respectively, whereas when both the deflectors were angled at 30°, C_p increased to 0.21 and C_T to 0.24.

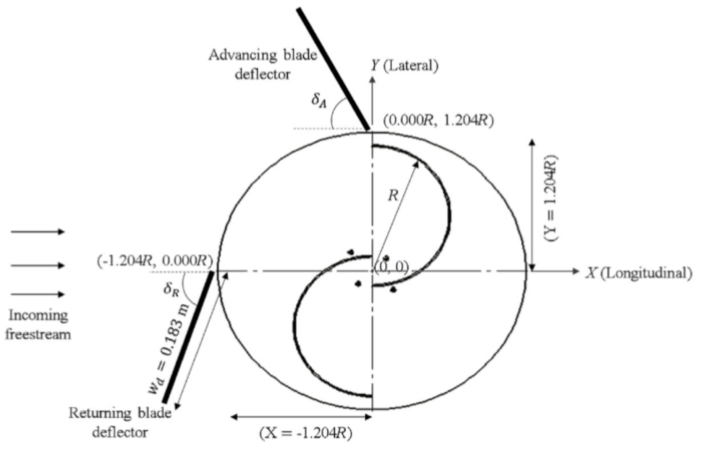


Figure 3.3.2: Geometrical setup of the turbine and deflector blades from [114].

In [115], the authors analysed three hydrokinetic Savonius type blade profiles: both a two- and three bladed semi circular rotor, and an elliptic rotor. All three turbines used an aspect ratio of one, overlap ratio of 0.15, diameter of 0.25 m, blade thickness of 1.3E-03 m, and end plate diameter of 1.1088 timed the turbine diameter. Moreover, all of the blades had a chord length of 0.144 m. The elliptic turbine was made such that the semimajor- and semiminor axis had a 3/2 relation, with a sectional cut angle of 47.5°, ensuring the proper chord length was applied, as shown in Figure 3.3.3. Their elliptic turbine had similar dimensions as the one found in [106], which was recommended by [102]. For turbulence modeling, the $k - \omega$ SST model was used, whereas pressure-velocity coupling was done using SIMPLE. Mesh refinements were made, such that $y^+ < 1$, moreover, a mesh independence study was carried out. The inlet was specified as velocity inlet, with an incoming flow of 0.8 $\mathrm{m\,s^{-1}}$, while the outlet was defined as a pressure outlet. Contrary to the results found in [102], [106], the elliptic profile performed worse than the two bladed semi circular profile, with the three bladed version showed the worst performance, showing C_p values of 0.20, 0.28, and 0.17, respectively. The corresponding C_T for the maximum C_p was found to be 0.26, 0.31, and 0.24, respectively.

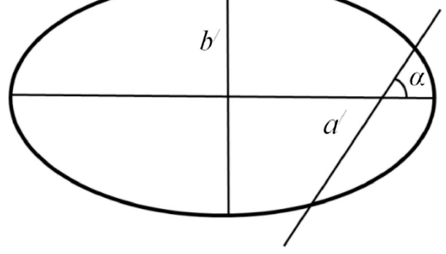


Figure 3.3.3: Sketch of the sectional cut for the elliptic Savonius turbine in [115].

An integrated surrogate optimization to maximize the power coefficient of an elliptical Savonius wind turbine was carried out in [116]. Here, the overlap ratio, cut angle, and semimajor axis varied from 40°to 90°, 0.1 to 0.3, and 0.14 to 0.2 m, respectively. Both the rotor height and diameter were kept constant at 0.5 m, with an end plate diameter 1.1 times the rotor diameter. Additionally, the cut angle was made at a point M, such that the length from the center to this point was equal to 0.54 times the length of the semimajor axis, similar to the one in Figure 3.3.3. Their results suggest that the optimal overlap ratio should be between 0.14 and 0.15 and the cutting angle should stay between 40°and 50°. Compared to other elliptic

Chapter 4

Research Questions

In this chapter, the theory and literature review will be boiled down to a set of research questions that define the goal of this thesis.
Due to a time limitation, the mesh for this project can not safeguard a y^+ value less than 1. Therefore, wall functions will be implemented to calculate the flow behaviour close to walls. Furthermore, simulations will be performed in 2D, due to the same reason, which inevitably will cause some flow phenomena to not be captured.
As previously discussed, most turbine ideas are of the horizontal axis kind, with several bearing heavy resemblance to commercial wind turbines. Nevertheless, a large amount of research has been put into the Savonius type turbine, with optimization of the blade profile and the surrounding augmentation techniques being key aspects in these. One of the blade profiles that has gotten some attention is the elliptic profile, where simulated power coefficients have been between 0.1296 in [116] to 0.33 in [106]. The current thesis intends to investigate this elliptical turbine, using the data found in [102], [106], [115], [116] as reference. Moreover, the standard, semi circular Savonius will also be taken into consideration for comparison. Regarding augmentation techniques, the author of the current thesis took interest in the design by Alizadeh et al. [112], due to its simple nature and seemingly easy installment and maintenance. Additionally, the advancing blade deflector found in [114] was employed to have deflectors both for the advancing- and the returning blade.
Thus, the resulting research questions for this thesis are:

- How does a two dimensional CFD simulation with $y^+ \in [30,500]$ for the semi circular- and elliptical hydrokinetic Savonius turbines compare to results found in literature for similar turbines?
- How does the performance of these turbines change with the added augmentation techniques found in [112] (barrier) and [114] (advancing blade deflector)?
- How can these designs be further improved?

Chapter 5

Method

This chapter will go into detail on the important steps of the procedure, from creating the geometry in SOLIDWORKS, to meshing in Gmsh, then finally to the simulation in OpenFOAM. The four geometries to be studied are recapitulated in Table 5.0.1. For the rotation of the turbine, the AMI method was chosen, since, in comparison with the overset mesh, it seems to be more frequently used in literature.

Table 5.0.1: Turbine geometries assigned to case numbers.

Turbine	**Case number**
Semi circular without augmentations	1
Semi circular with augmentations	2
Elliptic without augmentations	3
Elliptic with augmentations	4

5.1 SOLIDWORKS

5.1.1 Blade Profiles

Both Savonius-type turbines were made with an arbitrary diameter, D_t, of 2 meters, chord length of $0.575D_t$, and blade thickness of $0.015D_t$. Moreover, based on literature, an OR of 0.15 was used with two blades. The chord length was chosen to safguard an overlap ratio of 0.15, whereas the thickness was based on the relationship between turbine diameter and blade thickness found in literature. The elliptic blade was made using Figure 5.1.1, where the ratio between OA and OB was 3/2, the distance OP was 0.54×OA, and the angle, θ, was 47.5°.

5.1.2 Augmentation Techniques

Regarding the augmentation techniques found in [112] and [114], they were positioned along the edge of the supposed end plates, at 1.1 times the turbine diameter. Recalling Figure 3.3.1, the optimal placement of the barrier in [112] was from section 4 to section 6, corresponding to an arc of 30°along the end plate. Section 4 started at an angle of -30°relative to the upstream horizontal, whereas section 6 ended at -60°. The deflector in [114] was angled 30°with the horizontal, which was reported to give the best performance, with a length of $0.55623D_t$. Table 5.1.1 summarizes

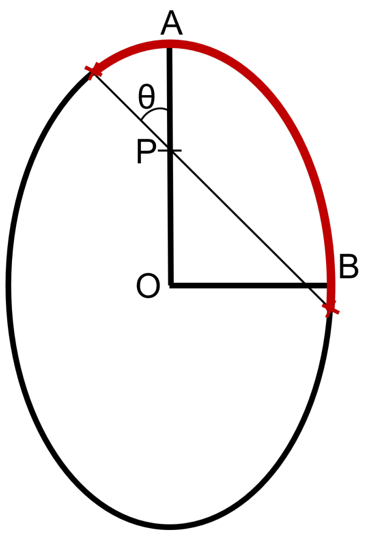


Figure 5.1.1: Sketch of how the elliptic blade profile was made, based on the description found in [106], [107].

the geometrical parameters used in this report.

Table 5.1.1: Geometrical parameters of the two Savonius-type turbines.

		Semi Circular	**Elliptic** [106], [107]
Number of blades	[-]	2	2
Diameter turbine, D_t	[m]	2	2
Chord length	[m]	$0.575D_t$	$0.575D_t$
Semimajor axis (OA)	[m]	-	$0.400495D_t$
Semiminor axis (OB)	[m]	-	$0.267D_t$
Center to sectional cut (ellipsis, OP)	[m]	-	0.54×OA
Sectional cut angle (θ)	[°]	-	47.5
Aspect Ratio	[-]	1	1
Overlap Ratio	[-]	0.15	0.15
Blade thickness	[m]	$0.015D_t$	$0.015D_t$
Diameter to augmentations (End plates)	[m]	$1.1D_t$	$1.1D_t$
Shield and deflector thickness	[m]	$0.015D_t$	$0.015D_t$
Shield arc angle	[°]	30	30
Deflector length	[m]	$0.55623D_t$	$0.55623D_t$

5.1.3 Splitting the Domain

Thereafter, a rectangle, concentric to the turbine, with a height of $10D_t$ and length of $30D_t$, was sketched. Using this length, the distance between the turbine and the end of the domain would be more than 10 times its diameter, which should give the flow time to develop enough for accurate results, as discussed in Chapter 2.4.1. A surface plane was made from this sketch, which then represents the fluid domain. The AMI boundary was sketched as a circle concentric to the turbine. For the cases without augmentations a diameter of $1.5D_t$ was used for this. Moreover, two additional circles with diameters $1.875D_t$ and $1.125D_t$ were added to ensure a simple way to accurately control the mesh in the nearby vicinity of the AMI. With the added augmentation structures, however, the AMI boundary was confined to be $1.05D_t$ to ensure these additional structures did not rotate, with the surrounding circles being $1.025D_t$ and $1.075D_t$. To create regions for a multiblock grid, several split lines were sketched as indicated in Figure 5.1.2. The square surrounding the turbine and AMI boundary has a side length of $2.5D_t$.

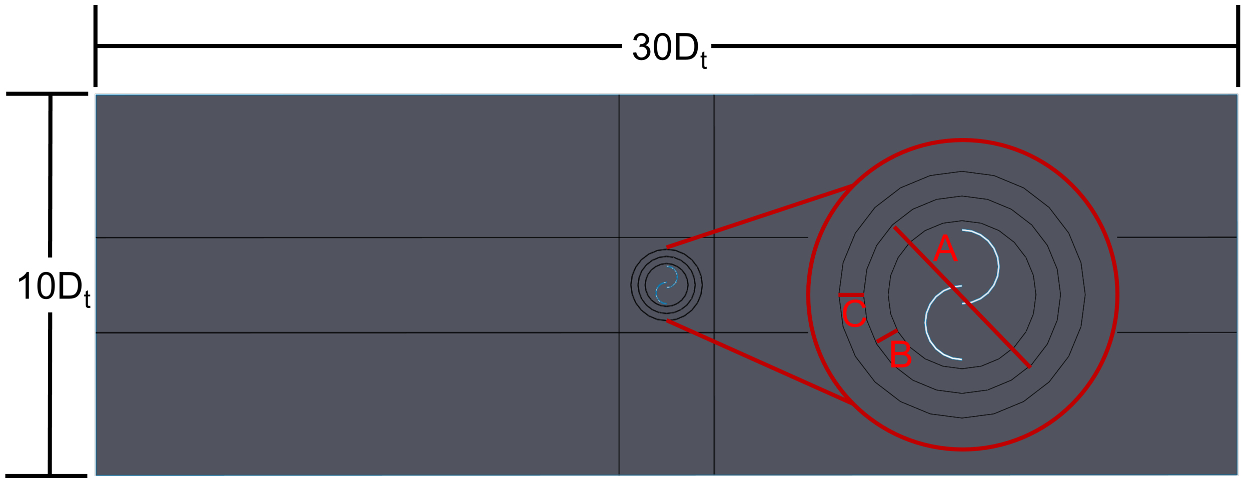


Figure 5.1.2: Fluid domain split up for multiblock mesh regions for the semi circular blade without augmentations. A=$1.5D_t$, B=C=0.375 m.

5.1.4 Mass Properties

The turbine sketch was further utilised to calculate its mass properties. This was done by extruding it by 2 meters to emulate an AR of 1, then going to Tools - Evaluate - Mass properties. Here, the density was set to 950 $\mathrm{kg\,m^{-3}}$, resulting in the values noted in Table 5.1.2.

Table 5.1.2: Mass and principle moment of inertia (x, y, z) after extruding the turbine profiles by 2 meters.

		Semi Circular	**Elliptic**
Mass	[kg]	200.56	174.48
Principle Moment of Inertia	[$\mathrm{kg\,m^2}$]	(81.05, 99.17, 151.82)	(63.69, 78.74, 131.37)

5.1.5 Exporting the Domain

Lastly, to export the domain to the meshing software, two STEP files had to be created. One where only the faces within the AMI boundary was selected, and one where only the faces outside the AMI boundary was selected. This would represent the rotating- and stationary region, respectively. These two STEP files were subsequently merged into the meshing software.

5.2 Gmsh

Gmsh creates a .geo file where it stores the script describing the mesh. This script is shown in Appendix D, using the semi circular Savonius with shield and deflector as an example. When merging the STEP files into Gmsh, the "Merge" function was used as shown in the script. For this, the .geo file had to be in the same folder as the STEP files. Note, that to import STEP files into Gmsh, the OpenCASCADE factory must be used. In the options tab, under mesh, the "Frontal-Delaunay for Quads" 2D algorithm was chosen, along with the "Delaunay" 3D algorithm, and the

"Blossom" 2D recombination algorithm. Additionally, the setting for recombining all triangular meshes was turned on, no subdivision algorithm was used, and both the smoothing steps and the element size factor was set to one. Recombination of triangular meshes would force triangular shapes to be combined into quadrilateral cells whenever possible, which was the first step of achieving a structured grid.
Thereafter, transfinite points, describing the resolution of the grid, were defined for all the curves, with a larger number of points being used for regions of interest, where the mesh should be finer. Appendix D includes information on the transfinite mesh made. The regions with the finest meshes were the wall boundaries and the arbitrary mesh interface. At the turbine, shield, and deflector, the mesh was made such that the first cell height corresponded to a y^+ of 300 for a velocity of 1.5 m s^{-1}. Note, that the velocity in the nearby vicinity of the turbine is subject to vary greatly depending on the position of the blades. This means that the y^+ value will also change continuously, such that despite calculating the theoretical height of the first cell to achieve a y^+ of 300, this might not give the desired results in practice. Table 5.2.1 contains the results of calculating the height of the first cell adjacent to the wall boundaries. An assumption that the results would not be significantly altered by using a coarser mesh at the bottom and top boundaries was made, due to being placed four times the turbine diameter above- and below the turbine. This was done to decrease simulation time.

Table 5.2.1: Calculations required to find the first cell height corresponding to $y^+ = 300$ at the wall boundaries, using Eqs. 2.22, 2.24, 2.25, 2.26, and 2.27

		Top/Bottom	**Semi Circular**	**Deflector**	**Shield**	**Elliptic**
Re	[−]	4.50E+07	5.28E+06	1.67E+06	1.67E+06	2.16E+06
$C_{f,x}$	[−]	2.08E-03	2.84E-03	3.43E-03	3.43E-03	3.28E-03
τ_w	[kg m^{-1} s^{-2}]	2.40	3.28	3.95	3.95	3.79
u^*	[m s^{-1}]	4.84E-02	5.66E-02	6.21E-02	6.21E-02	6.08E-02
y_H	[m]	1.24E-02	1.06E-02	9.66E-03	9.66E-03	9.87E-03

The second step to achieve a structured grid was to make the surfaces transfinite. This was not done for the region between the circle around the AMI and the square, nor was it done for the region around the turbine, inside the inner circle. As a result, the mesh in these regions would become hybrid meshes, recalling the discussion about this in Chapter 2.3.1. Figures 5.2.1 and 5.2.2 provide a visualisation of the mesh for the semi circular Savonius turbine with the shield and deflector, both as an overview of the entire domain, and a zoomed in image of the region within the square. For the freestream region outside of the square, the mesh is fully cartesian, using the "Bump" function to make the cells close to the borders smaller than those in the middle. The reason for this was to have a good smoothness across the multiblock domains, as the cells adjacent to each other at these interfaces would otherwise have large differences in their size.

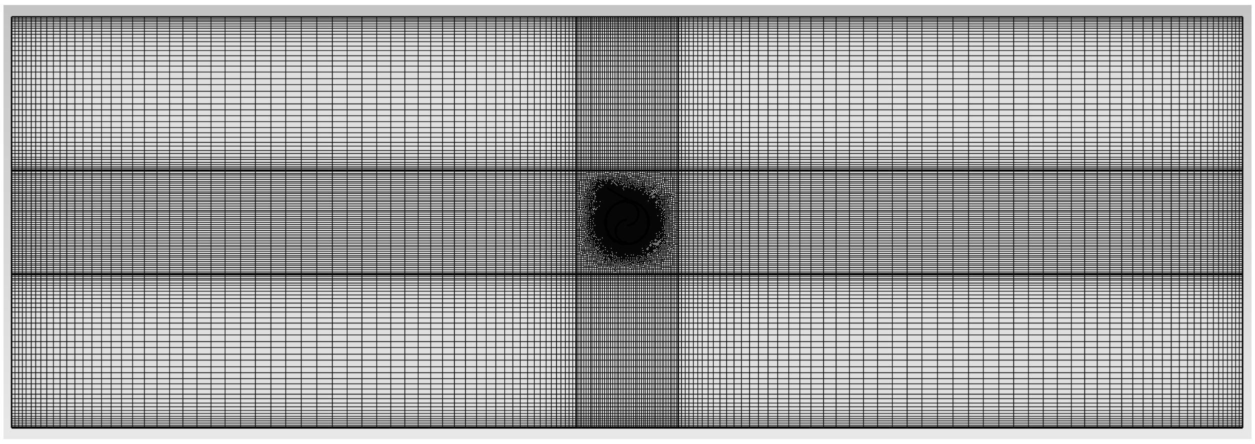


Figure 5.2.1: Overview of the mesh for the semi circular Savonius blade profile with augmentations.

Around the AMI, the mesh is curvilinear. Moreover, in this region, the cells become progressively smaller towards the interface. This was done to provide more accurate results in the interpolation region.

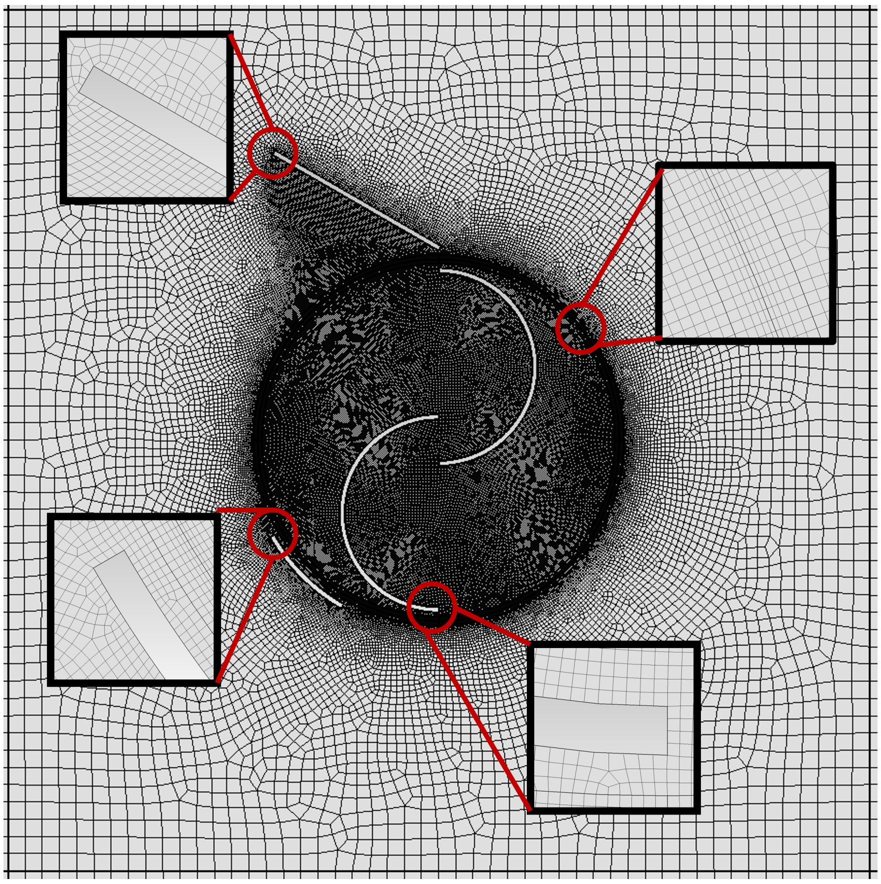


Figure 5.2.2: Enlarged image of the mesh close to the deflector, AMI, turbine blade, and shield.

Two dimensional simulations in OpenFOAM still require the domain to have one layer in the third dimension. Therefore, a mesh extrusion of 2 m with only one layer was made on all the faces of the domain. It is appreciable that neither the points nor the curves were extruded, as this would create duplicates of them. Following the extrusion, all the physical surfaces and volumes were defined, specified as shown in Figure 5.2.3 for the semi circular blade profile with augmentations. Since two STEP files were merged into the software, care had to be taken when defining the transfinite surfaces and physical boundaries in the vicinity of the AMI. In the figure, ami1 was defined as the surface belonging to the rotating domain. Conversely, ami2 belongs to the patch in the stationary domain. When saving the mesh, the file type was specified as .msh, using version 2 of the ASCII, while leaving the two boxes, "Save all elements" and "Save parametric coordinates", unchecked.

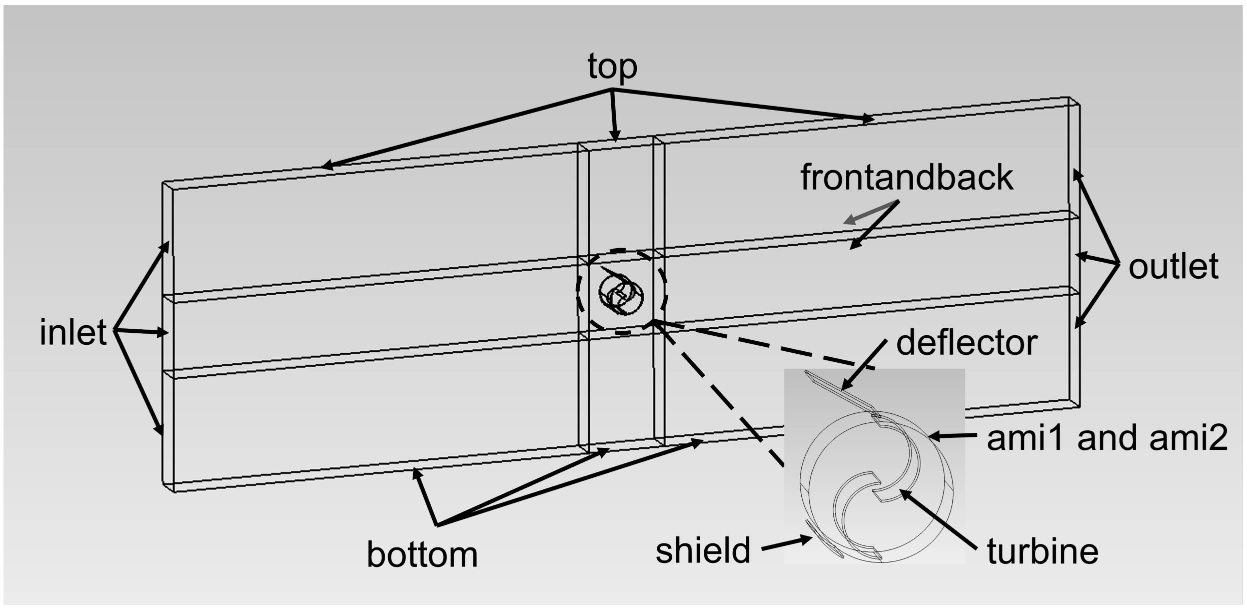


Figure 5.2.3: Physical groups of the domain for the semi circular Savonius rotor with augmentations.

5.3 OpenFOAM

Throughout this chapter, whenever a file is referred to, Figure. 5.3.1 can be used to identify the location of this file in the case directory. Moreover, in Appendix E, all the files can be found for the semi circular Savonius turbine with augmentations as an example. The files for the other three configurations will not be included as they are almost identical to this one. Nevertheless, the values and changes that do occur between them will be described throughout this methodology. Initially, the case setups were based on Dr. Tobias Holzmann's tutorial cases for the Kaplan turbine [117] and vertical axis wind turbine [118]. Dr. Holzmann is a prominent user of OpenFOAM, having published several tutorials employing the software, thus his work was regarded as trustworthy. Additionally, one of OpenFOAM's own tutorials, the wingMotion tutorial for incompressible flows using the RANS equations with PIMPLE, was used as inspiration. Based on the latter tutorial, a decision to first use SIMPLE on a static case, without a rotating turbine, was made. This was done to hopefully acquire more accurate results of the flow field, which then were mapped to the transient case. Also, since the flow field would fully develop in the static case, the transient case could decrease the simulation time, as quasi stable conditions would be reached sooner.

5.3.1 Importing the Mesh

Before simulations could be run, the mesh from Gmsh had to be imported into OpenFOAM, done using the command "gmshToFoam FILENAME". This would create the polyMesh folder, including all the files describing the mesh, most importantly the boundary file. The mesh could then be viewed in Paraview to ensure that it had been properly imported before proceeding. Moreover, the "checkMesh" command was used, providing information on skewness, non-orthogonality, and the aspect ratio of the cells. Upon importing the mesh, it also had to be scaled down from millimeters, used in Gmsh, to meters, being the SI unit employed in OpenFOAM. Here, the command "transformPoints "scale=(1e-3 1e-3 1e-3)"" was used, which was an important step before simulations could commence. Furthermore, to increase simulation speed, the mesh was renumbered to reduce bandwidth, with "renumberMesh -overwrite".

5.3.2 Boundary- and Initial Conditions

Whenever a new mesh was imported, the boundary types for the physical groups in the boundary file, Appendix E.4, defaulted to "patch". However, this was only desired for the inlet- and outlet boundaries. Therefore, the top-, bottom-, turbine-, shield-, and deflector boundaries were all defined as walls, whereas the two AMI boundaries were defined as cyclicAMI. As this was a two-dimensional study, the "frontandback" group was set to type "empty", indicating that a solution was not required in the cells normal to this boundary. In addition to defining the AMI boundaries as cyclicAMI, the neighbouring AMI patch had to be defined for each of them, such that ami2 had ami1 as its neighbour and vice versa. The interpolation method was chosen as "faceAreaWeightAMI", corresponding to the weighted interpolation method mentioned in Chapter 2.3.3.

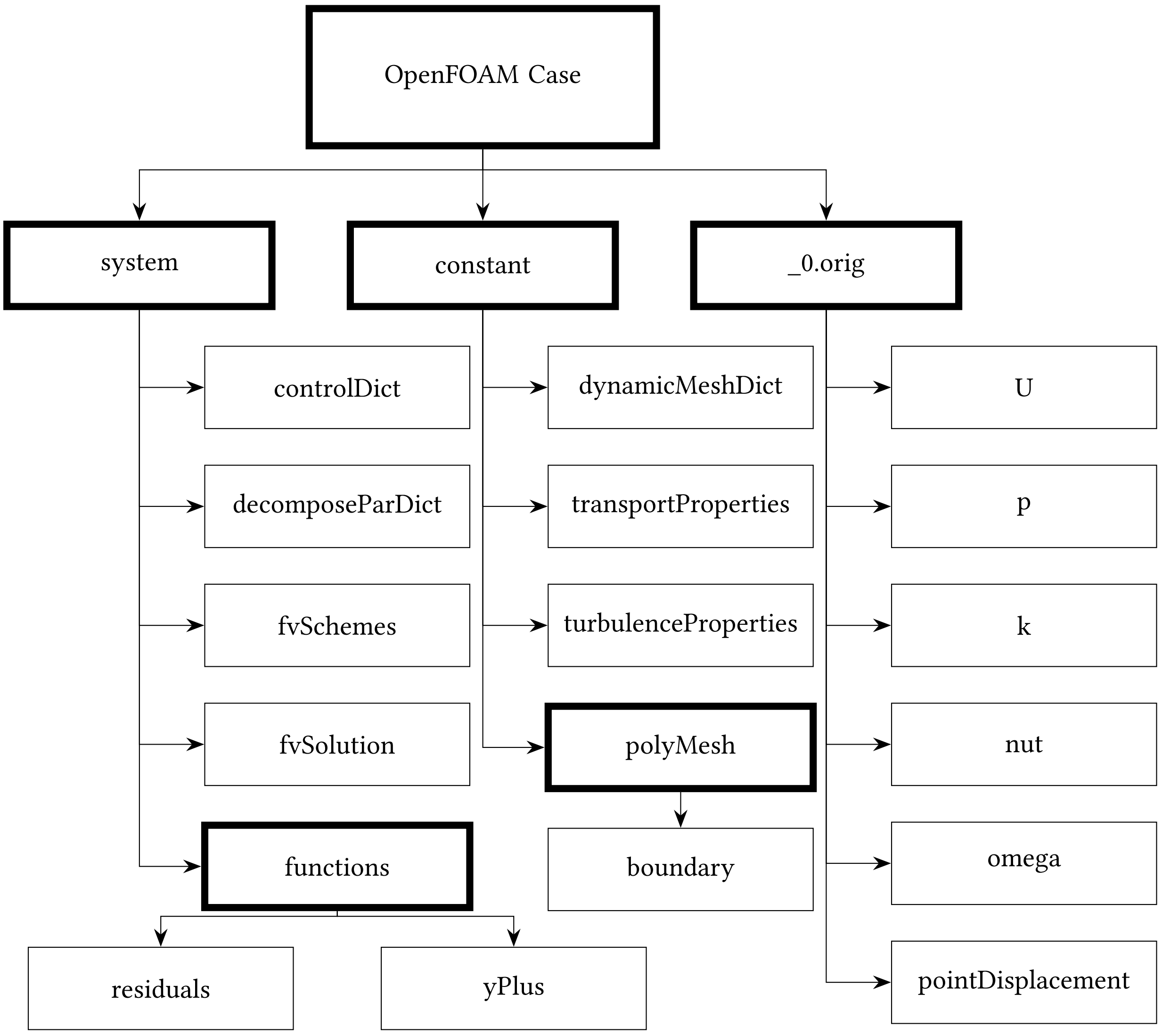


Figure 5.3.1: File structure of a transient OpenFOAM case

Velocity The inlet- and internal field velocity was set to 1.5 m s^{-1}, with the outlet being defined as an "inletOutlet", in case there was an occurrence of reverse flow. However, due to the length of the domain, reverse flow was unlikely to be a problem, so the inletOutlet was only defined as a precaution. Since the turbine rotates it was given the moving wall velocity type, in accordance with the discussion of moving walls in Chapter 2.4.1. Note, that even though the top boundary was defined as a wall, it was given the slip condition to simulate it as a free surface flow. All other wall boundaries were given the no slip condition.

Turbulent Kinetic Energy By setting the inlet to turbulentIntensityKineticEnergyInlet, Eq. 2.32 was used, recalling the discussion in Chapter 2.4.2. Here, a turbulent intensity of 0.1 was chosen, based on the values found in literature. For the walls, tests were made using both the kLowReWallFunction and the kqRWallFunction, with no significant difference between them.

Specific Turbulence Dissipation Rate Eq. 2.35 was used to estimate the inlet- and internal field value, with the inlet using the turbulentMixingLengthFrequencyInlet type to automatically employ this equation. Using Eq. 2.37, a hydraulic diameter L_D of approximately 3.64 m was found, based on the inlet height and width of 20 m and 2 m, respectively. A value of 0.085 was chosen for C, being the average of what was found in literature. All walls used the omegaWallFunction, with initial values for k and ω shown in Table 5.3.1.

Table 5.3.1: Boundary conditions for k and ω, calculatd using Eqs. 2.32, 2.34, and 2.35

k_{BC} [m^2 s^{-2}]	ω_{BC} [s^{-1}]	$\omega_{turbine/aug}$ [s^{-1}]	$\omega_{top/bot}$ [s^{-1}]
3.38E-02	1.09	109.08	8.18

Pressure For these simulations, the outlet was specified as the pressure reference for the domain, with a uniform value of zero. The inlet and wall boundaries were given the zero gradient condition.

Turbulent Eddy Viscosity As discussed in Chapter 2.4.5, there are several wall functions to choose from for ν_t. Assuming the coating on the turbine is sufficient enough to make the surface hydrodynamically smooth, as discussed in Chapter 2.3.4, attention was averted from wall functions made for rough walls. Based on the tutorial cases from Dr. Holzmann and OpenFOAM, the nutkWallFunction was chosen.

Point Displacement As the name suggests, the point displacement describes the displacement of the points in the mesh. The initial condition for this was naturally set to zero, as no movement of the mesh had occurred at time zero.

5.3.3 Constant Folder

Dynamic Mesh Dictionary This file was only employed in the transient simulations.
To specify the employment of a flow induced turbine, the motion solver "sixDoFRigidBodyMotion" (six degrees of freedom rigid body motion) was chosen. For this, the turbine patch had to be specified as the solid body subject to rotation. The inner- and outer morphing distances had to be set to values that prevented excessive morphing. In cases where this was done poorly, the mesh close to the AMI border would morph, such that large gaps in the mesh would emerge, causing the weighted interpolation method to give floating point exceptions.
The density of water was specified as 1025 kg m^{-3}, while the mass and moment of inertia corresponded to the values calculated in SOLIDWORKS, from Table 5.1.2. Moreover, the center of mass and center of rotation was specified, based on the turbine coordinates in Gmsh.
Since this study involves a flow induced rotation of the turbine, the velocity, acceleration, angular momentum, and torque were all set to zero. Also, an acceleration relaxation coefficient of 0.05 was chosen. Values above 0.1 would often instigate large discrepancies between expected- and simulated angular velocities.
Rotational movement was fixed around the z-axis, and translational movement was fixed at the center of mass. Lastly, the damping coefficient, B_r was specified as a restraint, using the following values: 200, 800, 900, 1000, 1200, 2000 kg m^2 s^{-1}. These values were chosen using the trial and error method.

Transport Properties The kinematic viscosity, ν, was defined in the "transportProperties" file, which for water is 1E-06 $m^2\ s^{-1}$. Additionally, the transport model was set to Newtonian, assuming a constant viscosity.

Turbulence Properties In this file, the turbulence closure model was specified as $k - \omega$ SST.

5.3.4 System Folder

Control Dictionary In this dictionary, the pressure-velocity coupling algorithm was specified. For the static case, "simpleFoam" was used (SIMPLEC), whereas for the transient case, pimpleFoam (PIMPLE) was used. To achieve a fully developed, converged flow, the static case would simulate 3000 seconds, with a time step of 1 second. The transient case would simulate 50 seconds, such that the turbine could reach a quasi-stable behaviour. An initial time step of 0.0001 seconds was set, however, the simulation was allowed to adjust this automatically based on the maximum Courant number of 5. Recalling the theory in Chapter 2.5, the Courant number in PIMPLE could be higher than 1, with values up to 10 giving accurate results. Inside the control dictionary, the functions for the residuals and y^+ were called upon, providing the residuals for U_x, U_y, k, ω, and p, and the minimum, maximum, and average y^+ values for all the wall boundaries.

Parallel Decomposition Dictionary Simulations were run on two AMD EPYC-Milan 2 GHz processors with 16 cores each, having 64 GB of RAM allocated to the system. To enable all 32 cores to work on the simulations in parallel, the case had to be decomposed using the "decomposeParDict". In this dictionary, the hierarchical method was used, splitting the domain into 32 (8x4x1) subdomains. This was upgraded to four processors and 128 GB RAM, however, running each simulation with only 12 cores seemed to be the most efficient.

Finite Volume Schemes In this file, the numerical schemes used to perform mathematical operations are specified. Table 5.3.2 specifies which operation each of the categories perform, while the setup used for this file can be found in Appendix E.7.

Table 5.3.2: Description of what operations the different numerical schemes perform [119], [120]

Category	**Performed Operation**
ddtSchemes	First and second order time derivatives
gradSchemes	Gradients ∇
divSchemes	Divergences $\nabla \cdot$
laplacianSchemes	Laplacians ∇^2
interpolationSchemes	Interpolations between points
snGradSchemes	Components of gradients normal to a cell face
wallDist	Calculates the distance to a wall

Finite Volume Solution This file specifies which solvers should be used for the equations, the algorithm used, as well as the relaxation factors and tolerances for all the parameters.

The solver used for the pressure equations was the generalised geometric-algebraic multi-grid solver (GAMG), whereas for the velocity, turbulent kinetic energy, and specific turbulence dissipation rate the smooth solver was employed. This was copied from the aforementioned wingMotion tutorial case. All solvers utilised the Gauss-Seidel smoother, although this is strictly only used by the smooth solver.

Obtaining the best values of tolerances and relaxation factors was done by simulating the static case of the semi circular Savonius turbine without augmentations using three different coefficient combinations. The combination yielding the lowest, most stable residuals, as well as providing the quickest convergence, was chosen. This combination, see Appendix E.8, was employed for all simulations.

For the PIMPLE algorithm, five outer correctors and three inner correctors were specified, using the SIMPLEC algorithm for the outer correctors. This means that for each outer loop, three inner loops would be made, resulting in each time step performing a total of 5 outer loops (SIMPLEC) and 15 inner loops (PISO).

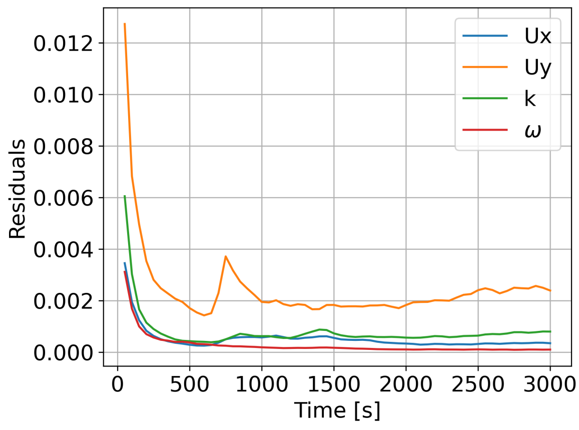
Chapter 6

Results and Discussion

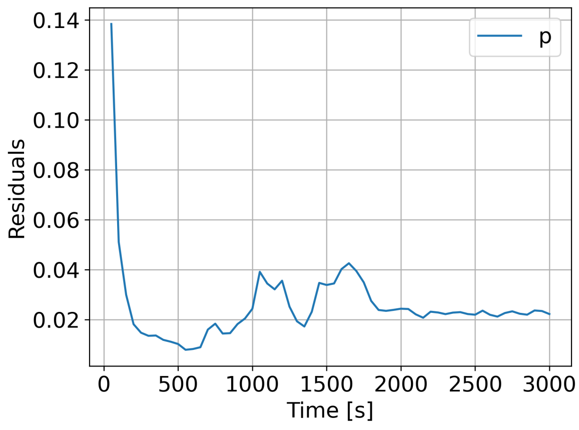
6.1 Static Cases

Recalling the methodology, to decrease simulation time, all turbines were initially run as static cases until the flow had fully developed. The development of the velocity field for the static SIMPLEC simulation using case 1 can be found in Appendix H. Running this case took 94 seconds using 32 processor cores, reducing the simulation time needed for the transient cases by almost 40%. The total simulation time needed for the four cases was approximately 105 hours when implementing the initial static simulation, when counting only the simulations used in this chapter. Thus, assuming all cases reduced their simulation time by 40%, the total time saved is estimated to 70 hours.

The residuals for the static case 1 are provided in Figure 6.1.1. All the residuals seem to stabilize within the 3000 seconds simulated. However, the observed values of the residuals are high, which should be rectified by setting stricter residual criteria.



(a) Residuals for k, ω, U_x, and U_y

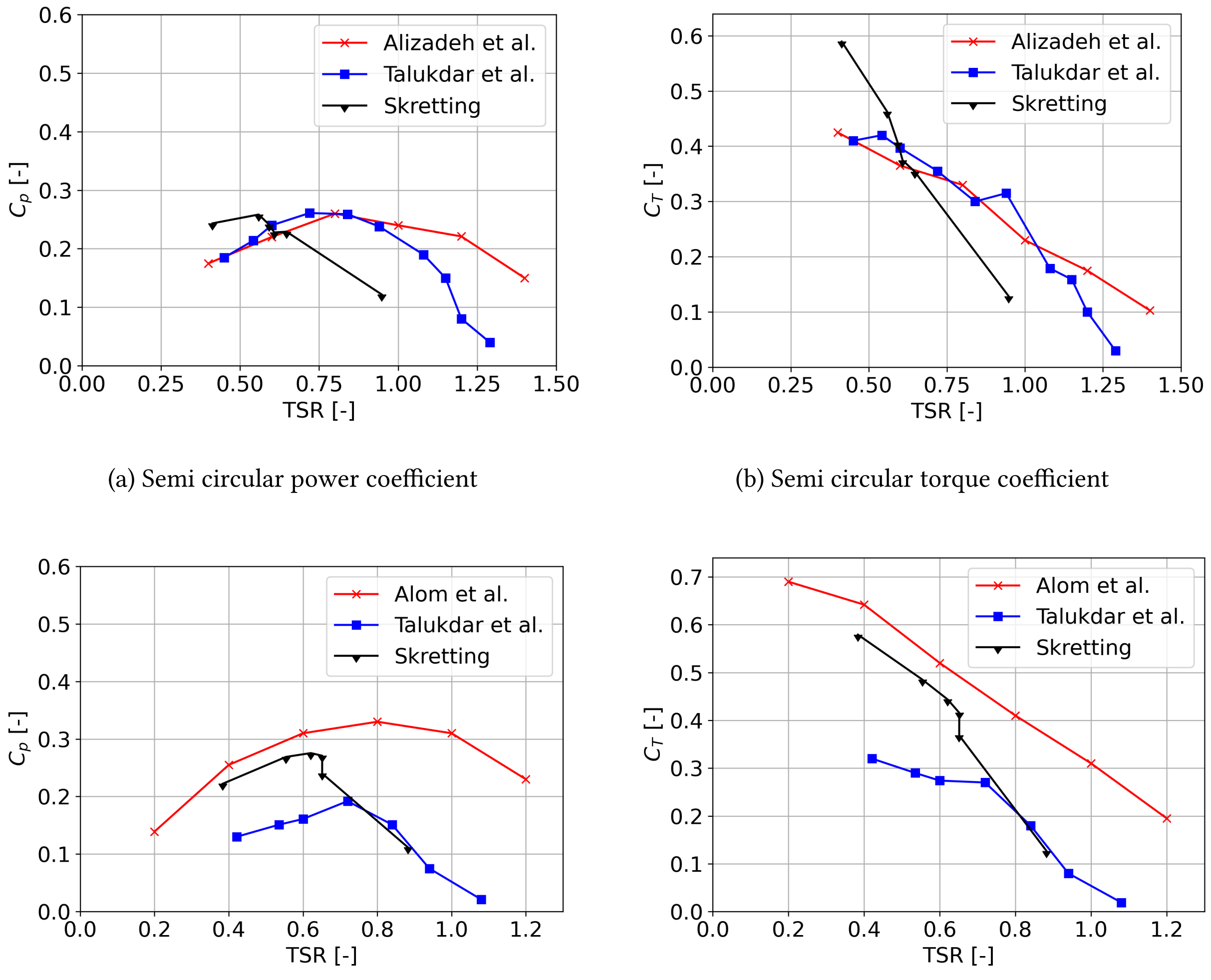


(b) Residuals for pressure, p

Figure 6.1.1: Residuals for the static case using the unaugmented semi circular.

6.2 Validation Against Literature

A comparison between the current blade profiles, without augmentation, and similar blade profiles in literature [106], [112], [115], can be found in Figure 6.2.1. It is apparent that neither of the performance metrics in the current thesis give the same results as the literature. For the semi circular profile, it seems like it is able to accurately predict the performance around a TSR of 0.60, however it overestimates its performance at lower TSR, and underestimates it at higher TSR. The elliptic turbine seems to resemble the results found by Talukdar et al. for TSR of 0.88, however, between 0.38 and 0.65 it follows the same trend as in Alom et al., albeit with slightly lower values.



(a) Semi circular power coefficient

(b) Semi circular torque coefficient

(c) Elliptic power coefficient

(d) Elliptic torque coefficient

Figure 6.2.1: Power- and torque coefficient as a function of TSR for the semi circular- and elliptic blade profiles without augmentations, compared with similar profiles in literature [106], [112], [115].

There are several possible explanations for this, one of which is that the turbine mesh is not fine enough. Although an average y^+ of roughly 300 was achieved at the turbine boundaries, there are regions that surpass this at times, such as the blade tip during flow separation, where adverse pressure gradients can be observed, as seen in Figure 6.2.2. Due to the large deviation from $y^+ \in [30,500]$, at most 1230 at the tip, the flow parameters most likely were not accurate, leading to inaccurate portrayal of separation. The low pressure during flow separation will increase the pressure drag force, therefore, if this effect is not captured properly, the performance of the turbine will act accordingly. Thus, if the simulated separation is smaller than the real separation, the power- and torque coefficients would appear lower than expected, which seems to be the case for TSR above 0.6. Conversely, if the separation is overestimated, the performance metrics would be higher than the real values, similar to what is observed at TSR below 0.6. Note, as long as convergence was reached, the impact of changing the relaxation coefficients were negligible on the flow properties, thus this was not deemed as a cause for the differences between literature data and the current data. Simulation time was, however, influenced by changing the relaxation values.

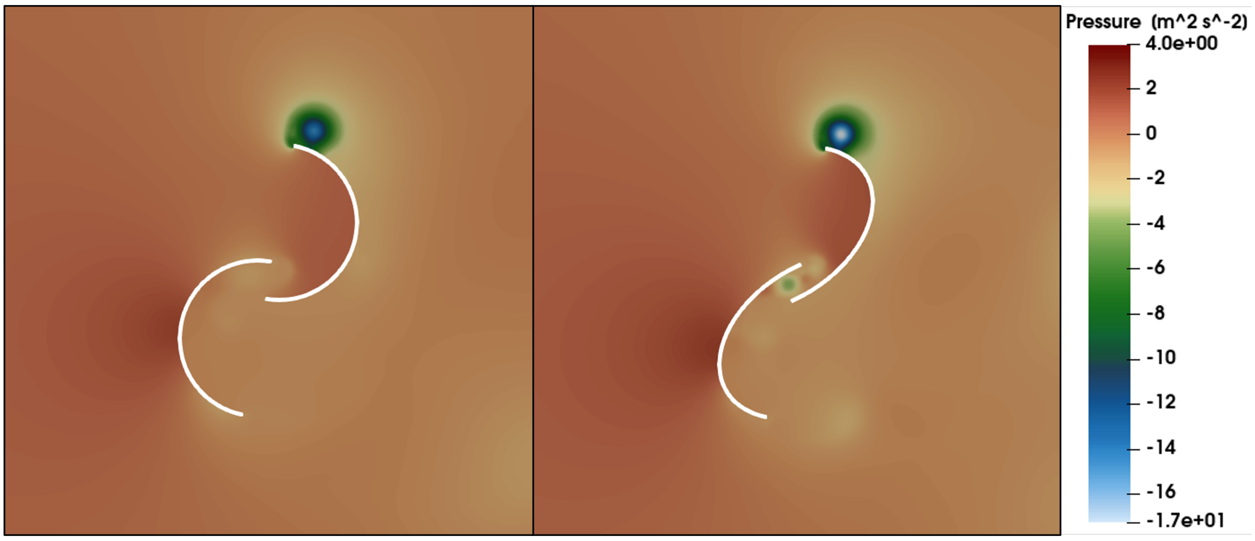
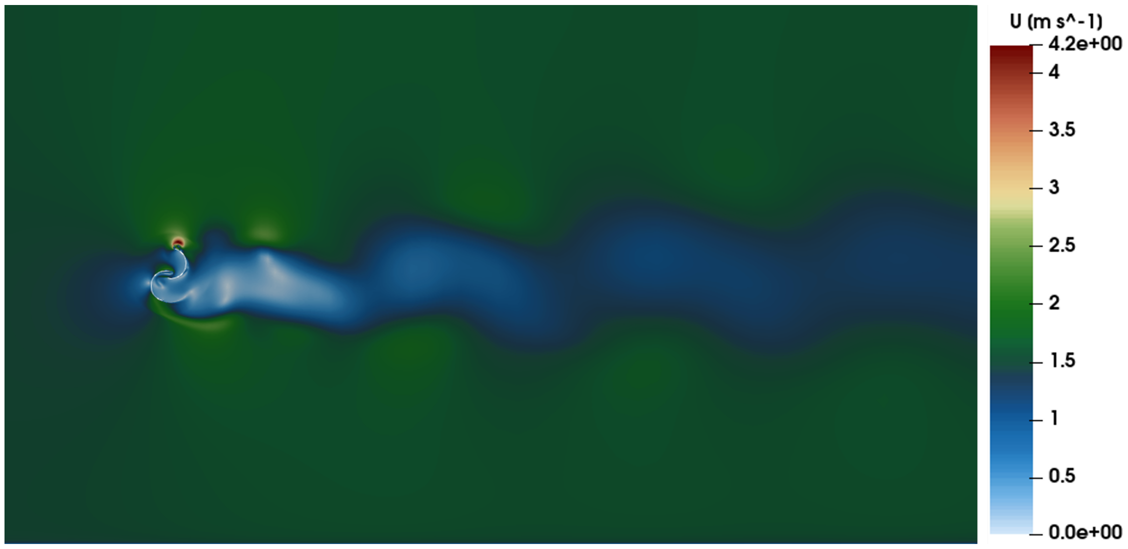
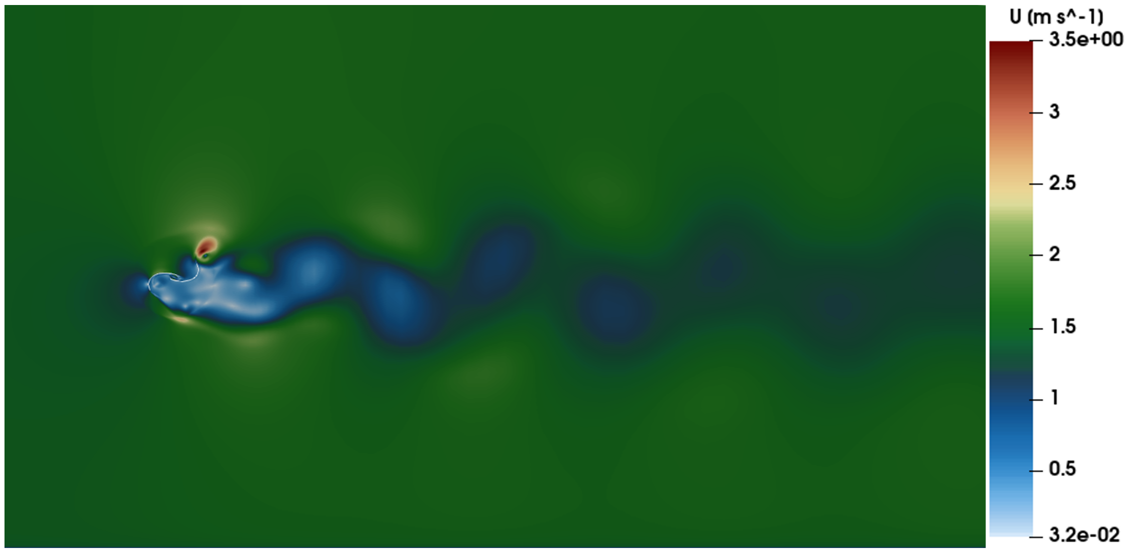


Figure 6.2.2: Pressure field around both blade profiles without augmentations at time of flow separation at the blade tip.

While the separation issue seems like the main cause of the poor correspondence to data in literature, the mesh at the top- and bottom boundaries might also affect the behaviour of the turbine, despite the distance between them. The assumption that a coarser mesh in these boundaries is sufficient might not hold, thus, a grid independence study should be performed. For the cells to achieve an average y^+ of 300, the first cell height would have to be 1.24E+02, as calculated in Chapter 5.2. However, with a domain of this size, a large amount of cells would be required to secure this refinement. To circumvent this a smaller turbine could be used, consequently decreasing the domain size. This would in turn make it more feasible, with regard to simulation time, to use a $y^+ < 1$, such that the separation can be properly captured. Disregarding the mesh, the size of the domain might not be sufficient enough with respect to the turbine, despite the theory in Chapter 2.4.1 suggesting so. Judging by the wake illustrated in Figure 6.2.3, which has a von Kármán vortex street shape, the flow has not fully developed following the obstruction by the turbine. Nevertheless, it is appreciable that the velocity seems to be stabilizing at 1.2 $\mathrm{m\,s^{-1}}$ in the x direction, and zero in the y direction, for both case 1 and case 3, which is further illustrated in Appendix J.1 and J.3, respectively. Corresponding plots for the augmented turbines are presented in Appendix J.2 for case 2 and J.4 for case 3, where they seem to stabilize at 1.3 $\mathrm{m\,s^{-1}}$ and 1.25 $\mathrm{m\,s^{-1}}$, respectively, for the x direction and zero in the y direction. For the flow to become fully developed, the velocity should reach the same value as the inlet velocity of 1.5 $\mathrm{m\,s^{-1}}$. A domain size independence test could indicate if the flow not fully developing has an impact on the turbine performance, moreover, the domain height could also be evaluated to see if any confinement effects are present. These confinement effects could be more prelevant for the augmented turbines, due to the obstruction of the augmentation geometries.



(a) Semi circular Savonius without augmentations



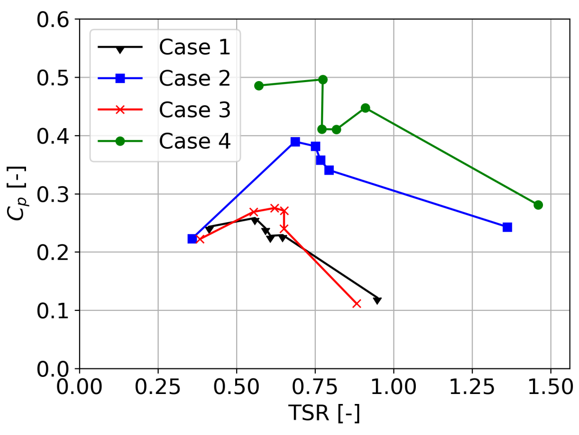
(b) Elliptic Savonius without augmentations

Figure 6.2.3: The wake velocity field of the unaugmented turbines.

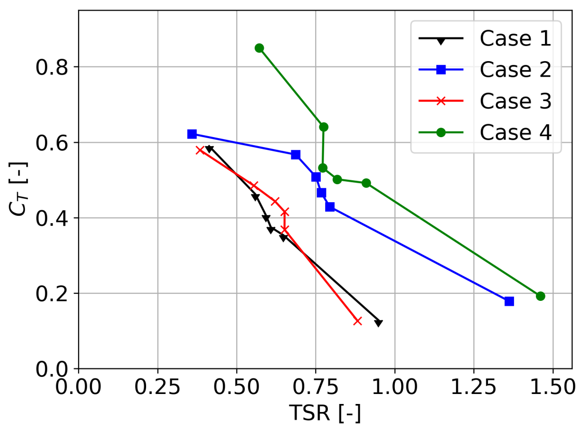
6.3 Comparing the Studied Cases

When comparing all four cases in Figure 6.3.1, it is apparent that case 4, the elliptic turbine with augmentations, has the highest performance, followed by the augmented semi circular turbine, elliptic turbine without augmentations, and the standard semi circular Savonius, respectively. Exact values for the data points corresponding to the maximum power coefficient can be found in Table 6.3.1, with tables containing data for all the points found in Appendix I.

All four turbines operate at TSR between 0.559 and 0.775, which is typical for Savonius type turbines. However, as previously discussed, the simulated power coefficients may have discrepancies with their actual values, therefore, the results can only be used as general indicators. For case 4, a $C_{p,max}$ of 0.496 seems high compared to other augmented drag turbines in literature, thus



(a) Power coefficient



(b) Torque coefficient

Figure 6.3.1: Comparison of the power- and torque coefficients as a function of TSR for all four cases.

it is plausible this value has been overestimated. Both the elliptical cases display a sudden step at a TSR of 0.651 (case 3) and 0.773 (case 4) in the power- and torque coefficient. At these TSR, it is plausible that significant overestimations of the separation and depression zones on the convex side of the advancing blade occur.

Table 6.3.1: Performance parameters of the Savonius turbines for the corresponding B_r.

Case nr.	TSR at $C_{p,max}$	$C_{p,max}$	C_T at $C_{p,max}$	B_r at $C_{p,max}$
1	0.559	0.258	0.462	1200
2	0.687	0.389	0.567	1200
3	0.621	0.276	0.444	1000
4	0.775	0.496	0.640	1200

Table 6.3.2 provides a comparison of the maximum power coefficient for all four cases, further underlining the differences between them. Adding augmentations to the semi circular profile increased its efficiency by 50.78%, whereas the elliptical profile efficiency increased by 79.71% with the addition of augmentations.

Table 6.3.2: Power coefficient comparison between all cases.

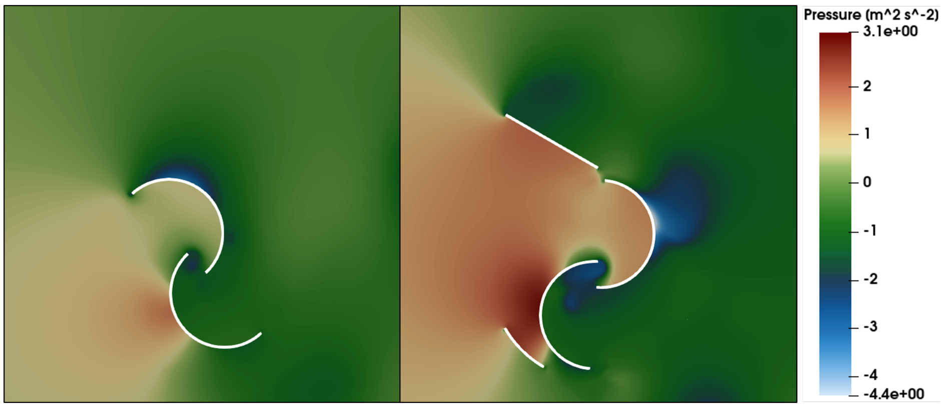
Case nr.	**1**	**2**	**3**	**4**
1	0	+50.78%	+6.98%	+92.25%
2	-33.68%	0	-29.05%	+27.51%
3	-6.52%	+40.94%	0	+79.71%
4	-47.98%	-21.57%	-44.35%	0

Similarly, the torque coefficients corresponding to the maximum power coefficients are compared in Table 6.3.3. This comparison highlights that despite providing a better power coefficient, case 3 has a 3.90% lower torque coefficient than case 1.

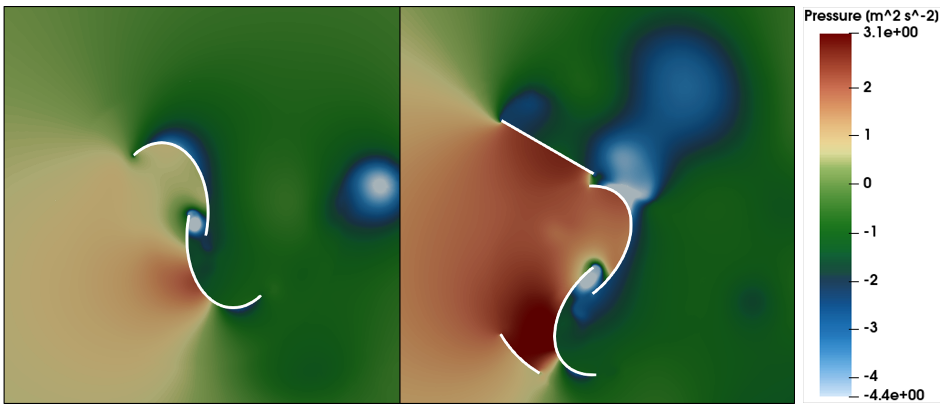
Table 6.3.3: Torque coefficient comparison between all cases.

Case nr.	**1**	**2**	**3**	**4**
1	0	+22.73%	-3.90%	+38.53%
2	-18.52%	0	-21.69%	+12.87%
3	+4.05%	+27.70%	0	+44.14%
4	-27.81%	-11.41%	-30.63%	0

Studying the pressure fields for the four cases at their maximum power point in Figure 6.3.2, it is apparent that the augmentation techniques have significant impact on the pressure on both the concave- and convex side of the blades. The upper deflector guides the flow towards the concave side of the advancing blade, increasing the pressure. Additionally, the addition of the deflector causes the pressure immediately downstream to drop, due the flow separation at this obstruction. Together, the pressure increase upstream and the pressure decrease downstream induce a greater drag force, due to the pressure difference. This pressure difference is observed to be largest for case 4, which is why it achieves a greater power coefficient. Immediately upstream of the returning blade, a region of high pressure is observed for all four cases, which acts against the preferred rotation.



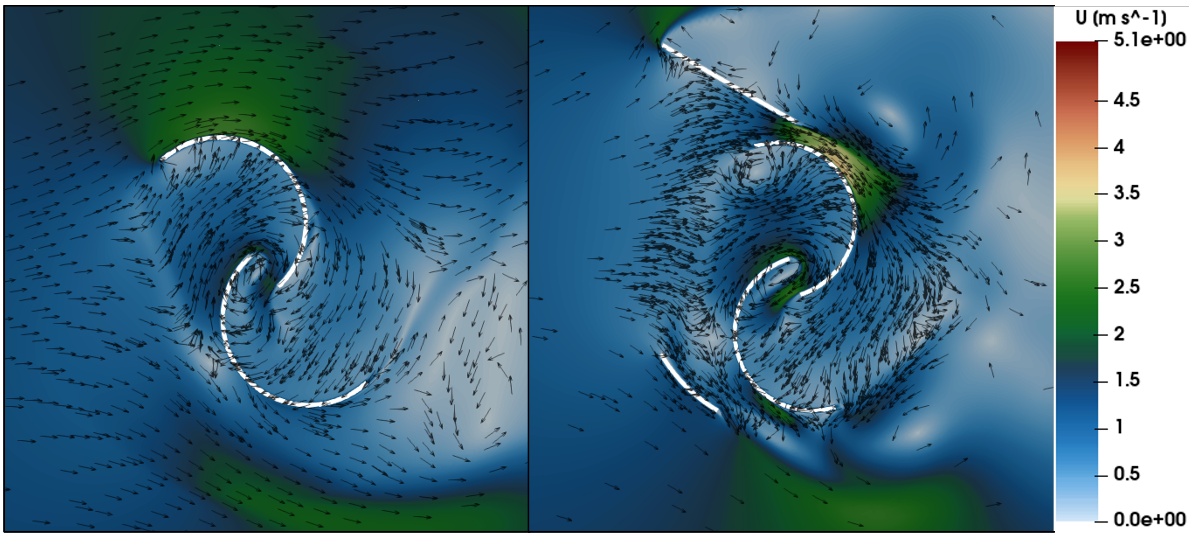
(a) Case 1 (left) and case 2 (right)



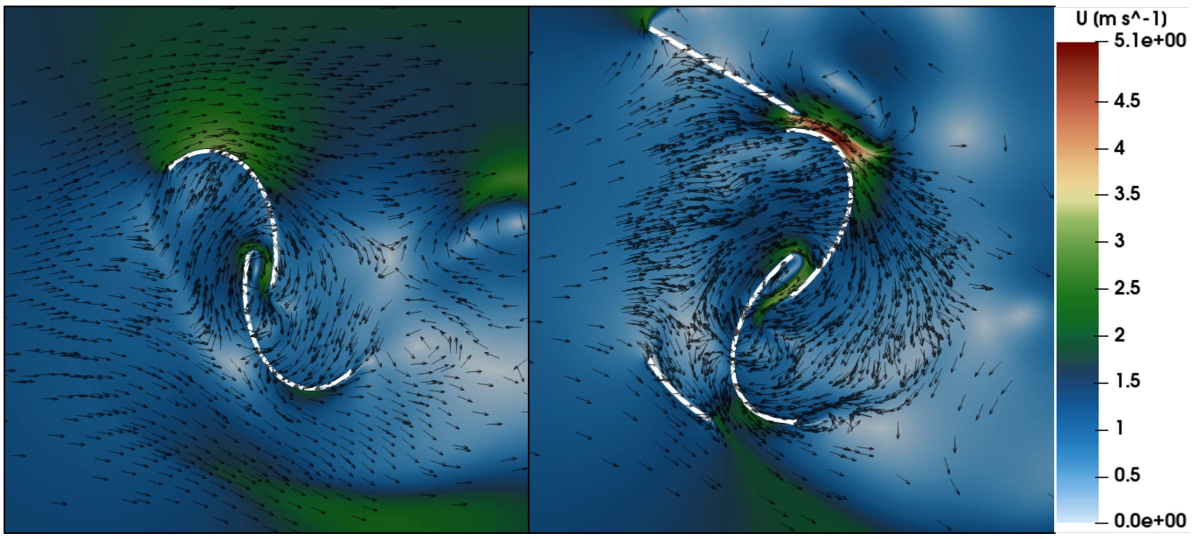
(b) Case 3 (left) and case 4 (right)

Figure 6.3.2: Pressure field around the turbines at their respective maximum power points.

The effect of the augmentation techniques is further observed using the velocity fields in Figure 6.3.3, where the barrier is seen to block a portion of the incoming flow from making contact with the returning blade. Furthermore, the effect of the overlap is also noticeable, where the flow is guided from the concave side of the advancing blade into the concave side of the returning blade. In this overlap region, there are two overlapping jets with different velocities. Between the deflector blade and the turbine, a high velocity jet is observed, which occurs due to the small cross sectional area and large pressure difference. Looking at the direction of the velocity field, this jet separates and continues towards the concave side of the returning blade.



(a) Case 1 (left) and case 2 (right)



(b) Case 3 (left) and case 4 (right)

Figure 6.3.3: Velocity field around the turbines at their respective maximum power points.

Chapter 7

Conclusions

In this thesis, two Savonius type turbines, the semi circular- and the elliptic blade profile, have been simulated both with- and without augmentation techniques for a flow induced rotation in OpenFOAM. These simulations were performed with the Reynolds Average Navier-Stokes equations, where the $k - \omega$ SST turbulence closure model was utilised, together with the PIMPLE pressure-velocity coupling algorithm. Prior to running the transient simulations, a static case simulation using SIMPLEC was performed to achieve a fully developed flow, saving an estimated time of 70 hours. Furthermore, wall functions were used to estimate the flow properties in the wall boundaries, using an average y^+ value of 300. Judging by the comparison with literature, this did not provide accurate results, as the wall functions could not estimate the flow separation properly. Nevertheless, if the obtained data are used as a general indicator of the performance of the turbines, it is apparent that the elliptical profile with augmentations has a superior performance, caused by the large pressure difference this augmentation induces. The power- and torque coefficient of this turbine was 0.496 and 0.640, respectively, at a tip speed ratio of 0.775, seeing a 79.71% and 44.14% increase, respectively, compared to its unaugmented counterpart. For the semi circular profile, the augmentation increased the power coefficient by 50.78%, from 0.258 to 0.389, whereas the torque coefficient improved by 22.73%, from 0.462 to 0.567. By evaluating the velocity fields of the wake, they seem to stabilize at constant values between 1.2 and 1.3 $\mathrm{m\,s^{-1}}$ in the x direction, and zero in the y direction, meaning they do not fully develop, as the inlet velocity was 1.5 $\mathrm{m\,s^{-1}}$.

Chapter 8

Further Work

This chapter aims to provide a guideline of how the results could be improved, as well as recommendations for future studies, based on the previous chapters.

- Decrease the turbine- and domain size
  - Fewer cells can be used, while providing similar results
  - Smoothness improved, as difference between the smallest and largest cells decrease
  - Using y^+ values below one may become feasible, with regard to simulation time.
    - Fully resolves boundary layer, increasing accuracy
    - Inflation layers should be used across the height of the boundary layer to ensure a smooth mesh.
  - Perform a domain size independence test to ensure the length and height of the domain do not impact the results significantly.
- Regardless of whether wall functions or fully resolved boundary layers are used, a grid independence test should be performed.
- Recommendations for future geometry studies:
  - Focus on the elliptical profile
  - Test with different configurations of the barrier and deflector
    - Use different arc lengths and positions for the barrier
    - Use different lengths, angles, and positions for the deflector
- Adjust the residual control criteria to smaller values
- Consider using a forced rotation, as opposed to a flow induced rotation
  - Better control on the orientation of the turbine
  - Easier to study the turbine at specific tip speed ratios
  - Easier to evaluate which time step to use
- To fully simulate the flow phenomena around the turbine, three dimensional simulations should be performed.
  - This will also allow for studies of helical blades, aspect ratio, and end plates.

Bibliography

[1] T. H. Skretting, "Innovative kinetic turbines for hydro kinetic energy conversion (hec)," M.S. thesis, University of Agder, 2021.

[2] S. J. Sangiuliano, "Turning of the tides: Assessing the international implementation of tidal current turbines," *Renewable and Sustainable Energy Reviews*, vol. 80, pp. 971–989, 2017, ISSN: 1364-0321. DOI: `https://doi.org/10.1016/j.rser.2017.05.045`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S1364032117306810`.

[3] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, "A tensorial approach to computational continuum mechanics using object-oriented techniques," *Computers in Physics*, vol. 12, no. 6, pp. 620–631, 1998. DOI: `10.1063/1.168744`. eprint: `https://aip.scitation.org/doi/pdf/10.1063/1.168744`. [Online]. Available: `https://aip.scitation.org/doi/abs/10.1063/1.168744`.

[4] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities," *International Journal for Numerical Methods in Engineering*, vol. 79, no. 11, pp. 1309–1331, 2009. DOI: `https://doi.org/10.1002/nme.2579`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2579`. [Online]. Available: `https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.2579`.

[5] P. G. Tucker, *Advanced computational fluid and aerodynamics*, eng, New York, 2016.

[6] P. A. Durbin, *Fluid dynamics with a computational perspective*, eng, New York, 2013.

[7] *Millennium problems | clay mathematics institute*, `https://www.claymath.org/millennium-problems`, (Accessed on 03/11/2021).

[8] D. De Tavernier, C. Ferreira, A. Viré, B. LeBlanc, and S. Bernardy, "Controlling dynamic stall using vortex generators on a wind turbine airfoil," *Renewable Energy*, vol. 172, pp. 1194–1211, 2021, ISSN: 0960-1481. DOI: `https://doi.org/10.1016/j.renene.2021.03.019`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0960148121003736`.

[9] S. Saedodin, M. Zaboli, and S. S. Mousavi Ajarostaghi, "Hydrothermal analysis of heat transfer and thermal performance characteristics in a parabolic trough solar collector with turbulence-inducing elements," *Sustainable Energy Technologies and Assessments*, vol. 46, p. 101 266, 2021, ISSN: 2213-1388. DOI: `https://doi.org/10.1016/j.seta.2021.101266`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S2213138821002769`.

[10] G. Cavazzini, J.-B. Houdeline, G. Pavesi, O. Teller, and G. Ardizzon, “Unstable behaviour of pump-turbines and its effects on power regulation capacity of pumped-hydro energy storage plants,” *Renewable and Sustainable Energy Reviews*, vol. 94, pp. 399–409, 2018, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2018.06.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032118304532.

[11] S. Rodriguez, “Les and dns turbulence modeling,” in *Applied Computational Fluid Dynamics and Turbulence Modeling: Practical Tools, Tips and Techniques.* Cham: Springer International Publishing, 2019, pp. 197–223, ISBN: 978-3-030-28691-0. DOI: 10.1007/978-3-030-28691-0_5. [Online]. Available: https://doi.org/10.1007/978-3-030-28691-0_5.

[12] H. K. Versteeg and W. Malalasekera, *An Introduction to Computational Fluid Dynamics*, 2nd ed. Harlow, Essex, United Kingdom: Pearson Education Limited, 2007, ISBN: 978-0-13-127498-3.

[13] N. Thuerey, K. Weißenow, L. Prantl, and X. Hu, “Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows,” *AIAA Journal*, vol. 58, no. 1, pp. 25–36, 2020. DOI: 10.2514/1.J058291. eprint: https://doi.org/10.2514/1.J058291. [Online]. Available: https://doi.org/10.2514/1.J058291.

[14] W. Liu, J. Fang, S. Rolfo, C. Moulinec, and D. R. Emerson, “An iterative machine-learning framework for rans turbulence modeling,” *International Journal of Heat and Fluid Flow*, vol. 90, p. 108 822, 2021, ISSN: 0142-727X. DOI: https://doi.org/10.1016/j.ijheatfluidflow.2021.108822. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142727X21000527.

[15] M. Xu, H. Cheng, and B. Ji, “Rans simulation of unsteady cavitation around a clark-y hydrofoil with the assistance of machine learning,” *Ocean Engineering*, vol. 231, p. 109 058, 2021, ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2021.109058. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801821004935.

[16] L. J. Voet, R. Ahlfeld, A. Gaymann, S. Laizet, and F. Montomoli, “A hybrid approach combining dns and rans simulations to quantify uncertainties in turbulence modelling,” *Applied Mathematical Modelling*, vol. 89, pp. 885–906, 2021, ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2020.07.056. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0307904X20304212.

[17] J. Weatheritt and R. D. Sandberg, “Hybrid reynolds-averaged/large-eddy simulation methodology from symbolic regression: Formulation and application,” *AIAA Journal*, vol. 55, no. 11, pp. 3734–3746, 2017. DOI: 10.2514/1.J055378. eprint: https://doi.org/10.2514/1.J055378. [Online]. Available: https://doi.org/10.2514/1.J055378.

[18] M. E. Nakhchi, S. W. Naung, and M. Rahmati, “Dns of secondary flows over oscillating low-pressure turbine using spectral/hp element method,” *International Journal of Heat and Fluid Flow*, vol. 86, p. 108 684, 2020, ISSN: 0142-727X. DOI: https://doi.org/10.1016/j.ijheatfluidflow.2020.108684. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0142727X20307074.

[19] H. Cao, X. Jia, Y. Li, C. Amador, and Y. Ding, "Cfd-dns simulation of irregular-shaped particle dissolution," *Particuology*, vol. 50, pp. 144–155, 2020, ISSN: 1674-2001. DOI: https://doi.org/10.1016/j.partic.2019.08.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1674200119301233.

[20] A. Tamburini, A. Brucato, M. Ciofalo, G. Gagliano, G. Micale, and F. Scargiali, "Cfd simulations of early- to fully-turbulent conditions in unbaffled and baffled vessels stirred by a rushton turbine," *Chemical Engineering Research and Design*, vol. 171, pp. 36–47, 2021, ISSN: 0263-8762. DOI: https://doi.org/10.1016/j.cherd.2021.04.021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263876221001830.

[21] B. Launder and D. Spalding, "The numerical computation of turbulent flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 3, no. 2, pp. 269–289, 1974, ISSN: 0045-7825. DOI: https://doi.org/10.1016/0045-7825(74)90029-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0045782574900292.

[22] D. WILCOX, "A half century historical review of the k-omega model," in *29th Aerospace Sciences Meeting*. DCW Industries Inc., 1991, p. 615. DOI: 10.2514/6.1991-615. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.1991-615. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.1991-615.

[23] F. R. Menter, M. Kuntz, and R. Langtry, "Ten years of industrial experience with the sst turbulence model," *Turbulence, heat and mass transfer*, vol. 4, no. 1, pp. 625–632, 2003. [Online]. Available: https://cfd.spbstu.ru/agarbaruk/doc/2003_Menter,%5C%20Kuntz,%5C%20Langtry_Ten%5C%20years%5C%20of%5C%20industrial%5C%20experience%5C%20with%5C%20the%5C%20SST%5C%20turbulence%5C%20model.pdf.

[24] S. ed-Din Fertahi, T. Bouhal, O. Rajad, T. Kousksou, A. Arid, T. El Rhafiki, A. Jamil, and A. Benbassou, "Cfd performance enhancement of a low cut-in speed current vertical tidal turbine through the nested hybridization of savonius and darrieus," *Energy Conversion and Management*, vol. 169, pp. 266–278, 2018, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2018.05.027. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890418305041.

[25] S. Zanforlin, "Advantages of vertical axis tidal turbines set in close proximity: A comparative cfd investigation in the english channel," *Ocean Engineering*, vol. 156, pp. 358–372, 2018, ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2018.03.035. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801818303007.

[26] H. W. Ren, F. A. Z. Mohd Saat, F. Shikh Anuar, M. A. Abdul Wahap, E. Mat Tokit, and T. B. Tuan, "Computational fluid dynamics study of wake recovery for flow across hydrokinetic turbine at different depth of water," *CFD Letters*, vol. 13, no. 2, pp. 62–76, Mar. 2021. DOI: 10.37934/cfdl.13.2.6276. [Online]. Available: https://akademiabaru.com/submit/index.php/cfdl/article/view/3362.

[27] W. Schleicher, J. Riglin, and A. Oztekin, "Numerical characterization of a preliminary portable micro hydrokinetic turbine rotor design," *Renewable Energy*, vol. 76, pp. 234–241, 2015, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2014.11.032. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148114007472.

[28] J. Riglin, C. Daskiran, J. Jonas, W. C. Schleicher, and A. Oztekin, "Hydrokinetic turbine array characteristics for river applications and spatially restricted flows," *Renewable Energy*, vol. 97, pp. 274–283, 2016, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2016.05.081. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148116304918.

[29] M. Zhao, D. Wan, and Y. Gao, "Comparative study of different turbulence models for cavitational flows around naca0012 hydrofoil," *Journal of Marine Science and Engineering*, vol. 9, no. 7, 2021, ISSN: 2077-1312. DOI: 10.3390/jmse9070742. [Online]. Available: https://www.mdpi.com/2077-1312/9/7/742.

[30] G. Saini and R. P. Saini, "Performance study of cross flow hybrid hydrokinetic turbine," in *Hydrological Extremes: River Hydraulics and Irrigation Water Management*, A. Pandey, S. Mishra, M. Kansal, R. Singh, and V. P. Singh, Eds. Cham: Springer International Publishing, 2021, pp. 249–257, ISBN: 978-3-030-59148-9. DOI: 10.1007/978-3-030-59148-9_17. [Online]. Available: https://doi.org/10.1007/978-3-030-59148-9_17.

[31] A. Kumar and R. Saini, "Performance analysis of a single stage modified savonius hydrokinetic turbine having twisted blades," *Renewable Energy*, vol. 113, pp. 461–478, 2017, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2017.06.020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148117305219.

[32] M. Mosbahi, A. Ayadi, Y. Chouaibi, Z. Driss, and T. Tucciarelli, "Performance study of a helical savonius hydrokinetic turbine with a new deflector system design," *Energy Conversion and Management*, vol. 194, pp. 55–74, 2019, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2019.04.080. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890419305242.

[33] *Openfoam: User guide: K-omega shear stress transport (sst)*, https://www.openfoam.com/documentation/guides/latest/doc/guide-turbulence-ras-k-omega-sst.html, (Accessed on 03/12/2021).

[34] L. Könözsy, "The k-ω shear-stress transport (sst) turbulence model," in *A New Hypothesis on the Anisotropic Reynolds Stress Tensor for Turbulent Flows: Volume I: Theoretical Background and Development of an Anisotropic Hybrid k-omega Shear-Stress Transport/Stochastic Turbulence Model*. Cham: Springer International Publishing, 2019, pp. 57–66, ISBN: 978-3-030-13543-0. DOI: 10.1007/978-3-030-13543-0_3. [Online]. Available: https://doi.org/10.1007/978-3-030-13543-0_3.

[35] F. R. Menter, *Improved two-equation k-omega turbulence models for aerodynamic flows - nasa technical reports server (ntrs)*, https://ntrs.nasa.gov/citations/19930013620, (Accessed on 02/11/2022), Oct. 1992.

[36] ——, “Influence of freestream values on k-omega turbulence model predictions,” *AIAA journal*, vol. 30, no. 6, pp. 1657–1659, 1992. [Online]. Available: https://cfd.spbstu.ru/agarbaruk/doc/1992_Menter_Influence%5C%20of%5C%20Freestream%5C%20Values%5C%20on%5C%20k-w%5C%20Turbulence%5C%20Model%5C%20Predictions.pdf.

[37] J. C. Kok, *Resolving the dependence on free-stream values for the k-omega turbulence model*, https://reports.nlr.nl/xmlui/handle/10921/1141, (Accessed on 02/11/2022), Jul. 1999.

[38] A. Lintermann, *Computational meshing for cfd simulations*, Oct. 2020. DOI: 10.1007/978-981-15-6716-2_6.

[39] I. Sadrehaghighi, *Unstructured Meshing for CFD*. CFD Open Series, May 2021. [Online]. Available: https://www.researchgate.net/publication/339285304_Unstructured_Meshing_for_CFD.

[40] A. de Boer, A. van Zuijlen, and H. Bijl, “Comparison of conservative and consistent approaches for the coupling of non-matching meshes,” *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 49, pp. 4284–4297, 2008, ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2008.05.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782508001916.

[41] M. Surendran, C. Lee, H. Nguyen-Xuan, G. Liu, and S. Natarajan, “Cell-based smoothed finite element method for modelling interfacial cracks with non-matching grids,” *Engineering Fracture Mechanics*, vol. 242, p. 107 476, 2021, ISSN: 0013-7944. DOI: https://doi.org/10.1016/j.engfracmech.2020.107476. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0013794420310365.

[42] X. Tunc, I. Faille, T. Gallouët, M. C. Cacas, and P. Havé, “A model for conductive faults with non-matching grids,” *Computational Geosciences*, vol. 16, no. 2, pp. 277–296, 2012. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s10596-011-9267-x.pdf.

[43] *All there is to know about different mesh types in cfd!* https://www.manchestercfd.co.uk/post/all-there-is-to-know-about-different-mesh-types-in-cfd, (Accessed on 06/12/2021).

[44] P. Pölzlbauer, A. Kümmel, D. Desvigne, and C. Breitsamter, “Numerical investigation of an optimized rotor head fairing for the racer compound helicopter in cruise flight,” *Aerospace*, vol. 8, p. 66, Mar. 2021. DOI: 10.3390/aerospace8030066.

[45] *Ijasar-s1-001fig9.jpg (945×555)*, https://scidoc.org/images/artical/IJASAR/SPL/S1/IJASAR-S1-001fig9.jpg, (Accessed on 07/12/2021).

[46] J. Kampman, “Dynamic moving mesh analysis of airfoil passing a down wind tower wake in openfoam,” 2-B Energy Holding B.V., Tech. Rep., Jul. 2019. DOI: 10.13140/RG.2.2.31851.08488.

[47] *Openfoam 2.3.0: Arbitrary mesh interface | openfoam*, https://openfoam.org/release/2-3-0/non-conforming-ami/, (Accessed on 07/12/2021).

[48] *Checkmesh - openfoamwiki*, https://openfoamwiki.net/index.php/CheckMesh, (Accessed on 05/07/2022).

[49] *Openfoam-9/meshqualitydict.cfg at master · openfoam/openfoam-9 · github*, https://github.com/OpenFOAM/OpenFOAM-9/blob/master/etc/caseDicts/mesh/generation/meshQualityDict.cfg, (Accessed on 05/07/2022).

[50] F. Aqilah, M. Islam, F. Juretic, J. Guerrero, D. Wood, and F. N. Ani, "Study of mesh quality improvement for cfd analysis of an airfoil," *IIUM Engineering Journal*, vol. 19, no. 2, pp. 203–212, 2018. DOI: https://doi.org/10.31436/iiumej.v19i2.905. [Online]. Available: https://journals.iium.edu.my/ejournal/index.php/iiumej/article/view/905.

[51] R. Gullberg, "Computational fluid dynamics in openfoam," *Report TKP*, vol. 4555, 2017.

[52] R. Lantz, "Quantitative Evaluation of Numerical Diffusion (Truncation Error)," *Society of Petroleum Engineers Journal*, vol. 11, no. 03, pp. 315–320, Sep. 1971, ISSN: 0197-7520. DOI: 10.2118/2811-PA. eprint: https://onepetro.org/spejournal/article-pdf/11/03/315/2156613/spe-2811-pa.pdf. [Online]. Available: https://doi.org/10.2118/2811-PA.

[53] J. Bredberg, "On the wall boundary condition for turbulence models," *Chalmers University of Technology, Department of Thermo and Fluid Dynamics. Internal Report 00/4. G oteborg*, pp. 8–16, 2000. [Online]. Available: http://www.tfd.chalmers.se/~lada/postscript_files/jonas_report_WF.pdf.

[54] H. Schlichting and K. Gersten, *Boundary-layer Theory*, 7th ed. McGraw-Hill, 1979, ISBN: 0-07-055334-3. [Online]. Available: %5Curl%7Bhttp://ae.sharif.edu/~viscousflow/Schlichting%5C%20-%5C%20Boundary%5C%20Layer%5C%20Theory.pdf%7D.

[55] Y. Cengel and J. Cimbala, *Fluid Mechanics: Fundamentals and Applications*, 3rd ed. McGraw-Hill, 2006, ISBN: 978-0-07-338032-2. [Online]. Available: %5Curl%7Bhttp://bayanbox.ir/view/8663792249632045937/Fluid-Cengel-3ed.pdf%7D.

[56] *Skin friction coefficient – cfd-wiki, the free cfd reference*, https://www.cfd-online.com/Wiki/Skin_friction_coefficient, (Accessed on 04/22/2022).

[57] I. A. Yeginbayeva and M. Atlar, "An experimental investigation into the surface and hydrodynamic characteristics of marine coatings with mimicked hull roughness ranges," *Biofouling*, vol. 34, no. 9, pp. 1001–1019, 2018, PMID: 30537869. DOI: 10.1080/08927014.2018.1529760. eprint: https://doi.org/10.1080/08927014.2018.1529760. [Online]. Available: https://doi.org/10.1080/08927014.2018.1529760.

[58] *Laws of flow in rough pipes - nasa technical reports server (ntrs)*, https://ntrs.nasa.gov/citations/19930093938, (Accessed on 12/12/2021).

[59] A. Segalini and P. Inghels, "Confinement effects in wind-turbine and propeller measurements," *Journal of Fluid Mechanics*, vol. 756, pp. 110–129, 2014. DOI: 10.1017/jfm.2014.440.

[60] *Openfoam: User guide: Klowrewallfunction*, https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-wall-turbulence-kLowReWallFunction.html, (Accessed on 03/31/2022).

[61] *Openfoam: User guide: Kqrwallfunction*, https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-wall-turbulence-kqRWallFunction.html, (Accessed on 04/06/2022).

[62] V. Neary, B. Gunawan, and D. Sale, "Turbulent inflow characteristics for hydrokinetic energy conversion in rivers," *Renewable and Sustainable Energy Reviews*, vol. 26, pp. 437–445, 2013, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2013.05.033. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032113003365.

[63] P. Mycek, B. Gaurier, G. Germain, G. Pinon, and E. Rivoalen, "Experimental study of the turbulence intensity effects on marine current turbines behaviour. part i: One single turbine," *Renewable Energy*, vol. 66, pp. 729–746, 2014, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2013.12.036. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S096014811400007X.

[64] *Openfoam: User guide: Omegawallfunction*, https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-wall-turbulence-omegaWallFunction.html, (Accessed on 04/22/2022).

[65] *Openfoam: User guide: Wall conditions*, https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-derived-wall.html, (Accessed on 04/22/2022).

[66] N. FUEYO and J. A. BLASCO, "Relaxation control in the solution of cfd problems," *International Journal of Computational Fluid Dynamics*, vol. 13, no. 1, pp. 43–63, 1999. DOI: 10.1080/10618569908940889. eprint: https://doi.org/10.1080/10618569908940889. [Online]. Available: https://doi.org/10.1080/10618569908940889.

[67] M. Zeng and W. Tao, "A comparison study of the convergence characteristics and robustness for four variants of simple-family at fine grids," *Engineering Computations*, 2003.

[68] *Openfoam guide/the simple algorithm in openfoam - openfoamwiki*, https://openfoamwiki.net/index.php/The_SIMPLE_algorithm_in_OpenFOAM, (Accessed on 08/12/2021).

[69] *Openfoam: User guide: Simplefoam*, (Accessed on 08/12/2021). [Online]. Available: %5Curl%7Bhttps://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-incompressible-simpleFoam.html#sec-applications-solvers-basic-simpleFoam-equations%7D.

[70] *Applications/solvers/incompressible/simplefoam · master · development / openfoam · gitlab*, https://develop.openfoam.com/Development/openfoam/-/tree/master/applications/solvers/incompressible/simpleFoam, (Accessed on 08/12/2021).

[71] *Openfoam guide/the piso algorithm in openfoam - openfoamwiki*, https://openfoamwiki.net/index.php/OpenFOAM_guide/The_PISO_algorithm_in_OpenFOAM, (Accessed on 08/12/2021).

[72] *Applications/solvers/incompressible/pisofoam · master · development / openfoam · gitlab*, https://develop.openfoam.com/Development/openfoam/-/tree/master/applications/solvers/incompressible/pisoFoam, (Accessed on 08/12/2021).

[73] *Openfoam guide/the pimple algorithm in openfoam - openfoamwiki*, https://openfoamwiki.net/index.php/OpenFOAM_guide/The_PIMPLE_algorithm_in_OpenFOAM, (Accessed on 08/12/2021).

[74] *Applications/solvers/incompressible/pimplefoam · master · development / openfoam · gitlab*, https://develop.openfoam.com/Development/openfoam/-/tree/master/applications/solvers/incompressible/pimpleFoam, (Accessed on 11/12/2021).

[75] F. Balduzzi, A. Bianchini, G. Ferrara, and L. Ferrari, "Dimensionless numbers for the assessment of mesh and timestep requirements in cfd simulations of darrieus wind turbines," *Energy*, vol. 97, pp. 246–261, 2016, ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2015.12.111. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544215017569.

[76] T. Holzmann, *Mathematics, Numerics, Derivations and OpenFOAM®*. Nov. 2019.

[77] R. Amano and B. Sundén, *Computational fluid dynamics and heat transfer: emerging topics*. WIT Press, 2011, vol. 23.

[78] I. Marinić-Kragić, D. Vučina, and Z. Milas, "Numerical workflow for 3d shape optimization and synthesis of vertical-axis wind turbines for specified operating regimes," *Renewable Energy*, vol. 115, pp. 113–127, 2018, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2017.08.030. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S096014811730784X.

[79] Z. Zhou, F. Scuiller, J. F. Charpentier, M. Benbouzid, and T. Tang, "An up-to-date review of large marine tidal current turbine technologies," in *2014 International Power Electronics and Application Conference and Exposition*, 2014, pp. 480–484. DOI: 10.1109/PEAC.2014.7037903.

[80] Z. Zhou, M. Benbouzid, J.-F. Charpentier, F. Scuiller, and T. Tang, "Developments in large marine current turbine technologies – a review," *Renewable and Sustainable Energy Reviews*, vol. 71, pp. 852–858, 2017, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2016.12.113. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032116311698.

[81] *Dnv gl issues statement of feasibility for alstom's oceade tidal turbine*, https://www.windpowerengineering.com/dnv-gl-issues-statement-of-feasibility-for-alstoms-oceade-tidal-turbine/, (Accessed on 05/08/2022).

[82] *Open hydro : Emec: European marine energy centre*, https://www.emec.org.uk/about-us/our-tidal-clients/open-hydro/, (Accessed on 05/08/2022).

[83] *Seagen turbine, northern ireland, uk*, https://www.power-technology.com/projects/strangford-lough/, (Accessed on 05/08/2022).

[84] EMEC, *Tidal developers : Emec: European marine energy centre*, http://www.emec.org.uk/marine-energy/tidal-developers/, (Accessed on 02/10/2021).

[85] G. Saini and R. P. Saini, "A review on technology, configurations, and performance of cross-flow hydrokinetic turbines," *International Journal of Energy Research*, vol. 43, no. 13, pp. 6639–6679, 2019. DOI: https://doi.org/10.1002/er.4625. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/er.4625. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/er.4625.

[86] D. Gorelov and V. Krivospitsky, "Prospects for development of wind turbines with orthogonal rotor," *Thermophysics and Aeromechanics*, vol. 15, pp. 153–157, Aug. 2008. DOI: https://doi.org/10.1134/S0869864308010149.

[87] M. Mohamed, G. Janiga, E. Pap, and D. Thévenin, "Optimal blade shape of a modified savonius turbine using an obstacle shielding the returning blade," *Energy Conversion and Management*, vol. 52, no. 1, pp. 236–242, 2011, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2010.06.070. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890410002918.

[88] E. Commission, D.-G. for Research, and Innovation, *Non-nuclear energy - Joule II : Wave energy: The exploitation of tidal and marine currents*. Publications Office, 1996.

[89] H. Chen, T. Tang, N. Aït-Ahmed, M. E. H. Benbouzid, M. Machmoum, and M. E.-H. Zaïm, "Attraction, challenge and current status of marine current energy," *IEEE Access*, vol. 6, pp. 12 665–12 685, 2018. DOI: 10.1109/ACCESS.2018.2795708.

[90] P. A. S. F. Silva, L. D. Shinomiya, T. F. de Oliveira, J. R. P. Vaz, A. L. Amarante Mesquita, and A. C. P. Brasil Junior, "Analysis of cavitation for the optimized design of hydrokinetic turbines using bem," *Applied Energy*, vol. 185, pp. 1281–1291, 2017, Clean, Efficient and Affordable Energy for a Sustainable Future, ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2016.02.098. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261916302549.

[91] P. Kumar and R. Saini, "Study of cavitation in hydro turbines—a review," *Renewable and Sustainable Energy Reviews*, vol. 14, no. 1, pp. 374–383, 2010, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2009.07.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032109001609.

[92] K. Golecha, T. Eldho, and S. Prabhu, "Influence of the deflector plate on the performance of modified savonius water turbine," *Applied Energy*, vol. 88, no. 9, pp. 3207–3217, 2011, ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2011.03.025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261911001826.

[93] B. D. Altan and M. Atılgan, "An experimental and numerical study on the improvement of the performance of savonius wind rotor," *Energy Conversion and Management*, vol. 49, no. 12, pp. 3425–3432, 2008, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2008.08.021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890408003063.

[94] J. N. Goundar, M. R. Ahmed, and Y.-H. Lee, "Design and Optimization of a Ducted Marine Current Savonius Turbine for Gun-Barrel Passage, Fiji," *Journal of Offshore Mechanics and Arctic Engineering*, vol. 141, no. 2, Oct. 2018, 021901, ISSN: 0892-7219. DOI: 10.1115/1.4041459. eprint: https://asmedigitalcollection.asme.org/offshoremechanics/article-pdf/141/2/021901/6250439/omae_141_02_021901.pdf. [Online]. Available: https://doi.org/10.1115/1.4041459.

[95] B. Yang and C. Lawn, "Fluid dynamic performance of a vertical axis turbine for tidal currents," *Renewable Energy*, vol. 36, no. 12, pp. 3355–3366, 2011, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2011.05.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148111002400.

[96] T. Harries, A. Kwan, J. Brammer, and R. Falconer, "Physical testing of performance characteristics of a novel drag-driven vertical axis tidal stream turbine; with comparisons to a conventional savonius," *International Journal of Marine Energy*, vol. 14, pp. 215–228, 2016, ISSN: 2214-1669. DOI: https://doi.org/10.1016/j.ijome.2016.01.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214166916300054.

[97] P. Jaohindy, S. McTavish, F. Garde, and A. Bastide, "An analysis of the transient forces acting on savonius rotors with different aspect ratios," *Renewable Energy*, vol. 55, pp. 286–295, 2013, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2012.12.045. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148113000074.

[98] K. Sornes, "Small-scale water current turbines for river applications," *Zero Emission Resource Organisation (ZERO)*, pp. 1–19, 2010.

[99] P. Bachant and M. Wosnik, "Performance measurements of cylindrical- and spherical-helical cross-flow marine hydrokinetic turbines, with estimates of exergy efficiency," *Renewable Energy*, vol. 74, pp. 318–325, 2015, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2014.07.049. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148114004479.

[100] S. Mohammadi, M. Hassanalian, H. Arionfard, and S. Bakhtiyarov, "Optimal design of hydrokinetic turbine for low-speed water flow in golden gate strait," *Renewable Energy*, vol. 150, pp. 147–155, 2020, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2019.12.142. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S096014811932021X.

[101] P. Dudhgaonkar, N. Duraisamy, and P. Jalihal, "Energy extraction from ocean currents using straight bladed cross-flow hydrokinetic turbine," *The International Journal of Ocean and Climate Systems*, vol. 8, no. 1, pp. 4–9, 2017. DOI: 10.1177/1759313116673081. eprint: https://doi.org/10.1177/175931311667308. [Online]. Available: https://doi.org/10.1177/17593131166730.

[102] N. Alom and U. K. Saha, "Evolution and Progress in the Development of Savonius Wind Turbine Rotor Blade Profiles and Shapes," *Journal of Solar Energy Engineering*, vol. 141, no. 3, Nov. 2018, 030801, ISSN: 0199-6231. DOI: 10.1115/1.4041848. eprint: https://asmedigitalcollection.asme.org/solarenergyengineering/article-pdf/141/3/030801/6409979/sol_141_03_030801.pdf. [Online]. Available: https://doi.org/10.1115/1.4041848.

[103] N. Sarma, A. Biswas, and R. Misra, "Experimental and computational evaluation of savonius hydrokinetic turbine for low velocity condition with comparison to savonius wind turbine at the same input power," *Energy Conversion and Management*, vol. 83, pp. 88–98, 2014, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2014.03.070. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S0196890414002714.

[104] S. Roy, "Aerodynamic performance evaluation of a novel savonius0style wind turbine through unsteady simulations and wind tunnel experiments," Ph.D. dissertation, 2014. [Online]. Available: http://gyan.iitg.ernet.in/handle/123456789/604.

[105] S. Roy and U. K. Saha, "Wind tunnel experiments of a newly developed two-bladed savonius-style wind turbine," *Applied Energy*, vol. 137, pp. 117–125, 2015, ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2014.10.022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306261914010630.

[106] *Aerodynamic Design Optimization of Elliptical-Bladed Savonius-Style Wind Turbine by Numerical Simulations*, vol. Volume 6: Ocean Space Utilization; Ocean Renewable Energy, International Conference on Offshore Mechanics and Arctic Engineering, V006T09A009, Jun. 2016. DOI: 10.1115/OMAE2016-55095. eprint: https://asmedigitalcollection.asme.org/OMAE/proceedings-pdf/OMAE2016/49972/V006T09A009/2570244/v006t09a009-omae2016-55095.pdf. [Online]. Available: https://doi.org/10.1115/OMAE2016-55095.

[107] *Unsteady Flow Analysis Around an Elliptic-Bladed Savonius-Style Wind Turbine*, vol. ASME 2014 Gas Turbine India Conference, Gas Turbine India Conference, V001T05A001, Dec. 2014. DOI: 10.1115/GTINDIA2014-8141. eprint: https://asmedigitalcollection.asme.org/GTINDIA/proceedings-pdf/GTINDIA2014/49644/V001T05A001/4240726/v001t05a001-gtindia2014-8141.pdf. [Online]. Available: https://doi.org/10.1115/GTINDIA2014-8141.

[108] K. Kacprzak and K. Sobczak, "Computational assessment of the influence of the overlap ratio on the power characteristics of a classical savonius wind turbine," *Open Engineering*, vol. 5, Jan. 2015. DOI: 10.1515/eng-2015-0039.

[109] P. Marsh, D. Ranmuthugala, I. Penesis, and G. Thomas, "The influence of turbulence model and two and three-dimensional domain selection on the simulated performance characteristics of vertical axis tidal turbines," *Renewable Energy*, vol. 105, pp. 106–116, 2017, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2016.11.063. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148116310473.

[110] K. Kacprzak, G. Liskiewicz, and K. Sobczak, "Numerical investigation of conventional and modified savonius wind turbines," *Renewable Energy*, vol. 60, pp. 578–585, 2013, ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2013.06.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960148113003029.

[111] M. H. Khanjanpour and A. A. Javadi, "Optimization of a horizontal axis tidal (hat) turbine for powering a reverse osmosis (ro) desalination system using computational fluid dynamics (cfd) and taguchi method," *Energy Conversion and Management*, vol. 231, p. 113 833, 2021, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2021.113833. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890421000108.

[112] H. Alizadeh, M. H. Jahangir, and R. Ghasempour, "Cfd-based improvement of savonius type hydrokinetic turbine using optimized barrier at the low-speed flows," *Ocean Engineering*, vol. 202, p. 107 178, 2020, ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2020.107178. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801820302377.

[113] N. R. Maldar, C. Y. Ng, L. W. Ean, E. Oguz, A. Fitriadhy, and H. S. Kang, "A comparative study on the performance of a horizontal axis ocean current turbine considering deflector and operating depths," *Sustainability*, vol. 12, no. 8, 2020, ISSN: 2071-1050. DOI: 10.3390/su12083333. [Online]. Available: https://www.mdpi.com/2071-1050/12/8/3333.

[114] M. B. Salleh, N. M. Kamaruddin, and Z. Mohamed-Kassim, "Experimental investigation on the effects of deflector angles on the power performance of a savonius turbine for hydrokinetic applications in small rivers," *Energy*, vol. 247, p. 123 432, 2022, ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2022.123432. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544222003358.

[115] P. K. Talukdar, A. Sardar, V. Kulkarni, and U. K. Saha, "Parametric analysis of model savonius hydrokinetic turbines through experimental and computational investigations," *Energy Conversion and Management*, vol. 158, pp. 36–49, 2018, ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2017.12.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196890417311627.

[116] M. A. Moreno-Armendáriz, E. Ibarra-Ontiveros, H. Calvo, and C. A. Duchanoy, "Integrated surrogate optimization of a vertical axis wind turbine," *Energies*, vol. 15, no. 1, 2022, ISSN: 1996-1073. DOI: 10.3390/en15010233. [Online]. Available: https://www.mdpi.com/1996-1073/15/1/233.

[117] T. Holzmann, *Kaplan turbine*, https://holzmann-cfd.com/community/training-cases/kaplan-turbine, (Accessed on 05/06/2022).

[118] ——, *Vertical axial wind turbine*, https://holzmann-cfd.com/community/training-cases/vertical-axial-wind-turbine, (Accessed on 05/06/2022).

[119] *Openfoam v6 user guide: 4.4 numerical schemes*, https://cfd.direct/openfoam/user-guide/v6-fvschemes/, (Accessed on 05/06/2022).

[120] *6.2 numerical schemes*, https://www.openfoam.com/documentation/user-guide/6-solving/6.2-numerical-schemes, (Accessed on 05/06/2022).

Appendix A

Power Density in Water and Wind.

Fig. A.0.1 was made in Python using the following script.

```
from matplotlib import pyplot as plt
import numpy as np

x=np.linspace(0,20,100)
yH=0.5*1025*x**3
yW=0.5*1.225*x**3

fig=plt.figure(figsize=(12,6), dpi=300)
plt.plot(x, yH, color='blue', linewidth=3, label='Hydro')
plt.plot(x, yW, color='green', linewidth=3, label='Wind')
plt.xlim((0,20))
plt.ylim((0,2000))
plt.ylabel('Power Density [W/m$^2$]', size=16)
plt.xlabel('Wind Speed [m/s]', size=16)
plt.xticks(size=12)
plt.yticks(size=12)
plt.legend(loc='best')
```

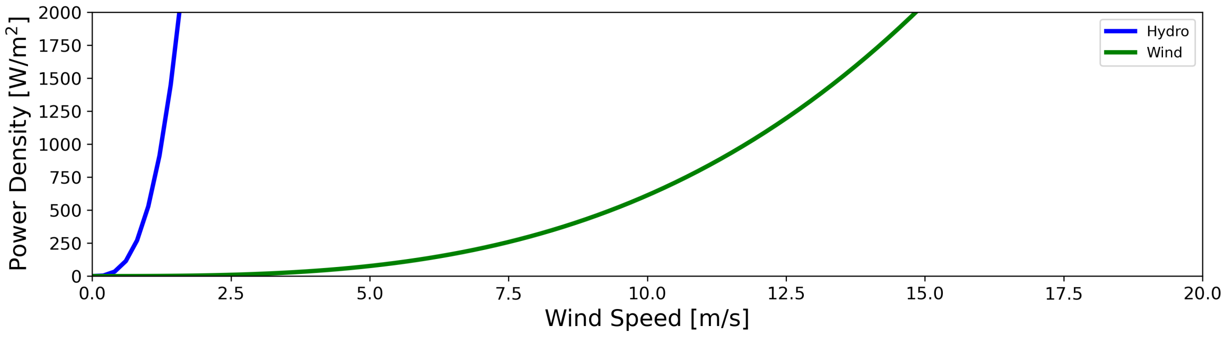


Figure A.0.1: Power density comparison between water and wind.

Appendix B

Blending Function

The blending function F_1 would look somewhat like Fig. B.0.1. It was made in Python for a range of x between zero and five, which replaced the argument inside tanh in Eq. 2.16, see below the figure. Note, that this graph would look the same for F_2, although due to the argument inside tanh being different, F_1 and F_2 would not be equal in value at all times. If, however, the argument for F_1 and the argument for F_2 were the same, the values for the blending functions would also be the same.

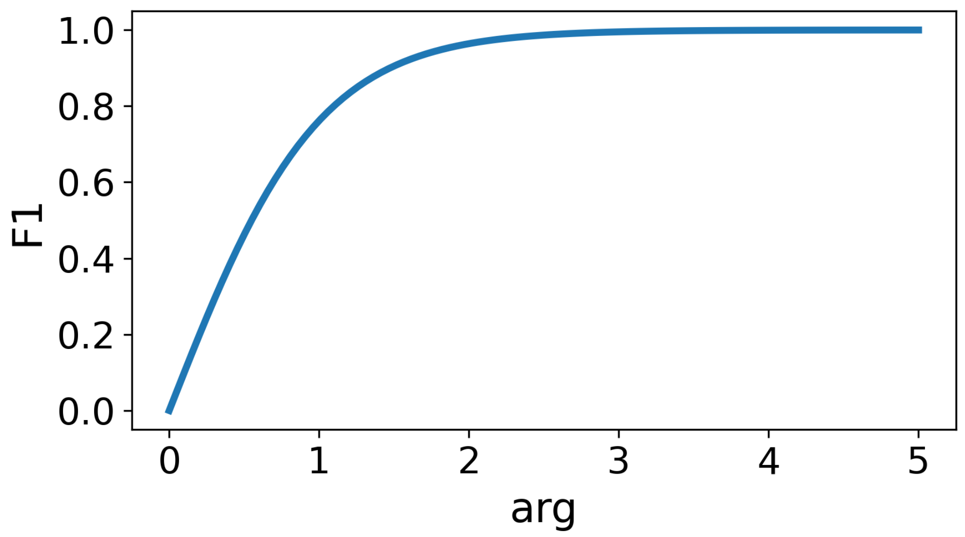
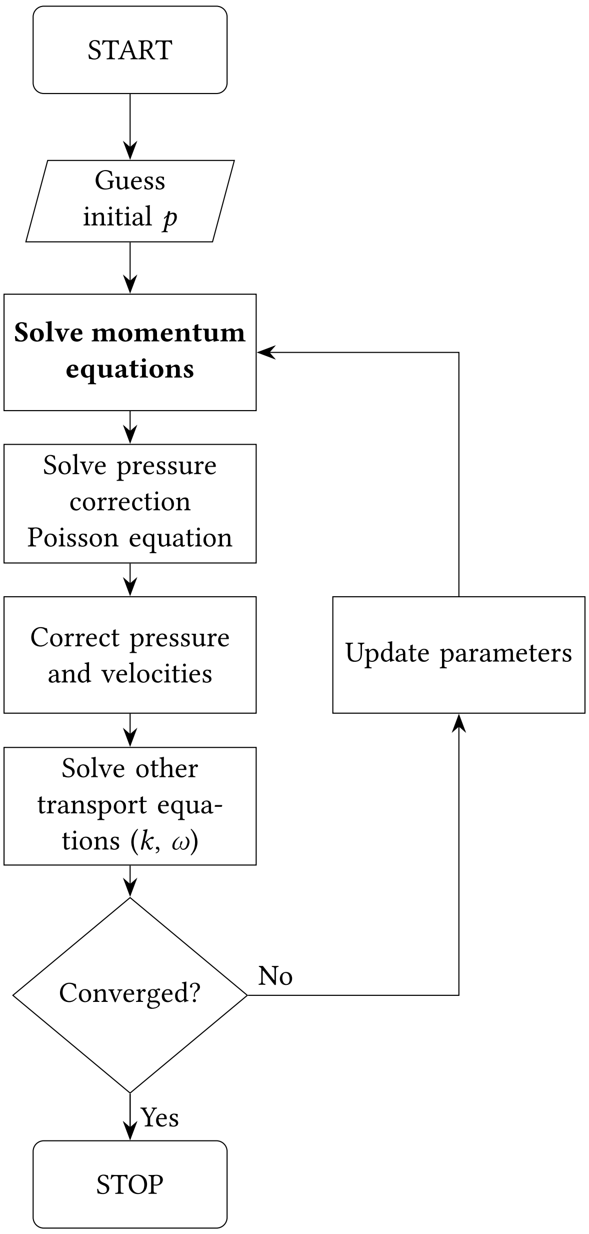


Figure B.0.1: Blending function F_1.

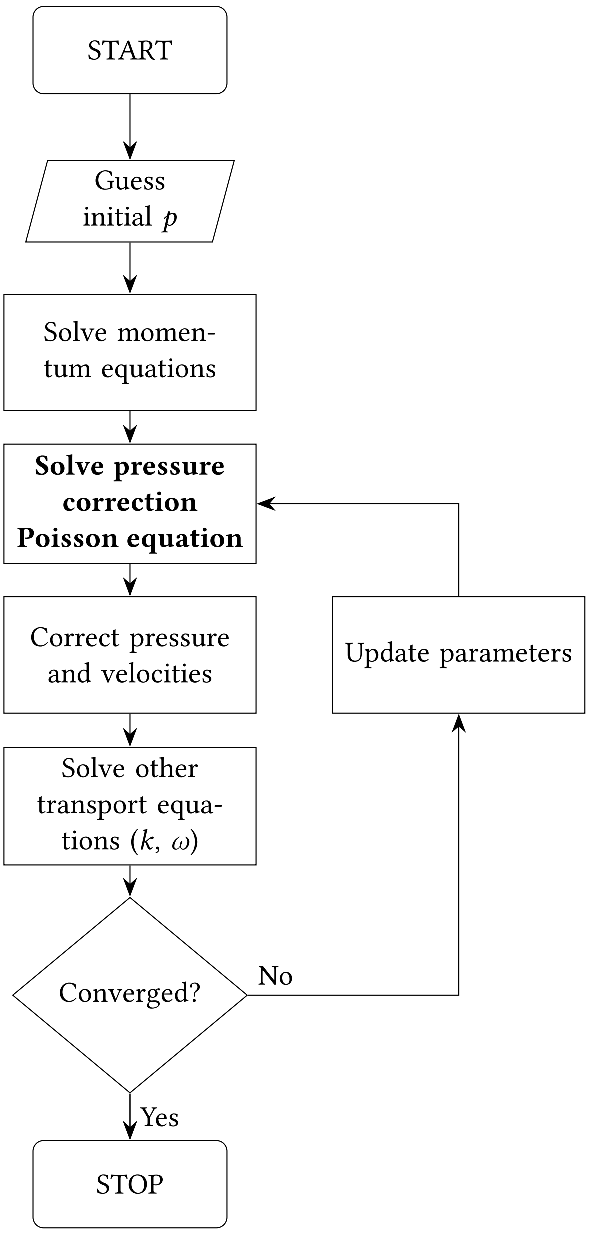
```
from matplotlib import pyplot as plt
import numpy as np

x=np.linspace(0,5,100)
y=np.tanh(x)

fig = plt.figure(figsize=(6,6), dpi=300)
plt.plot(x,y, linewidth=3)
plt.xlabel('arg', size=18)
plt.ylabel('F1', size=18)
plt.xticks(size=16)
plt.yticks(size=16)
```

Appendix C

SIMPLE and PISO flowcharts


START
Guess initial p
Solve momentum equations
Solve pressure correction Poisson equation
Correct pressure and velocities
Update parameters
Solve other transport equations (k, ω)
Converged?
No
Yes
STOP

(a) SIMPLE.


START
Guess initial p
Solve momentum equations
Solve pressure correction Poisson equation
Correct pressure and velocities
Update parameters
Solve other transport equations (k, ω)
Converged?
No
Yes
STOP

(b) PISO.

Figure C.0.1: Simplified flowcharts of the SIMPLE- and PISO algorithm, based on [5], [6], [12] and the OpenFOAM guides and source codes for SIMPLE [68]–[70] and PISO [71], [72].

Appendix D

Gmsh .geo Code for Semi Circular Savonius with Augmentations

```
SetFactory("OpenCASCADE");

Merge "InnerDomain.STEP";
Merge "OuterDomain.STEP";

// Long
Transfinite Curve {40, 42, 57, 63, 34, 36, 44, 59} = 60 Using Bump 0.2;
// Medium
Transfinite Curve {58, 60, 62, 64, 33, 35, 39, 41} = 40 Using Bump 0.2;
// Short
Transfinite Curve {47, 56, 46, 61, 43, 45, 37, 38} = 51 Using Progression 1;
// AMI + circles
Transfinite Curve {11, 12, 31, 32, 2, 4, 28, 30, 6, 7, 25, 27, 8, 9, 21, 23}
                    = 221 Using Progression 1;
// Line between AMI and circles
Transfinite Curve {10, 3, 1, 5} = 6 Using Progression 0.8;
Transfinite Curve {24, 22, 26, 29} = 6 Using Progression 1/0.8;
// Blade
Transfinite Curve {17, 19, 13, 15} = 201 Using Progression 1;
// Blade tip
Transfinite Curve {20, 14, 18, 16} = 5 Using Progression 1;
// Shield tip
Transfinite Curve {49, 51} = 5 Using Progression 1;
// Shield
Transfinite Curve {48, 50} = 81 Using Progression 1;
// Deflector
Transfinite Curve {53, 55} = 121 Using Progression 1;
// Deflector tip
Transfinite Curve {52, 54} = 5 Using Progression 1;

// Surfaces
Transfinite Surface {18} = {48, 47, 44, 35};
Transfinite Surface {17} = {47, 46, 35, 34};
Transfinite Surface {16} = {46, 45, 34, 33};
Transfinite Surface {15} = {44, 35, 31, 30};
Transfinite Surface {13} = {34, 33, 26, 28};
Transfinite Surface {12} = {31, 30, 32, 29};
Transfinite Surface {11} = {30, 26, 29, 25};
Transfinite Surface {10} = {26, 28, 25, 27};
Transfinite Surface {9} = {24, 20, 23, 18};
Transfinite Surface {6} = {20, 19, 18, 17};
Transfinite Surface {7} = {19, 22, 17, 21};
Transfinite Surface {8} = {22, 24, 21, 23};
Transfinite Surface {4} = {4, 8, 3, 7};
Transfinite Surface {3} = {8, 6, 7, 5};
Transfinite Surface {2} = {6, 2, 5, 1};
Transfinite Surface {1} = {2, 4, 1, 3};

Extrude {0, 0, 2000} {Surface{18}; Surface{17}; Surface{16};
                        Surface{15}; Surface{13}; Surface{12};
                        Surface{11}; Surface{10}; Surface{14};
                        Surface{5}; Surface{9}; Surface{4};
                        Surface{6}; Surface{3}; Surface{7};
                        Surface{2}; Surface{1}; Surface{8};
                        Layers {1}; Recombine;
}
// Physical groups
Physical Surface("inlet", 177) = {21, 32, 40};
Physical Surface("outlet", 178) = {29, 36, 49};
Physical Surface("top", 179) = {30, 26, 20};
Physical Surface("bottom", 180) = {41, 44, 48};
Physical Surface("frontandback", 181) = {9, 80, 23, 18, 35, 15, 43,
                                        12, 50, 10, 39, 13, 31, 16,
                                        17, 27, 47, 11, 63, 14, 3,
                                        90, 6, 87, 2, 96, 7, 93, 98,
                                        1, 100, 8, 5, 76, 4, 84};
Physical Surface("turbine", 182) = {68, 70, 69, 71, 73, 72, 74, 75};
Physical Surface("shield", 183) = {53, 52, 51, 54};
Physical Surface("deflector", 184) = {56, 58, 57, 55};
Physical Surface("ami1", 185) = {83, 89, 95, 97};
Physical Volume("internalVolume", 186) = {12, 14, 10, 17, 16};
Physical Surface("ami2", 187) = {78, 85, 91, 99};
Physical Volume("externalVolume", 188) = {1, 2, 3, 5, 8, 6, 7, 4, 11, 13, 15, 18, 9};
```

Appendix E

OpenFOAM Set-up, Boundary- and Initial Conditions

E.1 dynamicMeshDict

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      dynamicMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //


dynamicFvMesh      dynamicMotionSolverFvMesh;

motionSolverLibs ("libsixDoFRigidBodyMotion.so");

motionSolver            sixDoFRigidBodyMotion;

sixDoFRigidBodyMotionCoeffs
{
    //- List of mesh patches associated with the solid body
    patches         (turbine);

    //- Inner morphing distance (limit of solid-body region)
    innerDistance   1.0; //1.15;

    //- Outer morphing distance (limit of linear interpolation region)
    outerDistance   10; //1.49;

    rho             rhoInf;

    //- For incompressible cases [kg/m^3]
    rhoInf          1025;

    mass        200.559; //112906;
    momentOfInertia (81.05 99.17 151.82);

    //- Intial center of mass (default) and centre of rotation of the rigid-body
    centreOfMass (0 0 1);
```

```
//- Initial orientation of the rigid-body (default) and rotational orient.
orientation
(
    1 0 0
    0 1 0
    0 0 1
);

//- Linear velocity of the rigid-body
velocity        (0 0 0);

//- Total linear acceleration of the rigid-body
acceleration    (0 0 0);

//- Angular momentum of the rigid-body in local reference frame
angularMomentum (0 0 0);

//- Total torque on rigid-body in local reference frame
torque          (0 0 0);

//- Report motion data
report          on;

//- Acceleration relaxation coeff [0-1]
accelerationRelaxation 0.05;

//- Acceleration dumping coeff (for steady-state simulations) [0-1]
//accelerationDumping 0.1;

solver
{
    type Newmark;
}


//- Section for constraints
//  Checkout the openfoamwiki or the source code
constraints
{
    zAxis
    {
        //- Fix the axis
        sixDoFRigidBodyMotionConstraint axis;
        axis (0 0 -1);
    }

    fixedPt
    {
        //- Fix the point
        //  Here: motion also avoided in z direction
        //  Only possibility now -> rotate around z axis
        sixDoFRigidBodyMotionConstraint point;
        centreOfRotation (0 0 1);
    }
}


//- Section for restraints
//  Checkout the openfoamwiki or the source code
restraints
{
    //- Some dumping functions
```

```
        translationDamper
        {
                //- Nms/rad
                //  Acts against motion as friction
                sixDoFRigidBodyMotionRestraint  sphericalAngularDamper;
                coeff 1000;
        }

    }
}
```

E.2 turbulenceProperties

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      transportProperties;
}

// ************************************************************************* //

transportModel  Newtonian;

nu              [0 2 -1 0 0 0 0] 1e-06;
```

E.3 transportProperties

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType RAS;

RAS
{
    RASModel            kOmegaSST;
    turbulence          on;
    printCoeffs         on;
}
```

E.4 Boundary

```
FoamFile
{
    format      ascii;
    class       polyBoundaryMesh;
    location    "constant/polyMesh";
    object      boundary;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

10
(
    frontandback
    {
        type            empty;
        nFaces          201836;
        startFace       200037;
    }
    top
    {
        type            wall;
        nFaces          168;
        startFace       401873;
    }
    inlet
    {
        type            patch;
        nFaces          128;
        startFace       402041;
    }
    outlet
    {
        type            patch;
        nFaces          128;
        startFace       402169;
    }
    bottom
    {
        type            wall;
        nFaces          168;
        startFace       402297;
    }
    shield
    {
        type            wall;
        nFaces          168;
        startFace       402465;
    }
    deflector
    {
        type            wall;
        nFaces          248;
        startFace       402633;
    }
    turbine
    {
        type            wall;
        nFaces          816;
        startFace       402881;
    }
    ami2
    {
        type            cyclicAMI;
        nFaces          880;
        startFace       403697;
neighbourPatch ami1;
matchTolerance 0.0001;
transformType none;
method faceAreaWeightAMI;
    }
    ami1
    {
        type            cyclicAMI;
        nFaces          880;
        startFace       404577;
neighbourPatch ami2;
matchTolerance 0.0001;
transformType none;
method faceAreaWeightAMI;
    }
)
```

E.5 controlDict

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

application     pimpleFoam;

startFrom       latestTime;

startTime       0;

stopAt          endTime;

endTime         50;

deltaT          1e-4;

writeControl    adjustableRunTime;

writeInterval   0.1;

purgeWrite      0;

writeFormat     ascii;

writePrecision   10;

writeCompression off;

timeFormat      general;

timePrecision   6;

runTimeModifiable true;

adjustTimeStep  yes;

maxCo           5.0;

functions
{
    #include "functions/residuals"
    #include "functions/yPlus"
}
```

E.6 decomposeParDict

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      decomposeParDict;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 32;

method          hierarchical;
//method        scotch;

hierarchicalCoeffs
{
    n               (8 4 1);
    delta           0.001;
    order           xyz;
}
```

E.7 fvSchemes

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default Euler;
}

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

divSchemes
{
    default         none;
    div(phi,U)      Gauss linearUpwind grad(U);
    div(phi,k)      Gauss limitedLinear 1;
    div(phi,omega)  Gauss limitedLinear 1;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default         Gauss linear limited corrected 0.5;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         corrected;
}

wallDist
{
    method meshWave;
}
```

E.8 fvSolution

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    p
    {
        solver          GAMG;
        smoother        GaussSeidel;
        tolerance       1e-10;
        relTol          0.01;
    }

    pFinal
    {
        $p;
        tolerance       1e-10;
        relTol          0;
    }

    "pcorr.*"
    {
        solver          GAMG;
        smoother        GaussSeidel;
        tolerance       1e-10;
        relTol          0;
    }

    "(U|k|omega)"
    {
        solver         smoothSolver;
        smoother       GaussSeidel;
        tolerance      1e-08;
        relTol         0.1;
    }

    "(U|k|omega)Final"
    {
        solver         smoothSolver;
        smoother       GaussSeidel;
        tolerance      1e-08;
        relTol         0;
    }
}

PIMPLE
{
    correctPhi         yes;
    nOuterCorrectors   5;
    nCorrectors        3;
consistent          true;
}

relaxationFactors
```

```
{
    fields
    {
        p                0.3;
        pFinal             0.3;
    }
    equations
    {
        "(U|k|omega)"   0.4;
        "(U|k|omega)Final" 0.4;
    }
}

cache
{
    grad(U);
}
```

E.9 residuals

```
Description
    For specified fields, writes out the initial residuals for the first
    solution of each time step; for non-scalar fields (e.g. vectors), writes
    the largest of the residuals for each component (e.g. x, y, z).

\*---------------------------------------------------------------------------*/

residuals
{
     type            residuals;
     libs            ("libutilityFunctionObjects.so");

     writeControl    adjustableRunTime;
     writeInterval   0.1;

     fields (p U e k omega nuTilda);
}
```

E.10 yPlus

```
Description
    Calculates the turbulence y+, outputting the data as a yPlus field.

\*---------------------------------------------------------------------------*/

yPlus
{
    libs            ("libfieldFunctionObjects.so");
    type            yPlus;
    writeControl    adjustableRunTime;
    writeInterval   0.1;
}
```

E.11 Velocity

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    location    "0";
    object      U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (1.5 0 0);

boundaryField
{
    turbine
    {
        type            movingWallVelocity;
        value           uniform (0 0 0);
    }
    inlet
    {
        type            fixedValue;
        value           uniform (1.5 0 0);
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform (0 0 0);
        value           $internalField;
    }
    top
    {
        type            slip;
    }
bottom
    {
        type            noSlip;
    }
    frontandback
    {
        type            empty;
    }

"shield|deflector"
{
type noSlip;
}

ami1
    {
        type            cyclicAMI;
        value           uniform (0 0 0);
    }
    ami2
    {
        type            cyclicAMI;
        value           uniform (0 0 0);
        }
}
```

E.12 Pressure

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    turbine
    {
        type            zeroGradient;
    }
    inlet
    {
        type            zeroGradient;
value           uniform 0;
    }
    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    top
    {
        type            zeroGradient;
    }
bottom
    {
        type            fixedFluxPressure;
    }
    frontandback
    {
        type            empty;
    }

"shield|deflector"
{
        type            fixedFluxPressure;
}

ami1
    {
        type            cyclicAMI;
        value           uniform 0;
    }
    ami2
    {
        type            cyclicAMI;
        value           uniform 0;
    }
}
```

E.13 Turbulent Eddy Viscosity

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

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

internalField   uniform 0;

boundaryField
{
    turbine
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
    inlet
    {
        type            calculated;
        value           uniform 0;
    }
    outlet
    {
        type            calculated;
        value           uniform 0;
    }
    bottom
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
top
    {
        type            nutkWallFunction;
        value           uniform 0;
    }
    frontandback
    {
        type            empty;
    }

"shield|deflector"
{
        type            nutkWallFunction;
        value           uniform 0;
}

ami1
    {
        type            cyclicAMI;
        value           uniform 0;
    }
    ami2
    {
        type            cyclicAMI;
        value           uniform 0;
    }
}
```

E.14 Turbulent Kinetic Energy

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

kInlet          0.03375;

internalField   uniform $kInlet;

boundaryField
{
    turbine
    {
        type            kqRWallFunction;
        value           uniform $kInlet;
    }
    inlet
    {
        type            turbulentIntensityKineticEnergyInlet;
intensity       0.1;
        value           uniform $kInlet;
        //inletValue      uniform $kInlet;
        //value           uniform $kInlet;
    }
    outlet
    {
        type            zeroGradient;
        //inletValue      uniform $kInlet;
        //value           uniform $kInlet;
    }
    bottom
    {
        type            kqRWallFunction;
        value           uniform $kInlet;
    }
top
    {
        type zeroGradient;
//type            kqRWallFunction;
        //value           uniform $kInlet;
    }
    frontandback
    {
        type            empty;
    }
"shield|deflector"
{
        type            kqRWallFunction;
        value           uniform $kInlet;
}
ami1
    {
        type            cyclicAMI;
        value           uniform $kInlet;
    }
    ami2
    {
        type            cyclicAMI;
        value           uniform $kInlet;
    }
}
```

E.15 Specific Turbulent Dissipation Rate

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "0";
    object      omega;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 -1 0 0 0 0];

omegaInlet      1.0851506361396042;
omegaWallTurbine    102.25920628810017;
omegaWallTopBot 8.180736503048013;

internalField   uniform $omegaInlet;

boundaryField
{
    turbine
    {
        type            omegaWallFunction;
        value           uniform $omegaWallTurbine;
    }
    inlet
    {
        type            turbulentMixingLengthFrequencyInlet;
        //inletValue      uniform $omegaInlet;
mixingLength 3.6363636363636362*0.085;
// Inlet hydraulic diameter * 0.085 --> Dh=4*Area/Perimeter
        value           uniform $omegaInlet;
    }
    outlet
    {
        type            inletOutlet;
        inletValue      uniform $omegaInlet;
        value           uniform $omegaInlet;
    }
    bottom
    {
        type            omegaWallFunction;
        value           uniform $omegaWallTopBot;
    }
top
    {
        type            omegaWallFunction;
        value           uniform $omegaWallTopBot;
    }
    frontandback
    {
        type            empty;
    }

"shield|deflector"
{
        type            omegaWallFunction;
        value           uniform $omegaWallTurbine;
}
ami1
    {
        type            cyclicAMI;
        value           uniform 0;
    }
    ami2
    {
        type            cyclicAMI;
        value           uniform 0;
    }
}
```

E.16 pointDisplacement

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       pointVectorField;
    location    "0";
    object      pointDisplacement;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 0 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    ami1
    {
        //type            fixedValue;
        type            cyclicAMI;
        value           uniform (0 0 0);
    }

    ami2
    {
        //type            calculated;
        type            cyclicAMI;
        value           uniform (0 0 0);
    }

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

    outlet
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    "top|bottom|shield|deflector"
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    turbine
    {
        type            calculated;
        value           uniform (0 0 0);
    }

    frontandback
    {
        type            empty;
    }
}
```

Appendix F

Ubuntu Shell Script to Run Cases

```
#!/bin/sh
cd ${0%/*} || exit 1    # Run from this directory
. $WM_PROJECT_DIR/bin/tools/RunFunctions
processors=12
echo "Absolute start time:" $(date)
echo "Enter simpleFoam directory"
cd simpleFoam
if [ -e residuals.dat ] && [ -d ./3000 ]; then
echo "simpleFoam has already been run, continue to pimpleFoam"
elif [ -e log.simpleFoam ] && [ ! -d ./3000 ] && [ -d ./processor1 ]; then
echo "Restarting simpleFoam at:" $(date)
export OMPI_MCA_btl_vader_single_copy_mechanism=none
mpirun -np $processors simpleFoam -parallel >> log.simpleFoam
echo "Ending simpleFoam at:" $(date)
echo "Reconstruct the latest time"
reconstructPar -latestTime > log.reconstructPar
cp postProcessing/yPlus/0/yPlus.dat .
cp postProcessing/residuals/0/residuals.dat .
grep "ClockTime" log.simpleFoam | tail -1  > runTime.dat
else
cp -r _0.orig 0
echo "Decompose case"
decomposePar > log.decomposePar
echo "Starting simpleFoam at:" $(date)
export OMPI_MCA_btl_vader_single_copy_mechanism=none
mpirun -np $processors renumberMesh -overwrite -parallel >> log.decomposePar
mpirun -np $processors simpleFoam -parallel > log.simpleFoam
echo "Ending simpleFoam at:" $(date)
echo "Reconstruct the latest time"
reconstructPar -latestTime > log.reconstructPar
cp postProcessing/yPlus/0/yPlus.dat .
cp postProcessing/residuals/0/residuals.dat .
grep "ClockTime" log.simpleFoam | tail -1  > runTime.dat
fi

for i in Br200 Br800 Br900 Br1000 Br1200 Br2000
do
echo "Enter pimpleFoam $i at:" $(date)
cd ../$i
if [ -d ./processor1/50 ] && [ -e omega.dat ]; then
echo "$i has already been run, continue to next run"
elif [ ! -e omega.dat ] && [ -d ./processor1 ]; then
echo "Restarting $i Run at:" $(date)
export OMPI_MCA_btl_vader_single_copy_mechanism=none
mpirun -np $processors pimpleFoam -parallel >> log.pimpleFoam
echo "Ending $i Run at:" $(date)
```

```
grep "Angular velocity" log.pimpleFoam   | cut -d ":" -f 2 | tr -d "()" > omega
grep -e "^Time =" log.pimpleFoam | cut -d " " -f 3 > times
paste times omega > omega.dat
grep "ClockTime" log.pimpleFoam | tail -1  > runTime.dat
rm omega times
cp postProcessing/yPlus/0/yPlus.dat .
cp postProcessing/residuals/0/residuals.dat .
rm -r postProcessing
echo "Data extracted"
else
cp -r ../simpleFoam/constant/polyMesh constant
runApplication mapFields ../simpleFoam -sourceTime latestTime -consistent
cp _0.orig/pointDisplacement 0/pointDisplacement
echo "Initial conditions mapped to transient case"
echo "Decompose case"
decomposePar > log.decomposePar
echo "Finished decomposing"
echo "Starting $i Run at:" $(date)
export OMPI_MCA_btl_vader_single_copy_mechanism=none
mpirun -np $processors renumberMesh -overwrite -parallel >> log.decomposePar
mpirun -np $processors pimpleFoam -parallel > log.pimpleFoam
echo "Ending $i Run at:" $(date)
grep "Angular velocity" log.pimpleFoam   | cut -d ":" -f 2 | tr -d "()" > omega
grep -e "^Time =" log.pimpleFoam | cut -d " " -f 3 > times
paste times omega > omega.dat
grep "ClockTime" log.pimpleFoam | tail -1  > runTime.dat
rm omega times
cp postProcessing/yPlus/0/yPlus.dat .
cp postProcessing/residuals/0/residuals.dat .
rm -r postProcessing
echo "Data extracted"
#echo "Reconstruct case, starting at:" $(date)
#reconstructPar -time ':5, 45:' > log.reconstructPar
#echo "Finished reconstructing at:" $(date)
echo "Exit pimpleFoam $i at:" $(date)
fi
done
echo "Absolute end time:" $(date)
```

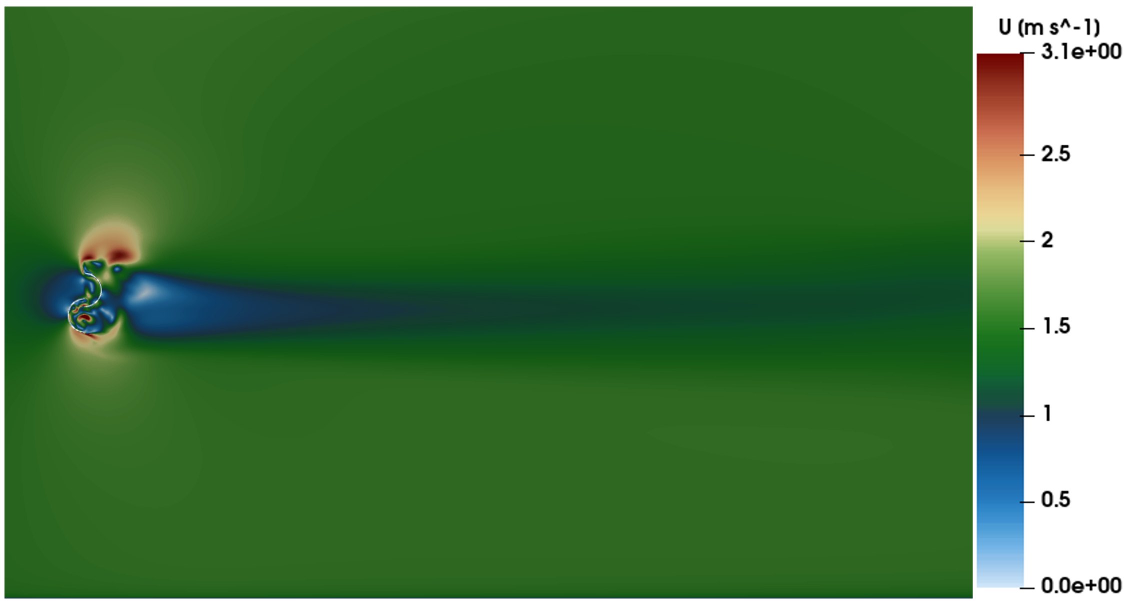
Appendix G

Python Script for Post-Processing

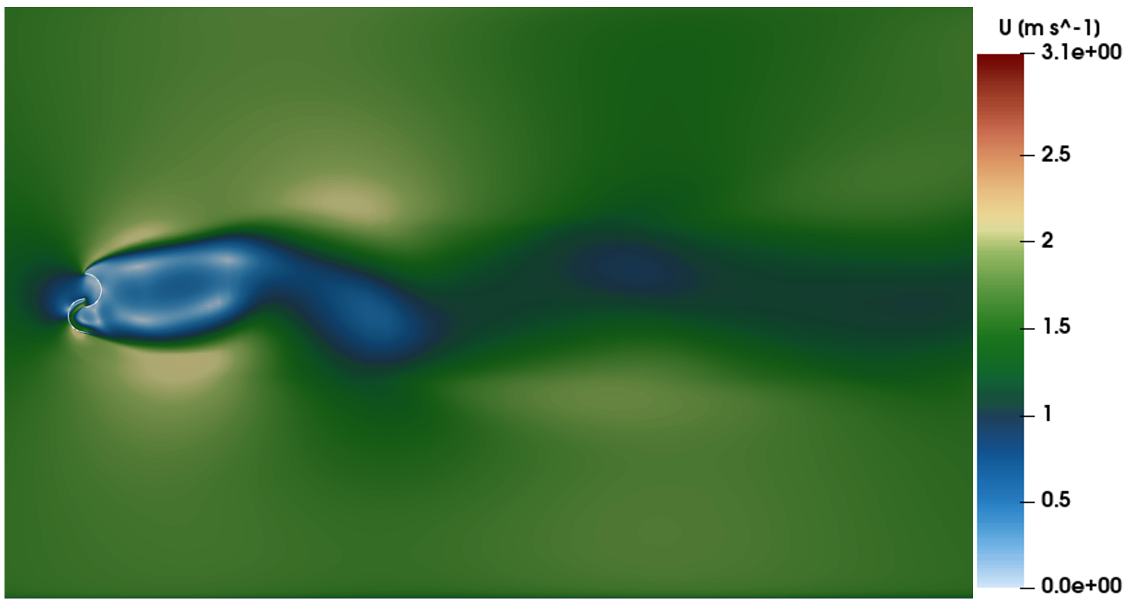
```
import matplotlib.pyplot as plt
import os
import numpy as np
import pandas as pd

list_Br= [200, 800, 900, 1000, 1200, 2000]
path=os.getcwd()


meanTime = 1105 # Depends on when quasi stability has been achieved

Cp = []
meantsr = []
Ct = []
for Br in list_Br:
    file="omega.dat"
    f = open(path+'\\Br'+str(Br)+'\\'+file,'r')
    M = []
    n = 1
    for line in f.readlines():
        M.append([ float (x) for x in line.split (' ')])
        n+= 1


    f.close();
    M = np.array(M) # convert class list to array

    time = M[:,0]
    omega = M[:,-1]*(-1)
    rpm = omega*60/(2*np.pi)
    meanrpm = np.mean(rpm[meanTime:])
    meanrpmplot = rpm*0 + meanrpm
    meanrpm = np.round(meanrpm, 2)
    ymin = min(rpm)-0.1*abs(min(rpm))
    ymax = max(rpm)+0.1*abs(max(rpm))

    plt.plot(time,rpm,label="Angular velocity")
    plt.xlabel("Time [s]")
    plt.ylabel("$\Omega_r$ [RPM]")
    plt.plot(time,meanrpmplot, label="Mean angular velocity = "+str(meanrpm)+' rpm',
                                                                color =  'darkred')
    plt.xlim(0, time[-1])
    plt.ylim(ymin, ymax)
    plt.grid()
    plt.legend()
    plt.savefig("Omega"+' '+str(Br)+'.png', dpi=300)
    plt.clf()

    R = 1      # Radius
    U = 1.5         # Inlet velocity
    tsr = omega*R/U
    meantsr_prov = np.mean(tsr[meanTime:])
    meantsrplot = tsr*0 + meantsr_prov
    meantsr.append(np.round(meantsr_prov, 4))
    ymin_tsr = min(tsr)-0.1*abs(min(tsr))
    ymax_tsr = max(tsr)+0.1*abs(max(tsr))


    plt.plot(time,tsr,label="Tip-speed ratio")
    plt.plot(time,meantsrplot, label="Mean TSR = "+str(np.round(meantsr_prov,3)),
                                                            color =  'darkred')
    plt.xlabel("Time [s]")
    plt.ylabel("TSR [-]")
    plt.xlim(0, time[-1])
    plt.ylim(ymin_tsr, ymax_tsr)
    plt.grid()
    plt.legend()
    plt.savefig("TSR"+' '+str(Br)+'.png', dpi=300)
    plt.clf()

    power = np.absolute(Br*omega**2)

    meanpower = np.mean(power[meanTime:])
    meanpowerplot = power*0 + meanpower
    meanpower = np.round(meanpower, 1)
    yminP = min(power)-0.1*abs(min(power))
    ymaxP = max(power)+0.1*abs(max(power))
    textpower_y = meanpower/2 # 21029 corresponds to time 420

    plt.plot(time[0:],power[0:],label="Instantaneous power")
    plt.plot(time[0:],meanpowerplot[0:],label="Mean power = "+str(meanpower)+' W/m',
                                                                color =  'darkred')
    plt.xlabel("time [s]")
    plt.ylabel("Power [W/m]")
    plt.ylim(yminP, ymaxP)
    plt.xlim(0, time[-1])
    plt.grid()
    plt.legend()
    plt.savefig("power"+' '+str(Br)+'.png', dpi=300)
    plt.clf()

    rho = 1025
    P_ke_L = rho*R*U**3      # Power per length or rotor, 1/2 rho height U^3
    Cp_prov = meanpower/P_ke_L
    Cp.append(meanpower/P_ke_L)   # height = 2*R
    Ct.append(Cp_prov/meantsr_prov)

    file="residuals.dat"
    f = open(path+'\\Br'+str(Br)+'\\'+file,'r')
    next(f)
    next(f)
    next(f)
    M = []
    n = 1
    for line in f.readlines():
        M.append([float(x) for x in line.split()])

        n+= 1
    M = np.array(M) # convert class list to array
    Residuals = pd.DataFrame(M, columns=['Time','p','Ux','Uy','k','omega'])

    plt.plot(Residuals['Time'],Residuals['p'],label="p")
    plt.xlabel("Time [s]")
    plt.ylabel("Residuals")
    plt.grid()
    plt.legend()
    plt.savefig("pRessidual"+' '+str(Br)+'.png', dpi=300)
    plt.clf()
    plt.plot(Residuals['Time'],Residuals['Ux'],label="Ux")
    plt.plot(Residuals['Time'],Residuals['Uy'],label="Uy")
    plt.plot(Residuals['Time'],Residuals['k'],label="k")
    plt.plot(Residuals['Time'],Residuals['omega'],label="$\omega$")
    plt.xlabel("Time [s]")
    plt.ylabel("Residuals")
    plt.grid()
    plt.legend()
    plt.savefig("otherRessiduals"+' '+str(Br)+'.png', dpi=300)
    plt.clf()

table=pd.DataFrame({'TSR': meantsr, 'Cp': Cp, 'Ct': Ct, 'Br': list_Br})

plt.plot(table['TSR'],table['Cp'])
plt.xlabel("TSR [-]")
plt.ylabel("$C_p$ [-]")
plt.ylim(0, 0.6)
plt.xlim(0, max(table['TSR'])+0.1)
plt.grid()
plt.savefig('CpVStsr.png', dpi=300)
plt.clf()

plt.plot(table['TSR'],table['Ct'])
plt.xlabel("TSR [-]")
plt.ylabel("$C_T$ [-]")
plt.ylim(0, max(table['Ct'])+0.05)
plt.xlim(0, max(table['TSR'])+0.1)
plt.grid()
plt.savefig('CtVStsr.png', dpi=300)
plt.clf()
```

Appendix H

Wake Development for the Static Case



(a) Time 500



(b) Time 3000

Figure H.0.1: The wake velocity field of the static case for the unaugmented semi circular turbine at time 500 and time 3000.

Appendix I

Performance Parameter Data

I.1 Semi Circular Savonius Turbine without Augmentations

Table I.1.1: Results for the semi circular Savonius turbine without augmentations

TSR	C_p	C_T	B_r
0.948	0.12184281842818429	0.12851975467352153	200
0.6472	0.2290587172538392	0.3539026726414282	800
0.6081	0.2278446251129178	0.37468226883030087	900
0.5922	0.2409683830171635	0.40687777876613257	1000
0.5589	0.25839927732610657	0.462335714292622	1200
0.4129	0.2433676603432701	0.5894749403980865	2000

I.2 Semi Circular with Augmentations

Table I.2.1: Results for the semi circular Savonius turbine with augmentations

TSR	C_p	C_T	B_r
1.3623	0.24307859078590785	0.17843454347229884	200
0.7944	0.34061065943992774	0.4287581311747343	800
0.7674	0.35775248419150857	0.46617248131851824	900
0.7509	0.381542908762421	0.5081259272074625	1000
0.6865	0.38946341463414635	0.5672788838694328	1200
0.3586	0.22307497741644083	0.6220897141688105	2000

I.3 Elliptical without Augmentations

Table I.3.1: Results for the elliptic Savonius turbine without augmentations

TSR	C_p	C_T	B_r
0.8824507	0.11160976	0.12647704	200
0.65143046	0.24004336	0.36848655	800
0.65161265	0.2709449	0.41580669	900
0.62092022	0.2755122	0.44371594	1000
0.55413717	0.2688925	0.48524538	1200
0.3835356	0.22217886	0.57929136	2000

I.4 Elliptical with Augmentations

Table I.4.1: Results for the elliptic Savonius turbine with augmentations

TSR	C_p	C_T	B_r
1.46044	0.280947	0.192371	200
0.909788	0.447364	0.491724	800
0.818196	0.410421	0.501617	900
0.771956	0.410855	0.532226	1000
0.774684	0.49613	0.640429	1200
0.571502	0.485695	0.849856	2000

Appendix J

Velocity Along the Wake

J.1 Semi Circular Savonius without Augmentations

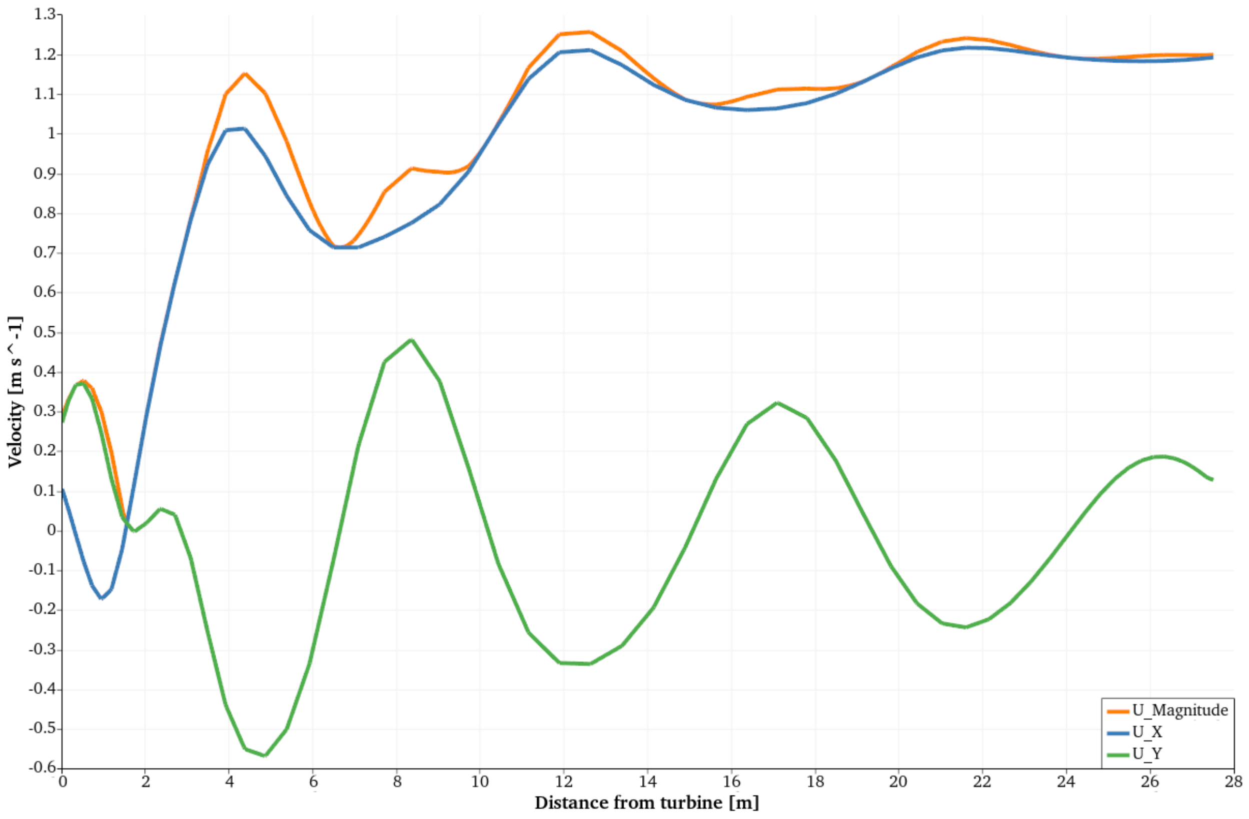


Figure J.1.1: Velocity along the wake of the semi circular Savonius without augmentations

J.2 Semi Circular Savonius with Augmentations

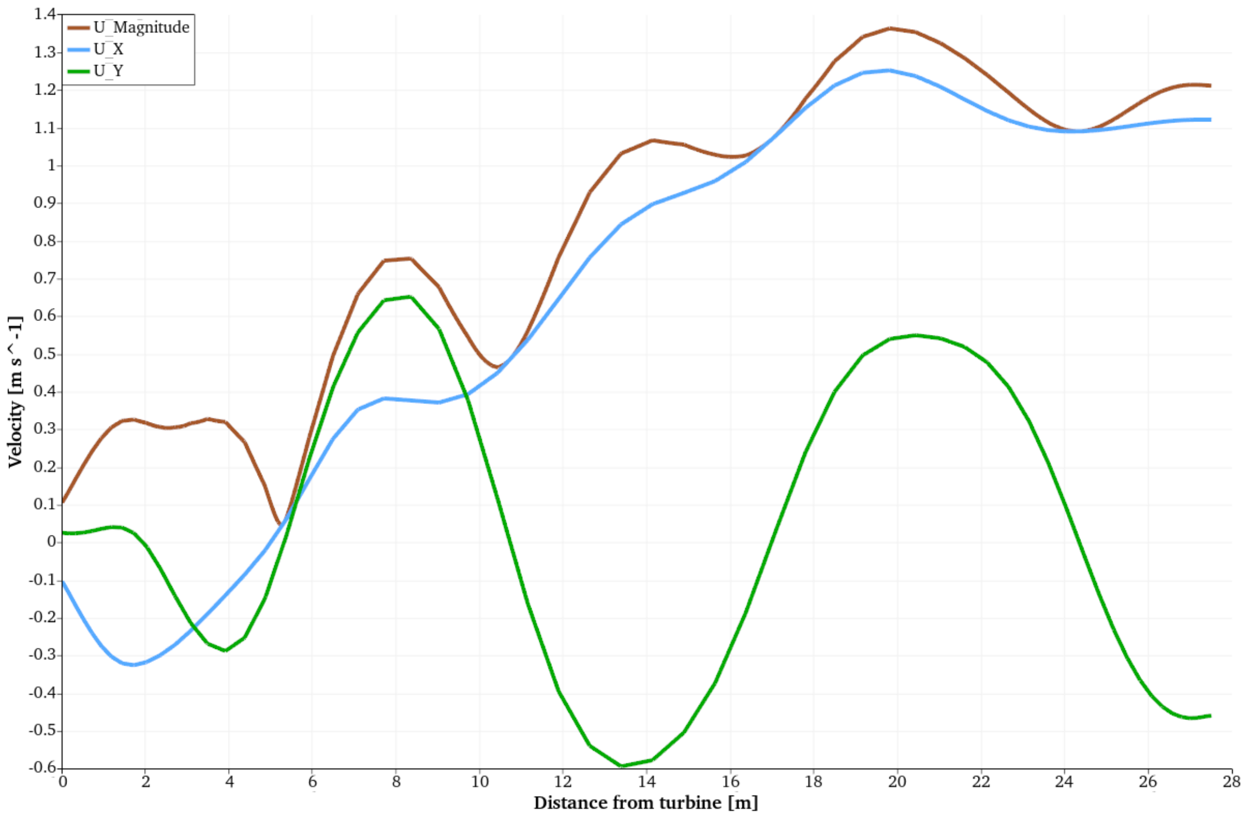


Figure J.2.1: Velocity along the wake of the semi circular Savonius with augmentations

J.3 Elliptic Savonius without Augmentations

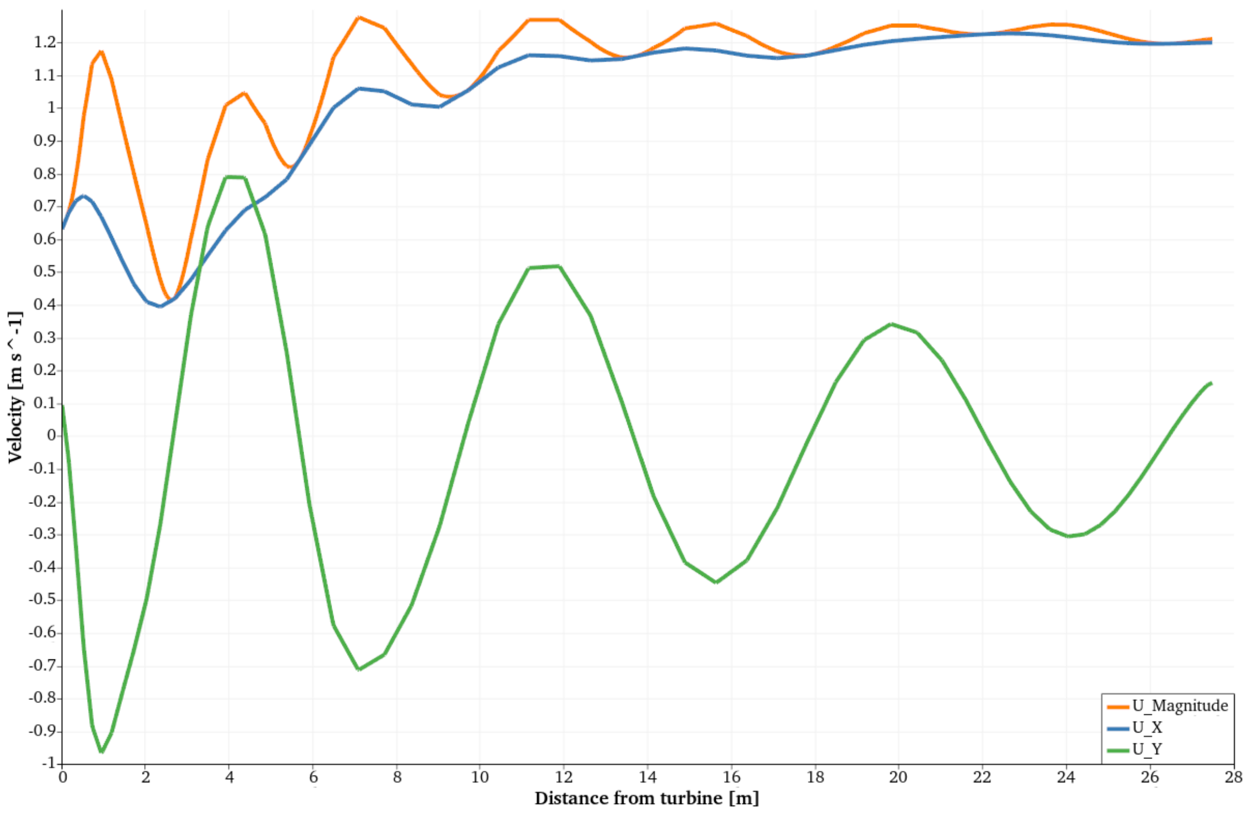


Figure J.3.1: Velocity along the wake of the elliptic Savonius without augmentations

J.4 Elliptic Savonius with Augmentations

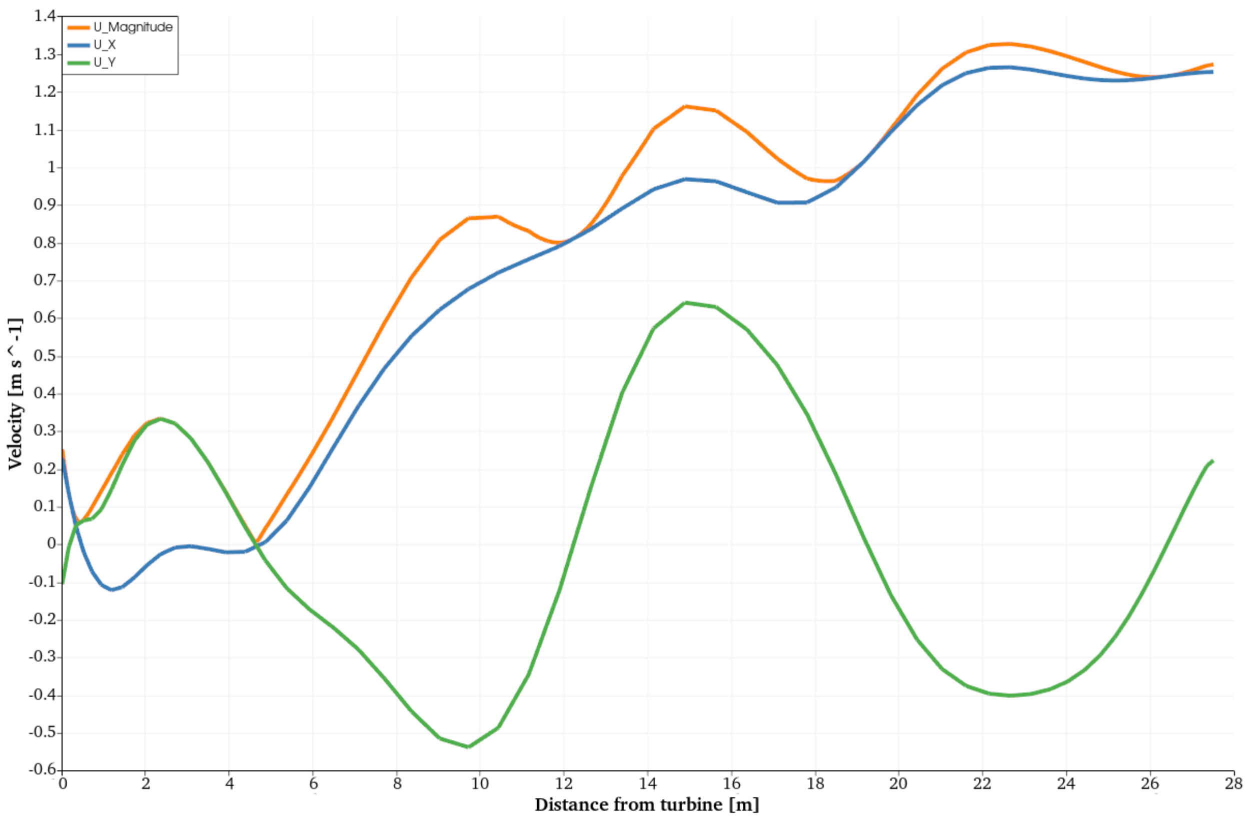


Figure J.4.1: Velocity along the wake of the elliptic Savonius with augmentations