

POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Corso di Laurea Magistrale in
Ingegneria Meccanica



CFD ANALYSIS OF H-DARRIEUS TURBINES FOR
HYDROKINETIC APPLICATIONS

Relatore: Prof. Giacomo PERSICO

Tesi di Laurea di:

Michele PIROTTA 899063

Luca SISSA 897575

Anno Accademico 2019/2020

Acknowledgements

We want to express our deepest gratitude to Professor Giacomo Persico for his precious guidance throughout this thesis work. His amazing enthusiasm kept us constantly engaged, and his willingness set us through a constant confrontation during the research.

A special thanks goes to our parents and families, who guided and supported us constantly in all our choices.

Contents

Acknowledgements	I
Contents	i
List of Figures	iii
List of Tables	v
Abstract	vii
Sommario	ix
1 Introduction	1
2 State of the art	4
2.1 HK turbines design overview	10
2.2 Confinement techniques	10
2.3 Channel design	12
2.4 Multiple turbines configurations	13
2.5 Vertical axis turbines classification	14
2.6 Physical modelling	15
2.7 H-Darrieus optimization	21
3 Computational Fluid Dynamics	25
3.1 The conservation laws	25
3.2 Turbulence Modelling	28
3.2.1 The $kT - kL - \omega$ eddy-viscosity transitional model	31
3.2.2 The equations	33
3.2.3 The $k - \omega SST$ eddy-viscosity model	36
3.2.4 The equations	37
3.3 CFD solver algorithms	38
4 OpenFOAM	41
5 Case setup	44
5.1 Domain general features and boundary conditions	47
5.2 Solver and numerical schemes	51
5.3 Mesh single turbine	53
5.4 Mesh non-engaging counter rotating turbines	59
5.5 Mesh engaging counter rotating turbines	62
6 Results	67

6.1	Single Turbine.....	69
6.2	Counter rotating non-engaging turbines.....	71
6.3	Counter rotating engaging turbines.....	74
6.4	Upscaled Turbine.....	77
6.5	Blockage Analysis	80
7	Conclusions	82
Appendix A.....		85
	Dictionaries for the mesh generation process	85
	Dictionaries for the simulation settings definition.....	97
References		100

List of Figures

Figure 2.1 - Conventional hydro versus hydrokinetic energy conversion schemes [18].	4
Figure 2.2 - General technology status of hydrokinetic turbine technologies [7].	5
Figure 2.3 - Outline of a hydrokinetic energy converter system [19].	5
Figure 2.4 - Classification of turbine rotors [7].	6
Figure 2.5 - Horizontal axis turbines (a) and vertical axis turbines (b) [7].	7
Figure 2.6 - Percentage of turbines considered for various placement arrangements; horizontal axis (a) and vertical axis (b) [7].	8
Figure 2.7 - Comparison of ducted and un-ducted SHP turbine [33].	11
Figure 2.8 - Shroud effect on streamlines [34].	11
Figure 2.9 - Velocity field around a turbine with and without a diffuser [34].	12
Figure 2.10 - Channel shapes (plan and front view) [7].	12
Figure 2.11 – Rendering of the double turbine configuration and its schematics studied in [36].	13
Figure 2.12 - Windside WS-4 Savonius turbine [38].	14
Figure 2.13 - H-Darrieus (a), Delta Darrieus (b) and Darrieus (c) turbines [39].	15
Figure 2.14 - Velocity and pressure fields along the x coordinate of the domain [40].	16
Figure 2.15 - Actuator disc and boundary stream tube model [41].	16
Figure 2.16 – Dependence of the velocity triangle of the blade with respect to its azimuthal position [39].	18
Figure 2.17 – System of fluid dynamic forces acting on the blade [39].	20
Figure 2.18 - Re and camber effects on stall angle [64][65].	22
Figure 2.19 - Darrieus type cross flow HKTs with straight and helical blades (a) [42] and the effect of blade tilt on energy per cycle (b) [45]. Positive energy means turbine will self-start and accelerate up to full speed.	22
Figure 2.20 – Variable-pitch H-Darrieus turbine [47].	23
Figure 3.1 – Control volume.	25
Figure 4.1 - Generic analysis structure in OpenFOAM	41
Figure 4.2 - Typical folder tree for OpenFOAM simulations	42

Figure 5.1 - Computational domains for the single turbine (a), engaging (b), non-engaging (c) configurations.....	46
Figure 5.2 - Single turbine blockMesh discretization.....	53
Figure 5.3 - Castellated mesh of the single turbine domain.....	54
Figure 5.4 - Blade profile in castellated mesh	55
Figure 5.5 - Meshed rotor region after snapping	55
Figure 5.6 - Blade profile after snapping	56
Figure 5.7 - Blade profile after layering.....	57
Figure 5.8 - Blade trailing edge with layers	57
Figure 5.9 - Layered blade leading edge	58
Figure 5.10 - Counter rotating configurations	59
Figure 5.11 - Castellated mesh of the non-engaging turbines domain.....	60
Figure 5.12 - Meshed rotors region	61
Figure 5.13 - Meshed blade profile	61
Figure 5.14 - Undeformed engaging turbines mesh.....	63
Figure 5.15 - Undeformed blade profile mesh	63
Figure 5.16 - Undeformed mesh detail.....	65
Figure 5.17 - Deformed mesh detail	65
Figure 6.1 – Single turbine power coefficient curves comparison.....	69
Figure 6.2 - Single turbine and non-engaging turbines power coefficient curves comparison.....	71
Figure 6.3 – Average torque behaviour (a) and 50% confidence interval (b) for different configurations	72
Figure 6.4 - Velocity field comparison: single turbine (a) vs non engaging turbines (b)	73
Figure 6.5 - Overall torque behaviour from engaging turbines configuration ...	74
Figure 6.6 - Overall torque behaviour from single turbine configuration	75
Figure 6.7 - Single blade torque behaviour comparison	75
Figure 6.8 - Velocity fields comparison: engaging turbines (a) vs single turbine (b)	76
Figure 6.9 - Meshed blade profile	78
Figure 6.10 - Layered blade trailing edge	78
Figure 6.11 - Upscaled machine convergence behaviour	79
Figure 6.12 - Blockage analysis of the different turbines configurations	80

List of Tables

Table 1.1 – HK turbine and non-turbine systems.....	3
Table 2.1 – General characteristics of horizontal and vertical axis turbines [24].	10
Table 3.1 – Re effects on flow behaviour in a pipe.....	30
Table 3.2 – Parameters adopted for the Re evaluation.....	31
Table 3.3 – Model constants.....	35
Table 5.1 - Single turbine fundamental characteristics	45
Table 5.2 - Domains dimensions	47
Table 5.3 - Velocity and pressure boundary conditions	48
Table 5.4 – κ , ω and νT boundary conditions for the κ - ω SST model	49
Table 5.5 - kT , kL , ω and νT boundary conditions for the $kT - kL - \omega$ model...	50
Table 5.6 - Solvers settings in OpenFOAM.....	51
Table 5.7 - Adopted numerical schemes.....	52
Table 5.8 - Single turbine domain discretization	53
Table 5.9 - Refinement level for surfaces (S) and regions (R)	54
Table 5.10 - First layer thickness evaluation	56
Table 5.11 - dynamicMeshDict settings	58
Table 5.12 - Non-engaging turbines domain discretization.....	60
Table 5.13 - Engaging turbines domain discretization	62
Table 5.14 - pointMotionU boundary conditions.....	64
Table 6.1 - Torque and convergence behaviour for different flow regimens	70
Table 6.2 - Pressure torque, viscous torque, net torque and power coefficient comparison between single and non-engaging turbines.....	73
Table 6.3 - Pressure torque, viscous torque, net torque and power coefficient comparison between single and engaging turbines.....	74
Table 6.4 - Upscaled non-engaging turbines fundamental characteristics	77
Table 6.5 - First layer thickness evaluation	78
Table 6.6 - Large scale machine performance evaluation	79

Abstract

The present thesis aims at proposing an accurate performance analysis of hydrokinetic vertical axis turbines, with particular focus on multiple turbines configurations in channel.

As over the last few years, renewable energy sources got more and more attention from the energy industry, mainly due to pollution effects on global climate and the overall electricity demand increment, the hydrokinetic research effort started considering energy sources on a smaller scale, such as vertical axis turbines in rivers or existing channels. These machines design is an inheritance from the wind energy industry, where the not satisfactory efficiencies and the overall technology costs prevented them from emerging as a competitive industrial alternative.

This work considers the specific H-type Darrieus turbines design, firstly patented by Georges Jean Marie Darrieus in 1926 for wind applications. As confirmed by recent literature from the wind field, this type of machines generally benefits from close distance arrangement, making it a promising opportunity for exploiting unconventional energy sources with limited available space. The following analysis hence considers two different counter rotating configurations, where the rotation centres of two closely spaced H-Darrieus turbines are arranged along the same cross-stream coordinate in a channel.

The performances analysis was carried out exploiting the CFD open source software OpenFOAM. Thanks to its built-in applications and utilities and to the customizability offered by the software package, it was possible to employ properly built meshes and solvers for each considered case study, granting a satisfactory accuracy level for the simulations results. The post processing stage was instead carried out using the Matlab software, which allowed an efficient elaboration of the simulations data.

This thesis results prove that, thanks to the reciprocal influence which the generated flow fields have on each other, multiple turbines configurations can actually perform better than the isolated turbine configuration, and, when introduced in a realistic scenario, vertical axis hydrokinetic turbines represent a solid addition to the already consolidated renewable energy technologies.

Sommario

Il presente lavoro di tesi mira a proporre un'accurata analisi delle prestazioni relative a turbine idrocinetiche ad asse verticale, con particolare attenzione a configurazioni caratterizzate da turbine multiple posizionate all'interno di un canale.

A causa degli effetti dell'inquinamento atmosferico esercitati sul clima e del continuo incremento della richiesta energetica globale, nel corso degli ultimi anni le fonti rinnovabili hanno acquisito sempre più importanza nell'industria energetica; conseguentemente, la ricerca scientifica legata al settore idrocinetico ha iniziato a considerare l'utilizzo di turbine ad asse verticale in fiumi o canali come fonte energetica su scala ridotta. Queste macchine derivano dall'industria eolica, dove le efficienze non soddisfacenti e i costi generali legati alla tecnologia non hanno permesso a questi dispositivi di emergere come valide alternative.

Nello specifico questo studio considera le turbine di tipo H-Darrieus, brevettate da Georges Jean Marie Darrieus nel 1926 per applicazioni eoliche. Come confermato dalla letteratura più recente, queste macchine traggono beneficio da configurazioni che le vedono poste a distanza ridotta l'una dall'altra, dando luogo ad una promettente opportunità di sfruttare sorgenti energetiche non convenzionali con poco spazio a disposizione. L'analisi riportata in questa sede prende in considerazione due differenti configurazioni di turbine controrotanti di tipo H-Darrieus all'interno di un canale, con i centri di rotazione posizionati a distanze ridotte lungo lo stesso piano ortogonale al flusso.

L'analisi delle prestazioni è stata portata termine sfruttando il software open source OpenFOAM per il calcolo CFD. Grazie alle utilities integrate e all'ampia possibilità di personalizzazione offerti dal software, è stato possibile procedere con la costruzione di mesh e l'implementazione di solvers specifici per ogni caso soggetto allo studio, garantendo un soddisfacente livello di precisione nei risultati ottenuti. La fase di post-processing è stata invece sviluppata tramite l'utilizzo del software Matlab, che ha permesso un'efficiente elaborazione dei dati provenienti dalle simulazioni.

I risultati ottenuti provano che, grazie alla reciproca influenza generata dai flussi, le configurazioni con turbine multiple possono raggiungere livelli di resa superiori a quelle singole e, se introdotte in uno scenario realistico, le turbine idrocinetiche ad asse verticale sono in grado di rappresentare una solida aggiunta alle già consolidate tecnologie rinnovabili.

1 Introduction

Renewable energy represents today approximately the 34% of global installed power capacity, with a total supplement of 2179 GW; hydropower is the largest contributor, accounting for 1151 GW [1, 2]. Due to climate change crisis and continuous increase of global electricity demand, in order to reduce carbon emission, a strong step in favour of renewables is highly requested. Ideally, as reported in [3], “*a renewable energy conversion technology should have a minimum cost per annual average energy production as well as minimal and mitigatable environmental impacts with a maximum power output*”. One of the possible solutions, in addition to the adoption of mature and cost-effective renewable technologies such as solar, hydro and wind turbines [4], would be concentrating efforts on extracting untapped energy reserves, including low-head (potential) hydropower and hydrokinetic (HK) power conversion systems [3], whereby energy is extracted from kinetic energy of flowing water.

HK energy is mainly extracted from oceans, waterflows in rivers, channels and constructed waterways; among them, marine current and wave energy from the oceans are the most promising form of energy sources [5]. Various unexploited structures like irrigation canals, rivers or low height dams, where available potential energy is very limited, may represent a new important starting point, also for bringing energy to rural and remote areas of the world [6]. As reported in [3], due to their intrinsic nature, these resources do not require the construction of new dams or diversion systems and can exploit already existing water-infrastructures, which makes them a solid alternative from an avoiding huge environmental impacts perspective.

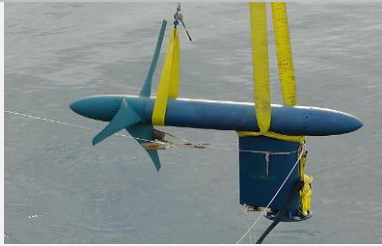


According to [7], the main classification system divides HK devices in turbines and non-turbines technologies; a brief summarization is reported below, while Table 1.1 proposes a graphical representation.

Turbine systems:

- Axial (Horizontal): rotational axis of the rotor is parallel to the incoming water stream.
- Vertical: rotational axis of the rotor is normal to the water surface and to the incoming water stream as well.
- Cross-flow: rotational axis of the rotor is parallel to the water surface but orthogonal to the incoming water stream.
- Venturi: accelerated water resulting from a choked system which creates a pressure gradient is used to run an in-built or on-shore turbine.
- Gravitational vortex: artificially induced vortex effect is used in driving a vertical turbine.

Non-turbine systems:

- Flutter Vane: systems that are based on the principle of power generation from hydro-elastic resonance (flutter) in free-flowing water.
- Piezoelectric: piezo-property of polymers is exploited for electricity generation when a sheet of material is placed in the water stream.
- Vortex induced vibration: exploits vibrations resulting from vortices forming and shedding on the downstream side of a bluff body in a current.
- Oscillating hydrofoil: vertical oscillation of hydrofoils can be used for the generation of pressurized fluids and subsequent turbine operation. A variant of this class includes biomimetic devices for energy harvesting.
- Sails: Employs drag motion of linearly/circularly moving sheets of foils placed in a water stream.

<p>Axial-flow turbine</p> <p>Rite GEN 4 Demonstration [8]</p>	
<p>Vertical axis turbine</p> <p>Yakima Hydrokinetic Project [9]</p>	
<p>Cross-flow turbine</p> <p>Atlantisstrom plant [10]</p>	
<p>Venturi</p> <p>HydroVenturi [11]</p>	






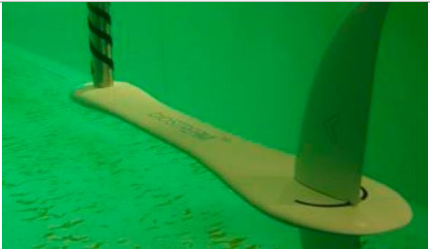
<p>Gravitational vortex</p> <p>Zotlöterer plant [12]</p>	
<p>Flutter vane</p> <p>Oscillating cascade power system [13]</p>	
<p>Piezoelectric</p> <p>The energy harvesting Eel [14]</p>	
<p>Vortex induced vibrations</p> <p>VIVACE (Vortex Induced Vibrations for Aquatic Clean Energy) [15]</p>	
<p>Sails</p> <p>Deep Green [17]</p>	
<p>Oscillating hydrofoil</p> <p>bioSTREAM [16]</p>	

Table 1.1 – HK turbine and non-turbine systems.

2 State of the art

In 2006, the US Department of Energy defined HK systems as “Low Power/Unconventional Systems” that may use hydro resources with less than 8 feet head [18]; a graphical comparison between conventional turbine hydropower systems and un-conventional HK devices is showed in Figure 2.1.

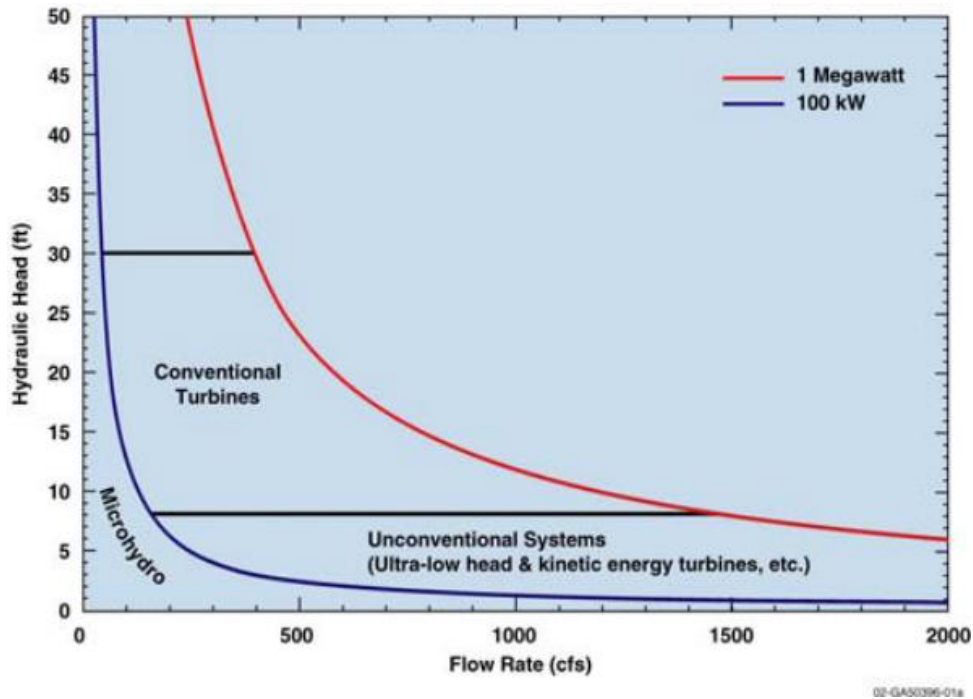


Figure 2.1 - Conventional hydro versus hydrokinetic energy conversion schemes [18].

As previously discussed, hydrokinetic energy conversion systems employ both turbine and non-turbine technologies: non-turbine technologies are mostly at the proof-of-concept stage, except for some isolated cases, whilst turbine devices have been widely studied and represent a concrete opportunity for near future [7]. As showed in Figure 2.2 (a), a broad survey of existing and discontinued RD&D initiatives is explored and classified in various maturity groups (from ‘concept’ to ‘commercial’).

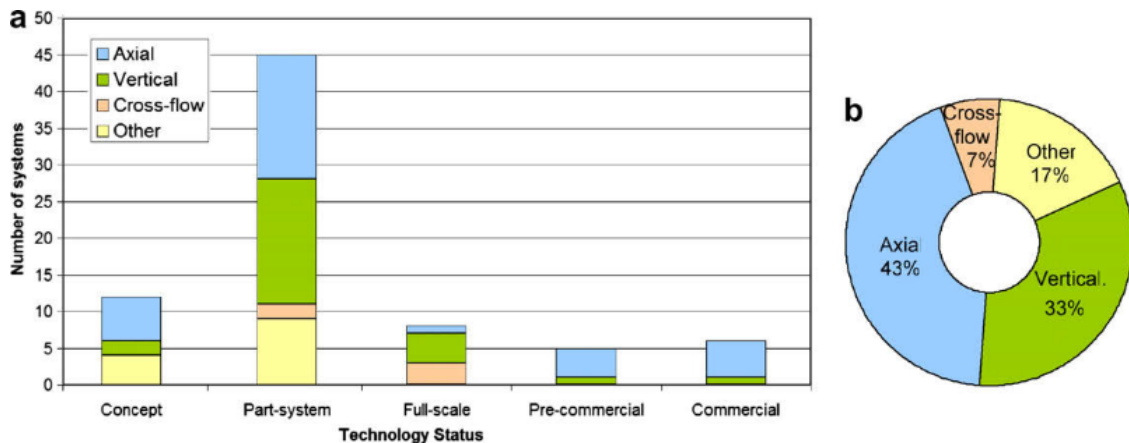


Figure 2.2 - General technology status of hydrokinetic turbine technologies [7].

Figure 2.2 (b) clearly shows that vertical and horizontal axis turbines share the widest part of the applications. This should not be surprising at all, as axial flow turbines are well known machines and vertical axis turbines are seeing a renewed interest, due to the almost total rejection from wind energy research.

As showed in Figure 2.3, a Hydrokinetic Energy Conversion system (HKEC) is mainly composed by five different parts [5]:

- HK turbine
- Generator
- Support structure
- Control system
- Transmission system

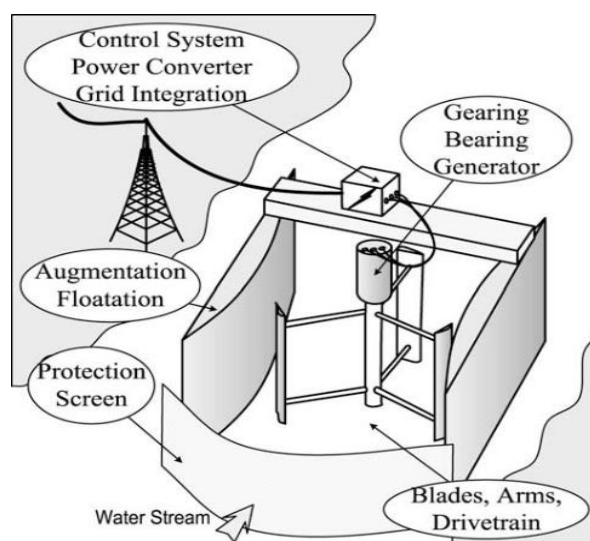


Figure 2.3 - Outline of a hydrokinetic energy converter system [19].

A general classification of horizontal, vertical and cross-flow turbines is provided in Figure 2.4.

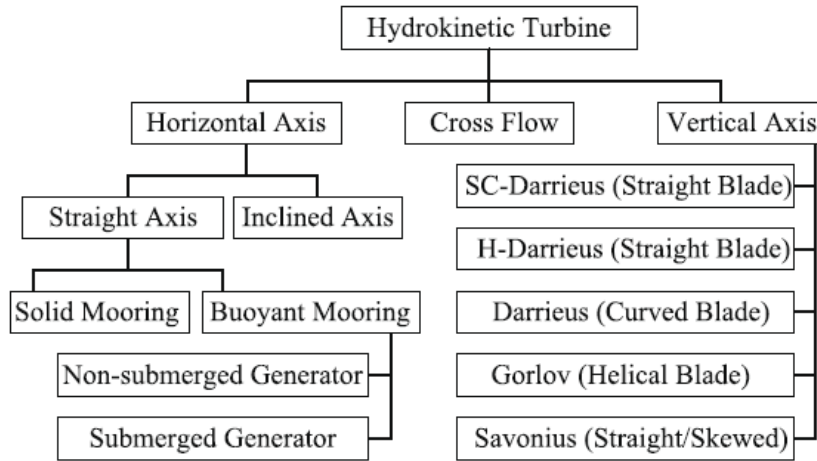


Figure 2.4 - Classification of turbine rotors [7].

The horizontal axis (axial-flow) turbines have the axis parallel to the fluid stream and employ propeller type rotors; various arrangements of these machines are presented in Figure 2.5 (a). Inclined axis turbines are included in this family as well: these devices have been mostly studied for small applications in rivers, but it is not clear whether they are still commercialized [7]. In general, axial-flow turbines are exploited in the oceans for tidal energy applications and, additionally, they are very similar to wind turbines, both conceptually and from the design point of view [7]. The cross-flow turbines (also known as floating waterwheels) present the axis perpendicular to the fluid flow and parallel to the water surface; being drag-driven machines, those are characterized by lower efficiencies when compared to their lift-based counterparts, and additionally the large amount of required material could represent another possible issue. Finally, the vertical axis turbines have axis orthogonal both to water surface and to the fluid stream and, among all the available configurations, Darrieus turbines represent the most prominent option [7]. Various arrangements of vertical axis turbines are showed in Figure 2.5 (b).

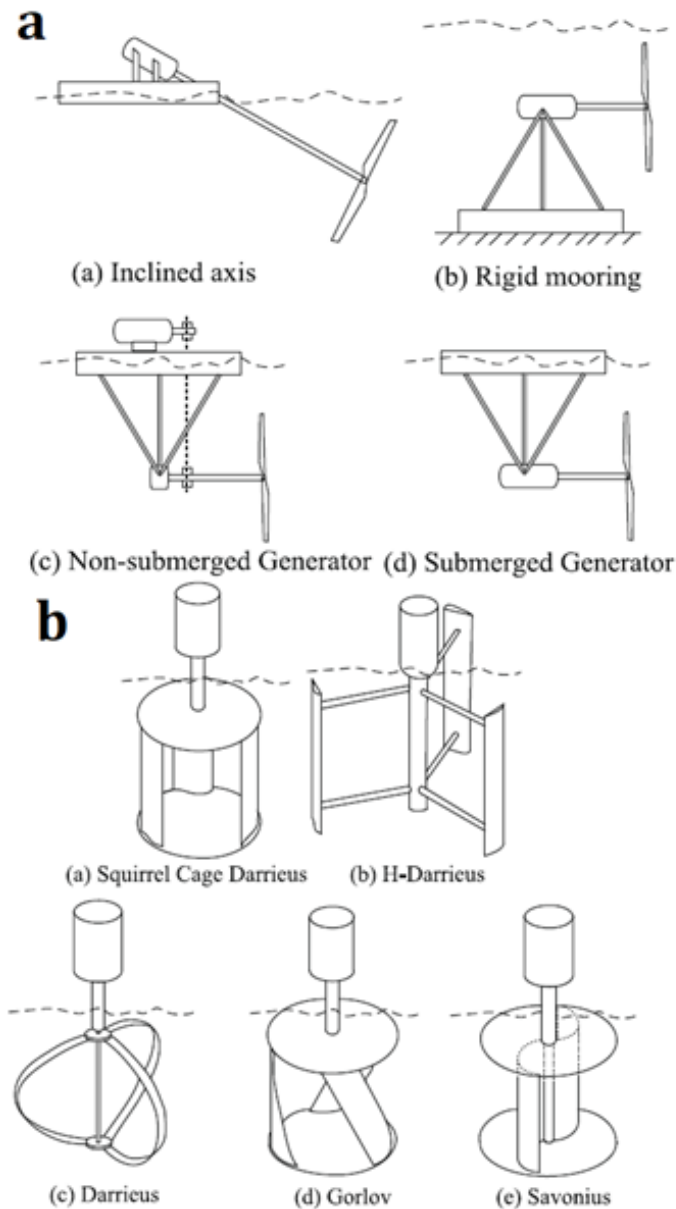


Figure 2.5 - Horizontal axis turbines (a) and vertical axis turbines (b) [7].

Concerning the rotor placement, according to [7] a generic turbine may incorporate:

- Bottom structure mounting (BSM), where the converter is fixed near seafloor/riverbed.
- Floating structure mounting (FSM).
- Near-surface mounting (NSM)

Figure 2.6 shows the current trend relative to turbine rotor placement for horizontal (a) and vertical axis (b) turbines. It is interesting to notice that more than half of the vertical axis turbine are assembled according to the NSM configuration, which grants the possibility of placing the generator above the

water level. At the present state of this technology, it is not completely clear whether a configuration is better than another one, mostly due to various issues related to the energy flux extraction (always higher near the water surface), competing users (fishing, shipping and recreational boating may object to FSM or NSM systems), constructive challenges and footprint [7].

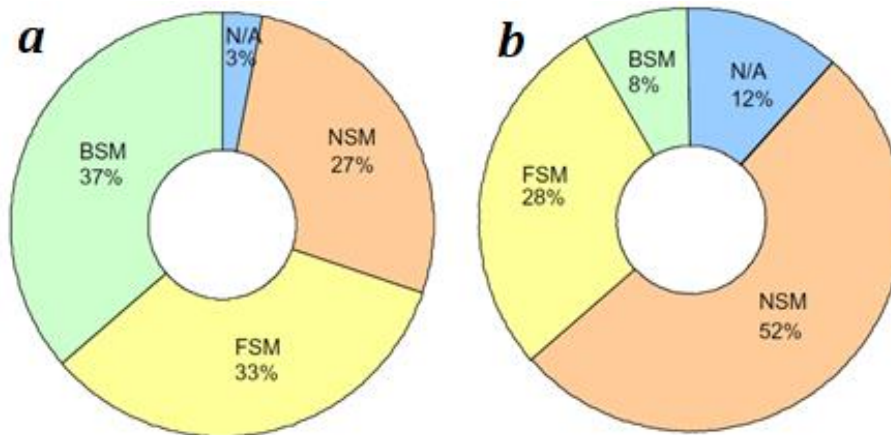


Figure 2.6 - Percentage of turbines considered for various placement arrangements; horizontal axis (a) and vertical axis (b) [7].

Of particular interest is a review of both horizontal and vertical axis turbines, which, as said, represent the widest part of exploited configurations. According to [7], vertical axis turbines, and especially straight blades Darrieus type, gained considerable attention due to some characteristic features, such as:

- *Design simplicity*: if compared to axial-flow turbines, being the blades straight, development and manufacturing costs are lower.
- *Generator coupling*: due to the machine configuration, the generator can be placed at one end of the shaft (e.g. above the water surface). This would not be possible for a horizontal axis machine, which requires a more complex design (e.g. underwater placement of the generator, right-angled gear coupling).
- *Flotation and augmentation equipment*: the cylindrical shape of Darrieus turbines allows convenient mounting of various curvilinear or rectangular ducts, exploitable also for mooring and floating uses [20].
- *Noise emission*: if compared to axial-flow counterparts, being blade tip losses lower for vertical axis turbines, also noise emission is reduced [21].
- *Skewed flow*: the vertical profile of water velocity variation in a channel may have significant impact on the turbine operations as, in a shallow channel, the upper part of a turbine faces higher velocity than the lower

section. Vertical turbines, especially the ones with helical/inclined blades, are reportedly more suitable for operation under such conditions [22].

Despite these remarkable advantages, vertical axis turbines suffer of low starting torque, torque ripple and low efficiencies. On the other hand, horizontal axis turbines eliminate many of these drawbacks and present other advantages, such as:

- *Knowledgebase*: studies and researches concerning axial-flow turbines are abundant, as these machines directly derive from wind and marine applications.
- *Performance*: concerning horizontal turbines, all the blades are designed to have sufficient taper and twist such that the lift forces are equally exerted along the blade height. As a consequence, these turbines are self-starting.
- *Control*: various control methods (stall or pitch regulated) have been widely studied for this class of machines.
- *Annular ring augmentation channels*: these types of augmentation channel provide greater augmentation of fluid velocity, being able to concentrate/diffuse flow in a three-dimensional manner [23].

As previously mentioned, axial-flow turbines present major issues related to blade design, underwater cabling and placement of the generator; a brief summarization of the main differences between horizontal and vertical axis turbines is reported in Table 2.1.

	Horizontal axis turbine	Vertical axis turbine
Minimum operating current velocity	<i>0.5 m/s</i>	<i>1 m/s – need higher velocity to self-start</i>
Operating blade speed ratio (BSR)	<i>Faster (BSR up to 4 -5)</i>	<i>Slower (BSR below 3)</i>
Coefficient of power Cp	<i>46%</i>	<i>35%</i>
Water to wire efficiency	<i>25% (calculated), less efficient transmission and generator</i>	<i>26% (claimed), efficient transmission and generator</i>
Debris resistant	<i>Poor</i>	<i>Good</i>
Torque ripple	<i>Smoother</i>	<i>Pulsating</i>

Rotor simplicity	<i>Fairly complex</i>	<i>Simple</i>
Material quantity and cost	<i>Less</i>	<i>More</i>
Weight	<i>Less</i>	<i>More</i>
Pontoon	<i>Smaller</i>	<i>Larger</i>
Mechanical power transmission	<i>Complex</i>	<i>Simple</i>

Table 2.1 – General characteristics of horizontal and vertical axis turbines [24].

2.1 HK turbines design overview

As mentioned in [3], low efficiency, cavitation, installation costs and unpredictable maintenance of the machines are the main problems related to the development of these technologies. In fact, optimizing a rotor does not necessarily mean reaching the maximum annual energy production, but maximizing the machine efficiency to approach the Betz limit [3], a parameter commonly adopted to fix an upper limit on the highest efficiency level achievable. However, from the power production point of view, it is known that higher velocities (usually around 2 – 3 m/s [3]), if included within the operational range, can exponentially increase the power output. Even though various models have been developed for possible enhancing techniques, in general the results validation in-field is rare [3]. In fact, as reported in [24, 25, 26, 27, 28], lifespan and operational ranges of tested machines have been adversely affected by environmental variations (e.g. clogging due to river borne plants, floating leaves, seaweed and mangrove trees), causing problems to both turbines and gearboxes. Moreover, a possible increment of the total drag could lead to a more expensive turbine anchoring system, and hence to an increase of debris accumulation. According to [3], there are a few viable HK devices enhancement techniques, which are presented in the following sections.

2.2 Confinement techniques

The confinement technique refers to the arrangement of the turbine within a confined structure (i.e. a duct); this strategy, despite finding wide applications in tidal machines, it has also been extensively studied for wind turbines. Even though the contradictory study [32] based on a BEM-RAS model shows a consistent decrease of hydrodynamic forces along the axial flow, leading to a further decrease in power output, other various studies from the literature [29, 30, 31] ensure a sensible increase in power output. As an additional advantage, it seems that in some cases ducts are able to protect the rotor from debris and other

extreme conditions; anyway, as showed in Figure 2.7, the higher the operational velocities are, the larger is the available power output [29].

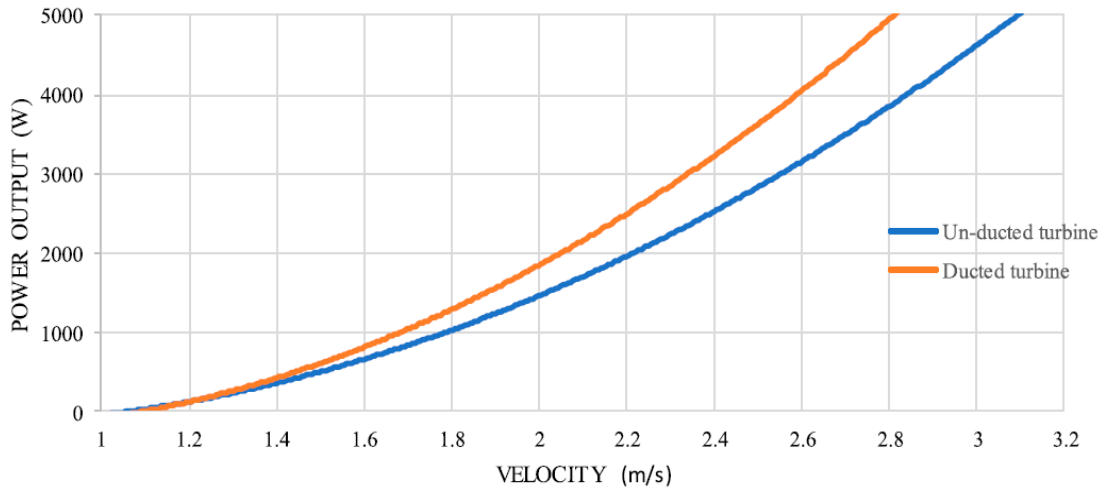


Figure 2.7 - Comparison of ducted and un-ducted SHP turbine [33].

Another possible feature that could increase the HK devices efficiency would be the shroud mechanism, which it seems to be capable of accelerating the fluid passing through the machine. The comparison between two different flow streams passing through a shrouded and an unshrouded turbine shows that, in the first case, there is an increment of the fluid volume passing through the machine, and hence an enlargement of the power output. However, also in this case, computational analysis provided contradictory results between experiments and theoretical modelling.

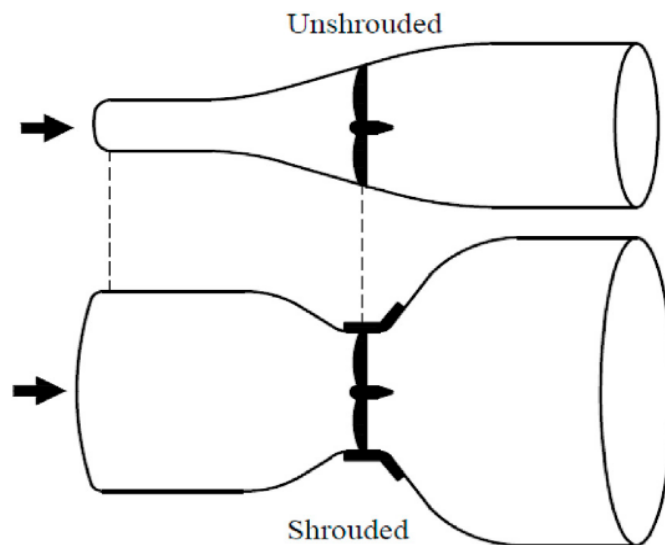


Figure 2.8 - Shroud effect on streamlines [34].

A great number of studies also indicates that a diffuser could have positive effects on the power output of a HK turbine. This device works similarly to a shroud, but it is placed downstream the machine, encircling and diverging outwards at a designed angle. As stated by Gaden [34]: “*The use of a diffuser at the outlet may seem counter-intuitive as diffusers reduce flow speed and increase pressure. However, it is the upstream effect of this that benefits the turbine.*”

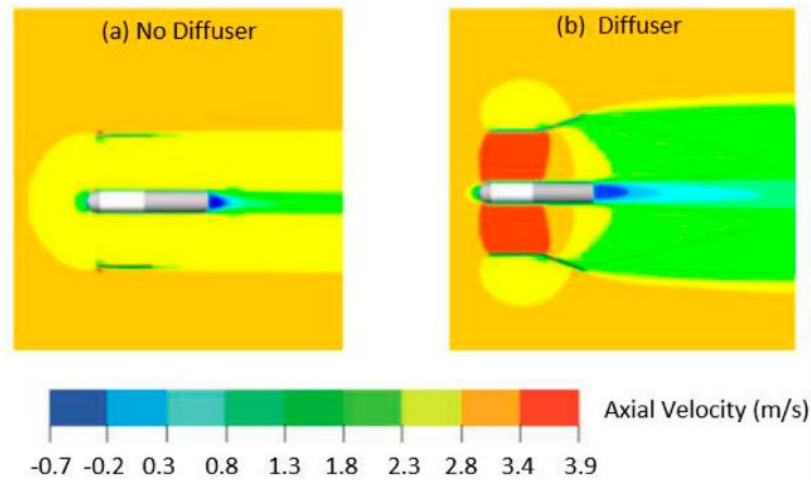


Figure 2.9 - Velocity field around a turbine with and without a diffuser [34].

2.3 Channel design

Modified channels exploit the Venturi’s principle to induce a sub-atmospheric pressure within a constrained area, increasing the flow velocity [7]; hence, a turbine working in this region would increase the power output due to the velocity increment in the rotor region. Typical channel modifications are reported in Figure 2.10.

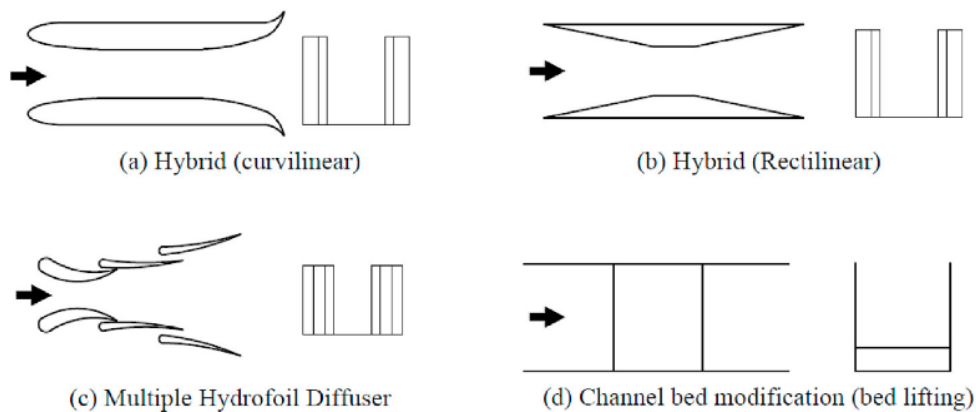


Figure 2.10 - Channel shapes (plan and front view) [7].

2.4 Multiple turbines configurations

Various studies indicate that multiple turbines configurations affect the system energy output, and researchers have analysed the link between the total power extracted and the amount of power dissipated by the presence of the HK devices [3]. Multiple possible configurations exist, and the most relevant issues are related to the eventual severe clogging effect along the channel cross-section and the partial unpredictability of the inter-effects between the devices, in terms of dissipations and flow disturbances. In this thesis, two particular configurations involving multiple turbines are analysed and both aim at evaluating the flow passing through two counter rotating turbines, firstly non-engaging and then engaging. In [35] a 2D CFD simulation of the flow passing through two counter rotating H-Darrieus turbines for wind applications has been carried out, showing an increase of the normalized tangential force on both the windward and leeward path, an increase in the power coefficient at all the relative distances tested and an increase of the optimal blade speed ratio for the counter rotating turbines with respect to the isolated machine. The studied configuration is showed in Figure 2.11.



Figure 2.11 – Rendering of the double turbine configuration and its schematics studied in [36].

Zanforlin and Nishino in [37] analysed two counter rotating vertical axis wind turbines, pointing out that the same registered benefits would apply also for tidal and marine turbines. Additionally, also in [36] the authors studied a configuration of two counter rotating VATs inside the wind tunnel of Politecnico di Milano, registering an increment of the power coefficient.

2.5 Vertical axis turbines classification

As previously mentioned, existing structures like rivers or irrigation canals could be exploited in combination with HK turbines in order to extract energy from the waterflow; usually, in these situations vertical axis machines represent the best solution. Vertical axis turbines (VATs) can be divided in two different categories:

- Fluid flow
- Forcing system

As for the fluid flow category, Vertical Axis Wind Turbines (VAWTs) were the first vertical axis turbine to be studied, but then, after the almost total rejection by the wind sector, these machines started to get relevant attention by the hydrokinetic field; the same technologies were then applied for the design of Vertical Axis Hydrokinetic turbines (VAHKs). In fact, despite the different type of fluid involved, the overall problem remained incompressible, as VAWTs Mach number was always below the incompressible threshold.

Concerning the forcing system category, there are two different types of VATs: drag driven and lift driven turbines. The drag driven VATs exploit the asymmetry in the geometry of their moving parts in order to generate different drag forces over the whole machine rotation, hence generating a torque. An example of one of the most common drag driven VATs is the Savonius turbine, represented in Figure 2.12.



Figure 2.12 - Windside WS-4 Savonius turbine [38].

Lift driven VATs are much more complex than their drag counterparts and their efficiencies are higher as well. These machines were firstly studied by the

aeronautical engineer Georges Jean Marie Darrieus, who in 1926 patented the first prototypes of H-Darrieus, Delta Darrieus and Darrieus turbines, represented respectively in Figure 2.13.

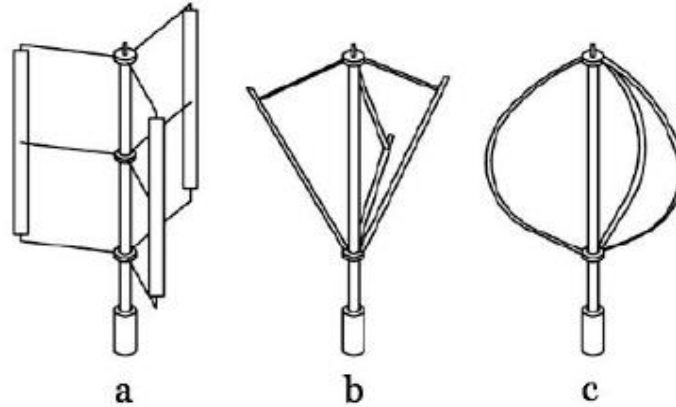


Figure 2.13 - H-Darrieus (a), Delta Darrieus (b) and Darrieus (c) turbines [39].

Despite the higher efficiencies, these machines suffer from low self-starting capabilities with respect to their drag driven counterparts and hence require a particular starting mechanism.

2.6 Physical modelling

According to literature, vertical and horizontal axis turbines can be modelled following multiple approaches: this chapter firstly introduces the simplest theory of the actuator disc, then presents the more refined blade element theory and a brief discussion about the topic. Despite the undiscussed validity of those methods though, this study employs a computational fluid dynamics (CFD) approach, in order to achieve even more precise results.

The actuator disc theory is based on a simple 1D momentum analysis in which the turbine rotor is replaced by an actuator disc. Even though some simplifying assumptions concerning the flow are required, the final results provide useful formulations widely employed and also adopted in this thesis.

The basic assumptions are:

- I. Steady uniform flow upstream of the disc.
- II. Uniform and steady velocity at the disc.
- III. No flow rotation produced by the disc.
- IV. Flow passing through the disc is contained both upstream and downstream by the boundary stream tube.
- V. Incompressible flow.

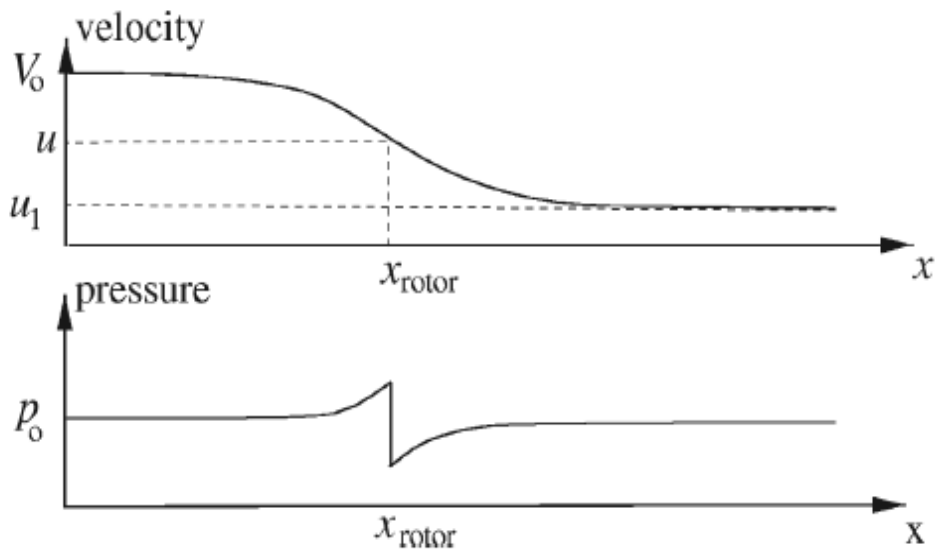


Figure 2.14 - Velocity and pressure fields along the x coordinate of the domain [40].

Figure 2.14 shows the velocity and pressure behaviours over the x coordinate for the considered domain length; the disc works as a resistance for the fluid stream, and therefore it generates a pressure increment upstream the disc, an abrupt pressure decay at the disc coordinate, and finally a gradual recovery towards downstream. As for the velocity, firstly the fluid flow decelerates approaching the rotor, then, due to the energy extraction process, a further decrease in velocity downstream the turbine takes place. Referring to Figure 2.15, it is possible to identify three different sections (1, 2 and 3) related to different planes, respectively upstream, in correspondence and downstream the actuator disc, where c_{x1} , c_{x2} and c_{x3} represent the velocity intensity of the stream.

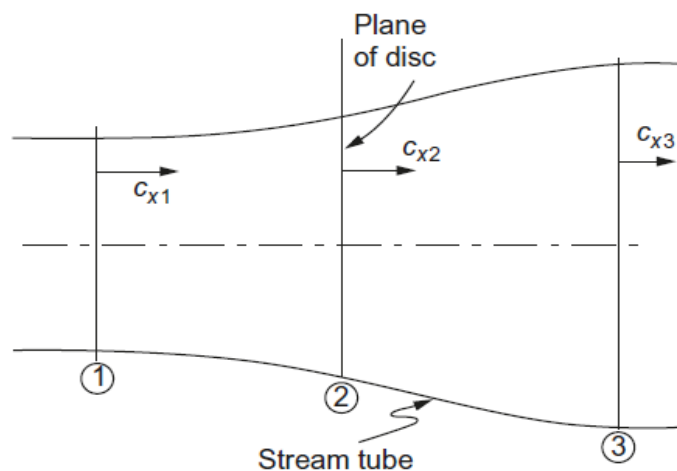


Figure 2.15 - Actuator disc and boundary stream tube model [41].

From the continuity equation, the mass flow rate is computed:

$$\dot{m} = \rho c_{x2} A_2 \quad (2.1)$$

where A_2 is the area of the actuator disc section and ρ the fluid density. The axial force acting on would then be:

$$X = \dot{m}(c_{x1} - c_{x3}) \quad (2.2)$$

Therefore, the power extracted by the turbine may be computed as:

$$P = X c_{x2} = \dot{m}(c_{x1} - c_{x3}) c_{x2} \quad (2.3)$$

Now it can be safely assumed that, in an ideal case, the power loss by the fluid would be equal to the power extracted by the disc:

$$\frac{1}{2} \dot{m}(c_{x1}^2 - c_{x3}^2) = \dot{m}(c_{x1} - c_{x3}) c_{x2} \quad (2.4)$$

This way Betz (1926) demonstrated that the velocity at the disc section is the mean between the velocities far upstream and downstream the rotor:

$$c_{x2} = \frac{1}{2}(c_{x1} + c_{x3}) \quad (2.5)$$

Combining (2.3) with (2.1), it is possible to obtain:

$$P = \rho A_2 c_{x2}^2 (c_{x1} - c_{x3}) \quad (2.6)$$

and inverting (2.5):

$$c_{x3} = 2c_{x2} - c_{x1} \rightarrow (c_{x1} - c_{x3}) = c_{x1} - 2c_{x2} + c_{x1} = 2(c_{x1} - c_{x2}) \quad (2.7)$$

Therefore, the expression for the extracted power can be rewritten as:

$$P = 2\rho A_2 c_{x2}^2 (c_{x1} - c_{x2}) \quad (2.8)$$

It is now convenient to define an induction factor a , in order to account for the velocity reduction due to the presence of the rotor:

$$a = \frac{(c_{x1} - c_{x2})}{c_{x1}} \quad (2.9)$$

Rewriting (2.6) according to the induction factor formulation, the final power expression results:

$$P = 2a\rho A_2 c_{x1}^3 (1 - a)^2 \quad (2.10)$$

Finally, identifying P_0 as the unperturbed wind power, accounting for the same disc area A_2 , it is possible to define the power (or performance) coefficient C_p as:

$$C_p = \frac{P}{P_0} = \frac{2a\rho A_2 c_{x1}^3 (1 - a)^2}{\frac{1}{2}\rho A_2 c_{x1}^3} = 4a(1 - a)^2 \quad (2.11)$$

The maximum value for the power coefficient can be evaluated by differentiating the previous expression with respect to the induction factor, obtaining:

$$\frac{dC_P}{da} = 4(1 - a)(1 - 3a) = 0 \rightarrow a_1 = \frac{1}{3}; \quad a_2 = 1$$

$$\text{For } a_1 \rightarrow C_{P,max} = \frac{16}{27} = 0.593$$

The computed C_P value represents the so-called Betz limit, the maximum power coefficient achievable by the turbine with the previous prescribed flow conditions. Therefore, it is possible to define again the power extracted by the rotor as:

$$P = \frac{1}{2} \rho A_2 c_{x1}^3 C_P \quad (2.12)$$

Despite the useful results achieved, the previous power coefficient expression does not consider any influence of other parameters, including solidity, blade speed ratio (BSR), lift and drag coefficients of the adopted blades and velocity triangles related to the interaction between the fluid and the blades themselves, as showed in Figure 2.16. A more precise formulation can be provided exploiting a 2D model, the Blade Element Theory (BET), based on the definition of the forcing system and the energy transfer for a blade of infinitesimal extension dz . The performance of the machine is analysed by evaluating the superimposition effects given by the N blades, each considered as isolated.

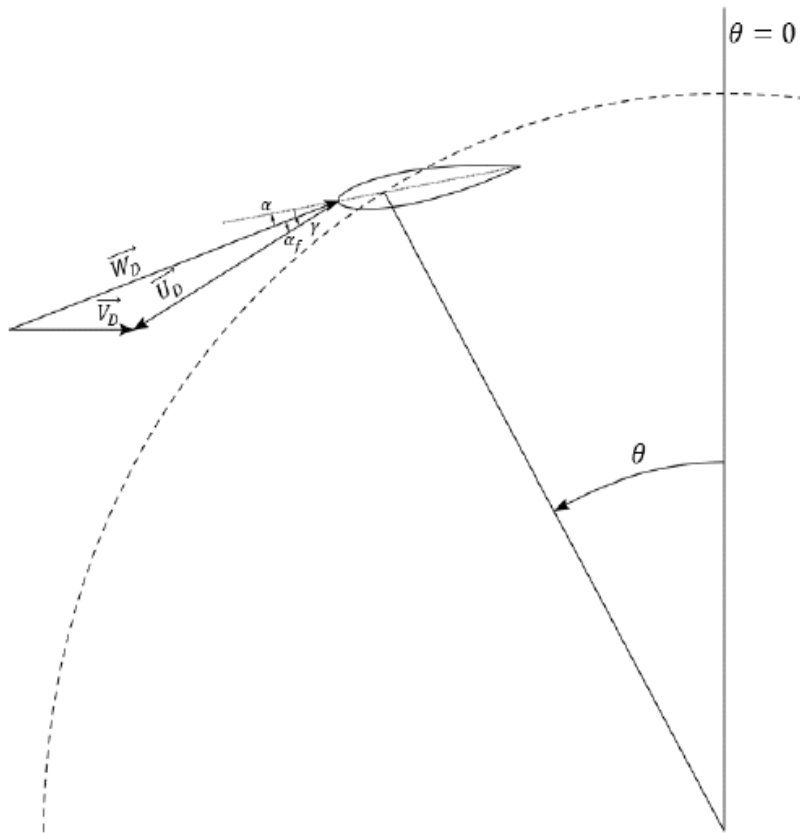


Figure 2.16 – Dependence of the velocity triangle of the blade with respect to its azimuthal position [39].

The blade angle of attack α depends on the azimuthal angle θ and on the BSR λ , defined as:

$$\lambda = \frac{\omega R}{c_{x1}} \quad (2.13)$$

The equation describing the angle of attack is:

$$\alpha = \text{atan} \left(\frac{(1-a)\sin \theta}{(1-a)\cos \theta + \lambda} \right) - \gamma \quad (2.14)$$

where γ is the stagger angle. In general, at a given azimuthal angle the blade stalls; nevertheless, if the BSR is high enough, it is possible to avoid the stall over the whole turbine rotation. Figure 2.17 shows the forcing system acting on the blade element, which results from the interaction between lift and drag forces, respectively dL and dD ; the resulting unique force dF can be further decomposed in its tangential and normal components, dF_T and dF_N . The lift and drag infinitesimal components are defined as:

$$dL = \frac{1}{2} \rho c_L W_D^2 c \, dz \quad (2.15)$$

$$dD = \frac{1}{2} \rho c_D W_D^2 c \, dz \quad (2.16)$$

where c_L and c_D represent respectively the lift and drag coefficients, W_D is the relative velocity between the fluid and the blade element and c is the blade chord. The tangential and normal components are then:

$$dF_T = \frac{1}{2} \rho (c_L \sin \alpha_f - c_D \cos \alpha_f) W_D^2 c \, dz \quad (2.17)$$

$$dF_N = \frac{1}{2} \rho (c_L \cos \alpha_f + c_D \sin \alpha_f) W_D^2 c \, dz \quad (2.18)$$

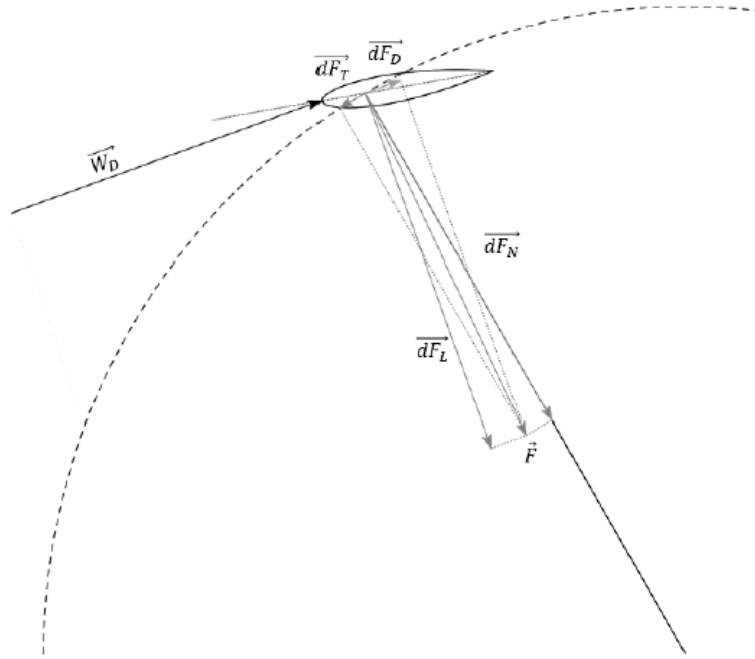


Figure 2.17 – System of fluid dynamic forces acting on the blade [39].

The overall torque produced, considering the N blades is:

$$dT = \frac{1}{2} \rho N R (c_L \sin \alpha_f - c_D \cos \alpha_f) W_D^2 c dz \quad (2.19)$$

and the total power is defined as:

$$dP = \omega dT = \frac{1}{2} \rho \omega N R (c_L \sin \alpha_f - c_D \cos \alpha_f) W_D^2 c dz \quad (2.20)$$

Remembering the C_p definition and defining the solidity as:

$$C_p = \frac{dP}{\frac{1}{2} \rho c_{x1}^3 D dz} \quad (2.21)$$

$$\sigma = \frac{Nc}{D} \quad (2.22)$$

it is possible to define the power coefficient in a more complete fashion:

$$C_p = \lambda \sigma (1 - a)^2 \left(\sin^2 \alpha_f + \left(\frac{\lambda}{1 - a} + \cos \alpha_f \right)^2 \right) (c_L \sin \alpha_f - c_D \cos \alpha_f) \quad (2.23)$$

This expression provides a much more detailed expression concerning the main parameters affecting C_p ; in particular, it is important to notice that the BSR affects the angle of attack, and therefore the lift and drag coefficients. Moreover, also the solidity must be taken into account due to its influence on the induction factor.

2.7 H-Darrieus optimization

As already mentioned in Section 1 of the present work, one of the main drawbacks of vertical axis turbines is related to the poor self-starting torque, caused by the continuous variation of the angle of attack α with respect to the azimuthal angle ϑ , especially for low blade speed ratios. Due to this particular behaviour, the machine continuously stalls, and it is subject to low efficiencies and important vibrations related to the torque ripple phenomenon. The variation of α is linked with variation of the tangential and radial forces exerted on the blades, which experience two peaks in both directions, approximately 180° apart. The tangential component variation affects the transmission, whilst the radial component affects the support structure, potentially causing huge failures due to the possible coincidence with the natural frequencies of the structure [42]. Obviously, H-Darrieus turbines are not immune to these issues and further investigations in order to improve their overall efficiencies are an actual field of study.

According to Kirke B.K. and Lazauskas L. studies [42], there are specific parameters that mostly affect the turbine performance. Firstly, as it is reported in Figure 2.18, it seems that high cambered blades are able to provide both high lift values and stall resistance; however, it may result also in higher drag values, still generating high resistant torque at the start, especially if blades are fouled by weed or corrosion [43]. At the same time, also high Reynolds blade chords tend to improve the blade stall angle; however, increasing the blade chord means increasing also the machine solidity σ . Higher solidity would bring to a high starting torque [44], but also to lower peak efficiencies and blade speed ratios. Therefore, acceptable efficiency values can only be achieved for large turbines with low solidities, high chord Reynolds numbers and high blade speed ratios where stall does not occur; nevertheless, due to low solidity, machines experience poor self-starting ability, and additionally high blade speed ratios may lead to stress and cavitation problems. In conclusion, a proper trade off must be achieved.

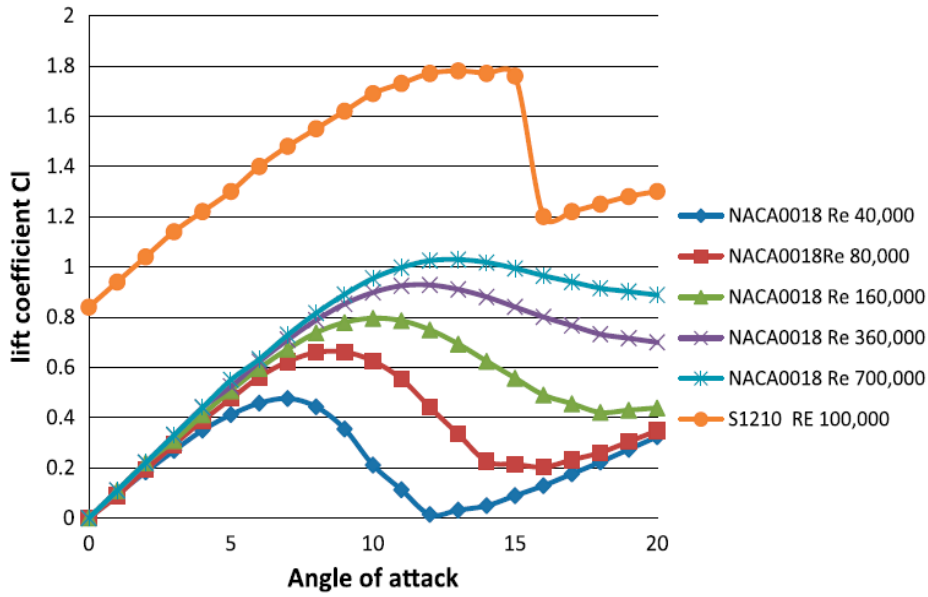


Figure 2.18 - Re and camber effects on stall angle [64][65].

An alternative design strategy would be exploiting helical blades instead of the straight configuration, as showed in Figure 2.19 (a); however, even though the use of inclined blades would lead to a smaller variation of the angle of attack versus the azimuthal angle, as showed in Figure 2.19 (b), the minimum required inclination for the machine to self-start is set to 40° , much greater than the one usually implemented for the most common helical turbines, therefore potentially affecting the reliability of the machine [45].

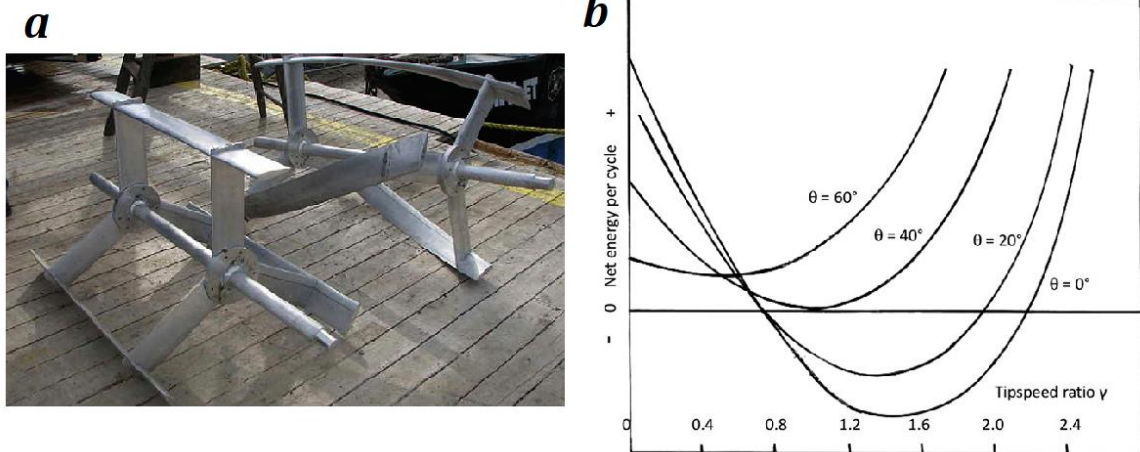


Figure 2.19 - Darrieus type cross flow HKTs with straight and helical blades (a) [42] and the effect of blade tilt on energy per cycle (b) [45]. Positive energy means turbine will self-start and accelerate up to full speed.

Finally, concerning vibrations related to the torque ripple, on one hand turbines with a higher number of blades are able to avoid torque fluctuations and reduce shaking, but, on the other hand, these machines do not prevent stall; a higher blade number would also mean higher solidity, lower blade speed ratios and lower efficiency. Helical turbines reduce the vibration problem because blades do not stall simultaneously along their full length, and additionally stall occurs less suddenly if compared to the straight counterpart.

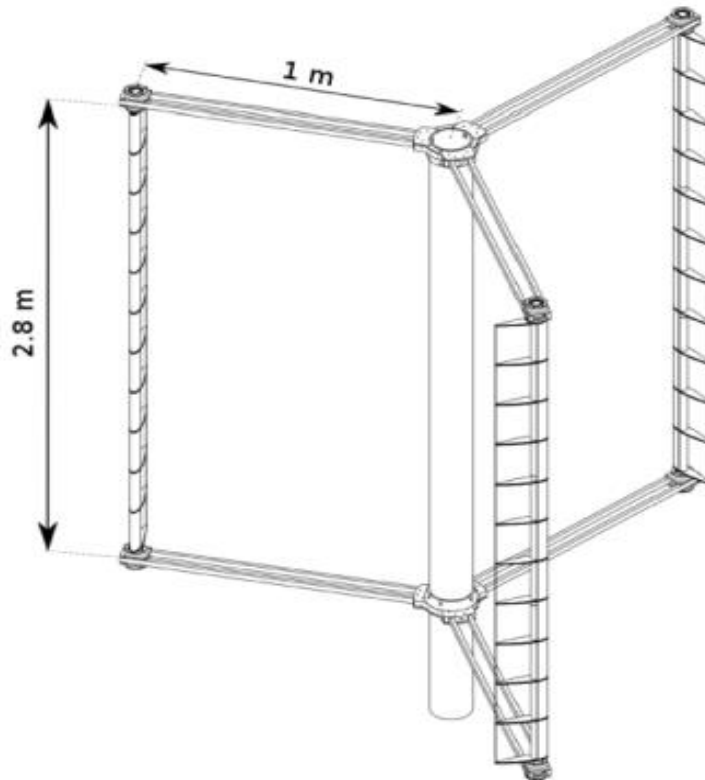


Figure 2.20 – Variable-pitch H-Darrieus turbine [47].

A definitive solution which would solve all the previously mentioned issues concerns the adoption of a variable pitch technology, showed in Figure 2.20. Through this configuration, exploiting a proper active or passive control of the blade orientation, it would be possible to reduce the angle of attack variation, achieving high efficiencies, high starting torques, sufficient stall resistance and reduced vibration problems. As reported in [42] though, the main problems related to this technology are firstly the severe mechanical complexity of a hypothetical active control system, and also the dubious design concepts linked to a passive control of the pitch, which would be driven by the complex interaction of fluid dynamic, inertial and other forces. Preliminary studies of this peculiar design were carried out exploiting the blade element momentum (BEM) method, but the results seemed to show that this configuration does not provide any

benefit on the efficiencies as previously expected. Due to these problems, this configuration has not been implemented commercially yet, and further studies must be carried out in order to get a proper validation, both from the economic and technological point of view.

For the sake of completeness, a critical comment to the future development of this class of machines is proposed by Kirke B. in [46]. In the paper, it was pointed out that velocities in the range of 2 – 3 m/s, for which vertical and horizontal axis turbines are usually designed, are virtually unheard in rivers, except where rapids would make hydrokinetic turbines impracticable, due to low depths and high fluctuation problems. This would cause poor consistency in the torque produced by the machine, and therefore in the electrical energy production. Depths of 1 to 2 meters are reasonably common in rivers, but most of the small axial flow and vertical axis turbines released on the market, designed to deliver the maximum power in the previously mentioned velocity range, would produce far less than their rated power; moreover, in case of an upscaled design required in order to deliver more power, these turbines would require a greater depth. Conclusively, Kirke confirms that the power output achievable by these systems would still be an improvement in remote off-grid villages, but, considering the work prices, transport and installation costs, the overall cost would not be worth the little power produced, and it would not be economically beneficial for both suppliers and customers. Through reliable analysis tools and properly built evaluation criteria, this thesis work aims than at providing an accurate overview of what is realistically obtainable in terms of energy extraction and efficiency levels, in order to predict whether these devices justify their hypothetical industrialization.

3 Computational Fluid Dynamics

3.1 The conservation laws

Computational fluid dynamics builds all its foundations on conservation laws. The definition of conservation law states that “*the variation of the total amount of a quantity inside a given domain is equal to the balance between the amount of that quantity entering and leaving the considered domain, plus the contribution from eventual sources generating that quantity*” [52]. Such a definition can be applied to all intensive physical properties of a fluid, whether those are scalar, vectorial or tensorial. In order to derive the conservation laws of fluid-dynamics then, the flux concept is fundamental. A flux is a vectorial quantity which describes the transfer of a given quantity through a surface; in particular, the flux perpendicular component is the one responsible for the passage. Considering the closed control volume Ω represented in Figure 3.1 and a generic scalar fluid property ϕ , it is possible to recognize a flux \vec{F} , a volume source Q_v and a surface source \vec{Q}_s , with $d\vec{S}$ being an infinitesimal oriented portion of the surface.

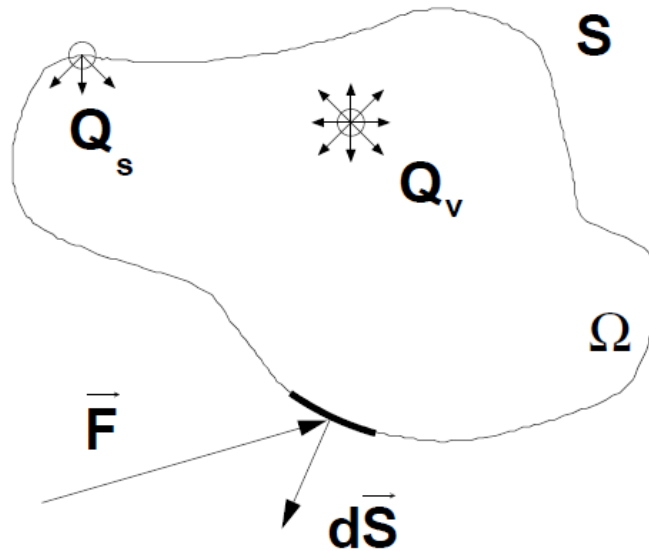


Figure 3.1 – Control volume.

The contribution of the flux integrated along the whole surface is given by:

$$\vec{F} \cdot d\vec{S} = F_n dS \quad \rightarrow \quad - \int_S \vec{F} \cdot d\vec{S} \quad (3.1)$$

A minus sign was introduced due to the fact that, by convention, the flux is considered positive if entering the surface. The total contribution of the sources is:

$$\int_S \vec{Q}_s \cdot \vec{dS} + \int_\Omega Q_v d\Omega \quad (3.2)$$

According to the definition, summing up all the terms and applying the Gauss's theorem, the general conservation equation results:

$$\frac{\partial}{\partial t} \int_\Omega \phi d\Omega = - \int_\Omega (\nabla \cdot \vec{F}) d\Omega + \int_\Omega (\nabla \cdot \vec{Q}_s) d\Omega + \int_\Omega Q_v d\Omega \quad (3.3)$$

where the first term represents the temporal variation of the considered fluid quantity ϕ . In differential form the relation becomes.

$$\frac{\partial \phi}{\partial t} + \nabla \cdot \vec{F} = Q_v + \nabla \cdot \vec{Q}_s \quad (3.4)$$

It is now important to differentiate between two types of fluxes:

- Convective flux
- Diffusive flux

The first one represents the amount of a given fluid quantity transported directly by the flow motion (advection), whilst the second accounts for the fluid quantity moved by its gradient. It is hence possible to introduce a new definition of the previous equation, which represents the general formulation of a transport equation for an arbitrary scalar quantity in differential form:

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \vec{V}) = \nabla \cdot (k \nabla \phi) + Q_v + \nabla \cdot \vec{Q}_s \quad (3.5)$$

The second term on the left-hand side of the equation represents the convective flux, where \vec{V} is the velocity. Shifting the attention to the right-hand side, the diffusive flux is represented by the first term, being k a diffusion coefficient. As previously mentioned, conservation laws are applicable to any intensive property, and the equation form will not change depending on the nature of the considered quantity (scalar, vectorial or tensorial).

Considering now a viscous fluid flow, its dynamics are completely described by five conservation equations:

- Mass conservation equation (continuity)
- Momentum conservation equations (Newton's law)
- Energy conservation equation

In addition, for compressible flows also the equation of state for ideal gases is used to relate density ρ , pressure P and temperature T .

The continuity equation for mass is defined as:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0 \quad (3.6)$$

It must be noted that sources terms are absent and only the convective flux is present, as mass does not diffuse.

The three momentum conservation equations are:

$$\frac{\partial \rho \vec{V}}{\partial t} + \nabla \cdot (\rho \vec{V} \vec{V} + P \vec{I} - \vec{\bar{\tau}}) = \rho \vec{f}_e \quad (3.7)$$

As mass, momentum does not diffuse, while contributions to the source terms come from external (e.g. gravity, applied forces) and internal (i.e. stresses) forces. Moreover, internal stresses are represented by two different terms; the isotropic component, represented by the pressure contribution and the anisotropic one, modelled through the viscous shear stress tensor $\vec{\bar{\tau}}$:

$$\vec{\bar{\tau}} = \mu(\nabla \vec{V} + \nabla \vec{V}^T) - \frac{2}{3}\mu(\nabla \cdot \vec{V})\vec{I} \quad (3.7)$$

The energy equation accounts for the total energy of the fluid, given by the sum of kinetic and internal energy contributions. In this case, also a diffusion term is present within the transport equation due to the temperature gradient (i.e. heat transfer). Being k the thermal conductivity:

$$E = e + \frac{1}{2}\vec{V}^2 \quad (3.8)$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho H \vec{V} - \vec{\bar{\tau}} \cdot \vec{V} - k \nabla T) = \rho \vec{f}_e \cdot \vec{V} + q_H \quad (3.9)$$

Volume sources are represented by the work of external forces and heat sources, whilst the surface sources are the result of the work done by the internal forces given by the pressure and the viscous shear stress.

When applied to a viscous case, the set of these five conservation equations takes the name of Navier-Stokes (NS) equations.

CFD takes advantage of numerical analysis in order to solve NS and other transport equations, used to predict a fluid flow behaviour. The most common technique employed by CFD is the Finite Volume Method (FVM): it consists of subdividing the solution domain into a finite number of contiguous control volumes (cells), to which the conservation equations in integral form are applied and approximated through Taylor's expansions. At the centroid of each cell lies a computational node P , where the unknowns are evaluated. The results of this process are algebraic equations for each control volume, where several neighbouring nodal values, denoted by the N subscript, appear due to

interpolation. The equations are then collected into a matrix and solved through proper algorithms:

$$a_P \phi_P + \sum_N a_N \phi_N = R \quad \rightarrow \quad [\mathbf{A}][\phi] = [\mathbf{Q}] \quad (3.10)$$

It is important to consider that transport equations accounting for turbulence are solved as well, but this topic will be covered extensively in the next section.

The accuracy of the final result depends mainly on two factors: firstly, the quality of the adopted mesh, in terms of refinement level and cell shape; secondly, on the adopted numerical schemes, in terms of interpolation and resolution of the matrix representing the whole system. The higher the order of the Taylor's expansion used to approximate the algebraic equation, the more precise will be the results; nevertheless, higher order schemes are also more prone to spurious oscillations, leading to unphysical modelling of the fluid properties and critical instabilities, which can cause divergence. For this work, second order schemes have been adopted, a structured hexahedral mesh has been developed and all the scalar and vectorial quantities are evaluated at the cell centroid, considering their fluxes passing through the surfaces (i.e. co-located grid).

3.2 Turbulence Modelling

As part of our daily life, turbulence is a three-dimensional, unsteady, rotational state of fluid motion with broad-banded fluctuations of flow quantities, such as pressure and velocity, occurring in both time and space [53]. It is interesting to notice that this phenomenon presents a dual nature: on one hand it represents a dissipative process, whilst on the other, if compared to a laminar flow, it improves the mixing and transport processes of the fluid properties. A turbulent flow is characterized by a specific non-dimensional parameter, known as the Reynolds number, which represents the relevance of the inertial forces with respect to the viscous ones. It is defined as:

$$Re = \frac{DV}{\nu}$$

where D is a characteristic dimension of the considered problem, V the flow velocity and ν the kinematic viscosity of the fluid. Another characteristic, both important and at the same time challenging, would be its chaotic nature, as it introduces a relevant issue: it is necessary to solve the consistency between the turbulent flow condition and the deterministic nature of the Navier–Stokes (NS) equations regulating the fluid dynamics. In fact, if for laminar flows it is possible to achieve a very good correspondence between theory and experiments, the same thing cannot be observed for turbulent cases, due to their high unsteadiness and randomness. In order to solve this problem, Reynolds (1894) introduced a

decomposition of the unsteady fluid properties into mean and fluctuating components, based on a time averaging process. Considering the generic fluid quantity φ , the decomposition would work as follow:

$$\varphi = \overline{\varphi(x)} + \varphi'; \quad \overline{\varphi(x)} = \frac{1}{\Delta T} \int_{t-\Delta T/2}^{t+\Delta T/2} \varphi(x, t) dt$$

$$\overline{\varphi + \phi} = \overline{\varphi} + \overline{\phi}; \quad \overline{\lambda\varphi} = \lambda\overline{\varphi}; \quad \overline{\frac{\partial\varphi}{\partial\alpha}} = \frac{\partial\overline{\varphi}}{\partial\alpha};$$

$$\overline{\int \varphi d\alpha} = \int \overline{\varphi} d\alpha; \quad \text{where } \lambda = \text{scalar and } \alpha = t, x$$

$$\overline{\varphi'} = 0; \quad \overline{\varphi \cdot \varphi'} = \overline{\varphi} \cdot \overline{\varphi'} = 0; \quad \overline{\varphi' \phi'} \neq 0; \quad \overline{\varphi'^m} \neq 0$$

Therefore, defining the Reynolds decomposition for the pressure P and the velocity \vec{V} and substituting these new variables into the NS set of equations, it is possible to obtain the so-called Reynolds Averaged Navier-Stokes equations (RANS):

$$P = \overline{P} + p'; \quad \vec{V} = \overline{\vec{V}} + \vec{v}$$

$$\frac{\partial\rho}{\partial t} + \nabla \cdot (\rho\overline{\vec{V}}) = 0$$

$$\frac{\partial(\rho\overline{\vec{V}})}{\partial t} + \nabla \cdot (\rho\overline{\vec{V}} \otimes \overline{\vec{V}}) = \rho\vec{g} - \nabla\overline{P} + \mu\nabla^2\overline{\vec{V}} + \frac{1}{3}\mu\nabla(\nabla \cdot \overline{\vec{V}}) - \nabla \cdot (\rho\overline{\vec{v}} \otimes \overline{\vec{v}})$$

In general, dealing with the NS equations, the problem consists of a system of four equations in four unknowns, where one would be for the pressure and three for the velocity. In the previously discussed case though, due to the Reynolds decomposition and subsequent substitution, another term was introduced. This parameter, appearing in divergence form and moved to the right hand side of the RANS equations, takes the name of Reynolds stress tensor and it derives from the dyadic product in the convective term within the momentum equation; due to the presence of this term, RANS equations are unclosed and hence require the implementation of a model in order to close the problem. Most of the RANS turbulence models assume a linear relationship between the Reynolds stress tensor and the mean strain rate tensor; moreover, being the Reynolds stress tensor symmetric, it is possible to decompose it into the sum of an isotropic and a deviatoric anisotropic component. In order to properly model the deviatoric component, in 1977 Boussinesq proposed a purely formal analogy with Newton's law concerning the relationship between stress and strain rate, developing the so called Boussinesq's hypothesis.

$$\begin{aligned}\bar{\bar{r}} &= -\rho \overline{\vec{v} \otimes \vec{v}} = -\rho \begin{pmatrix} \overline{u'^2} & \overline{u'v'} & \overline{u'w'} \\ \overline{v'u'} & \overline{v'^2} & \overline{v'w'} \\ \overline{w'u'} & \overline{w'v'} & \overline{w'^2} \end{pmatrix}; \quad \vec{v} = \vec{v}(u', v', w') \\ \bar{\bar{r}} &= -\rho \frac{2}{3} k * \bar{I} + \bar{\bar{a}}; \quad k = \frac{1}{2} (\overline{u'^2} + \overline{v'^2} + \overline{w'^2}) \\ \bar{\bar{a}} &= \bar{\bar{r}} + \rho \frac{2}{3} k * \bar{I} = -\mu_T (\nabla \vec{V} + \nabla \vec{V}^T) = -2\mu_T \bar{\bar{D}};\end{aligned}$$

The parameter k represents the average fluctuating turbulent kinetic energy (TKE), while μ_T regulates the linear relationship between the Reynolds stress tensor and the mean strain rate tensor; this term is called turbulent viscosity or eddy viscosity, and every relevant effect on the velocity field is modelled through its value. It is worth noticing that many nowadays adopted turbulence models are based on the definition of transport equations introduced just to properly predict μ_T . From a numerical point of view, it is possible to take advantage of other analyses (e.g. LES, DES, DNS) or other models (e.g. RSM, NEV, ARSM) which are able to better predict various vortexes scales in different conditions, but this goes beyond the purpose of this work.

For the purpose of this thesis, the starting point was a first rough evaluation of the Reynolds number, in order to better define the nature of the flow passing through the turbine. Opposingly to the standard procedure, the machine diameter was not considered as the characteristic dimension of the problem, but instead the turbine blade chord was selected as proper parameter to adopt. Considering a flow in a pipe, from theory it is well known that the Reynolds number thresholds for defining the state of motion are the ones reported in Table 3.1.

Re	Flow behaviour
<i>< 2000</i>	<i>Laminar</i>
<i>2000 ÷ 4000</i>	<i>Transitional</i>
<i>> 4000</i>	<i>Turbulent</i>

Table 3.1 – Re effects on flow behaviour in a pipe.

These values though are appropriate for that case, which though does not fit completely the topic covered in this thesis. Considering the machine, the turbine blades are the most critical components devoted to the forces and torque exchange with the fluid, and it is near these regions where it is required a good estimation of the Re. With these considerations, it is possible to link the case to the one concerning a plane lapped by a flow, and for this situation the threshold

related to a fully turbulent flow is set at $Re \approx 2 \times 10^5$. Evaluating the Re for a BSR = 1 and considering the values in Table 3.2, the final result of $Re \approx 6400$ was obtained, which is way below the turbulent threshold.

Chord C [m]	Kinematic viscosity ν [m²/s]	Flow velocity V [m/s]
0.0254	10^{-6}	0.25271

Table 3.2 – Parameters adopted for the Re evaluation.

The previous result though could mislead into a laminar classification of the flow, which would be an over simplification of the problem: in fact, it would not be completely appropriate to approximate a turbine blade as a plane lapped by a flow, as turbulent structures would be surely generated due to the flow detachment on suction side and in proximity of the leading and trailing edges. Therefore, even if the estimated Re was not high enough to consider the flow as fully turbulent, it would also be inappropriate to consider it laminar. Thanks to these considerations, a transitional turbulence model was chosen to be implemented, as already done by Padricelli [39] and Doan, Alayeto, Kumazawa and Obi [49]. The transitional eddy-viscosity chosen model was the Walters, Cokljat [54] $k_T - k_L - \omega$, which will be the reference model for the whole work. Nevertheless, concerning the single turbine analysis, also simulations implementing the Menter [55] $k - \omega$ SST eddy-viscosity model have been carried out, with the purpose of confirming the previous theoretical considerations.

3.2.1 The $k_T - k_L - \omega$ eddy-viscosity transitional model

According to the literature, transitional flows can be modelled following two different approaches: the first one consists of combining fully turbulent models with empirical transition correlations available from experimental databases, but these methods are of difficult use for complex three-dimensional flows [54]; the second approach consists of an additional transport equation to the current turbulence model, in order to take into account the transitional effects. The main difficulty related to these phenomenological models is related to the lack of knowledge about transitional flows, that even today represents an active research field. Nevertheless, most recent investigations helped underlining the most relevant scaling mechanisms able to better describe these types of streams, if compared to the empirical models [54].

The $k_T - k_L - \omega$ eddy-viscosity model developed by Walters and Cokljat [54] represents one of the possible transitional models available, and it is selected as the most fitting for the purposes of this work. It holds its basis on the $k - \omega$ eddy-

viscosity model, first developed by Wilcox [56] and based on two transport equations. Those are able to evaluate the turbulent kinetic energy k and the inverse time scale ω (i.e. specific dissipation frequency), which are required to correctly compute the eddy viscosity.

Considering the physical phenomenon of transition, the pre-transitional boundary layer is laminar in terms of the mean velocity profile; however, if the freestream turbulence intensity Tu_∞ increases, it is deformed accordingly and it is subject to a momentum increase in the inner region and a decrease in the outer region. Meanwhile, there is also a noticeable development of relatively high fluctuations along the streamwise direction, which could reach severe intensities when compared to the freestream [57]. Due to this process, the increment of frictions and the heat transfer, it is possible to obtain a breakdown of the streamwise fluctuations, a phenomenon which is called bypass transition. Now, Mayle and Schulz [58] assumed that the streamwise fluctuations, effectively representing the so called Klebanoff modes [57], were not classifiable as proper turbulence; hence, they introduced the laminar kinetic energy (LKE) concept k_L , in order to describe the process leading to bypass transition. Literally, they proposed the use of an additional kinetic energy equation. Even if the dynamics of the laminar kinetic energy are not completely clear, two critical aspects can be named: firstly, the selectivity of the boundary layer to certain freestream eddy scales, and secondly the amplification of low-frequency disturbances in the boundary layer by the mean shear. Moreover, the frequency content of k_L was found to be relatively independent from the forcing spectrum and its growth dynamics manifested as universal. The model adopted in the present thesis exploits then a transition initiation concept, based on the shear-sheltering and on the consideration of relevant time-scales for non-linear disturbance amplification and dissipation [54]; this leads to a more accurate prediction of the freestream turbulence length scale effect on the transition process. The term “shear-sheltering” refers to the damping of turbulence dynamics occurring in thin regions of high vorticity [59] and it inhibits non-linear turbulence breakdown mechanisms (e.g. occurring in pre-transitional boundary layers). From a practical point of view, once the transition starts, the shear-sheltering effects are limited to the viscous sublayer region, and transition could be interpreted as the growth of the pressure-strain terms in the Reynolds stresses equations. This is represented by an energy transfer from LKE, which models the Klebanoff modes, to the TKE, which models the three-dimensional fluctuations characteristic of a fully turbulent flow. Since the total fluctuation energy in the model is comprehensive of the sum of k_L and k_T , the energy transfer from one term to the other is interpreted as an energy redistribution (i.e. via the pressure-strain mechanism) rather than production (i.e. interaction with the mean flow) or dissipation (i.e. due to viscous mechanisms) [54]. Finally, shear-sheltering is incorporated through a damping production term, while transition initiation is included

through transfer terms in the k_L and k_T equations (i.e. ratio between the turbulent production time-scale and the molecular diffusion time-scale) [54].

3.2.2 The equations

The three transport equations for k_T , k_L and ω are:

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

$$\frac{Dk_L}{Dt} = P_{k_L} - R_{BP} - R_{NAT} - D_T + \frac{\partial}{\partial x_j} \left(\nu \frac{\partial k_L}{\partial x_j} \right)$$

$$\begin{aligned} \frac{D\omega}{Dt} = & C_{\omega 1} \frac{\omega}{k_T} P_{k_T} + \left(\frac{C_{\omega R}}{f_W} - 1 \right) \frac{\omega}{k_T} (R_{BP} + R_{NAT}) - C_{\omega 2} \omega^2 f_W^2 + C_{\omega 3} f_\omega \alpha_T f_W^2 \frac{\sqrt{k_T}}{d^3} \\ & + \frac{\partial}{\partial x_j} \left[\left(\nu + \frac{\alpha_T}{\sigma_\omega} \right) \frac{\partial \omega}{\partial x_j} \right] \end{aligned}$$

According to [60], a typographical correction was applied to the third term on the right-hand side of the ω equation with respect to the one reported in [54].

The production of TKE and LKE by mean strain is modelled as:

$$P_{k_T} = \nu_{T,s} S^2$$

$$P_{k_L} = \nu_{T,l} S^2$$

The ‘‘small-scale’’ eddy-viscosity is defined as:

$$\nu_{T,s} = f_\nu f_{INT} C_\mu \sqrt{k_{T,s}} \lambda_{eff}$$

where $k_{T,s}$ is the effective small-scale turbulence:

$$k_{T,s} = f_{SS} f_W k_T$$

where the kinematic wall effect is modelled through an effective turbulence length scale λ_{eff} and a damping function f_W , corrected according to [60]:

$$\lambda_{eff} = \min(C_\lambda d, \lambda_T)$$

$$\lambda_T = \frac{\sqrt{k_T}}{\omega}$$

$$f_W = \left(\frac{\lambda_{eff}}{\lambda_T} \right)^{\frac{2}{3}}$$

The viscous wall effect is incorporated exploiting a proper damping function and accounting for the effective turbulence Reynolds number:

$$f_v = 1 - \exp\left(-\frac{\sqrt{Re_T}}{A_v}\right)$$

$$Re_T = \frac{f_W^2 k_T}{\nu \omega}$$

The shear-sheltering effect is included in:

$$f_{SS} = \exp\left[-\left(\frac{C_{SS}\nu\Omega}{k_T}\right)^2\right]$$

while the turbulent viscosity coefficient C_μ is defined as:

$$C_\mu = \frac{1}{A_0 + A_s\left(\frac{S}{\omega}\right)}$$

The effect of turbulence intermittency is accounted as (also in this case, a correction was applied according to [60]):

$$f_{INT} = \min\left(\frac{k_T}{C_{INT}k_{TOT}}, 1\right)$$

The large-scale turbulent contribution is:

$$k_{T,l} = k_T - k_{T,s}$$

in addition, considering the definition for P_{k_L} :

$$\nu_{T,l} = \min\left\{f_{\tau,l}C_{l1}\left(\frac{\Omega\lambda_{eff}^2}{\nu}\right)\sqrt{k_{T,l}}\lambda_{eff} + \beta_{TS}C_{l2}Re_\Omega d^2\Omega, \frac{0.5 \cdot (k_L + k_{T,l})}{S}\right\}$$

where:

$$Re_\Omega = \frac{d^2\Omega}{\nu}$$

$$\beta_{TS} = 1 - \exp\left(-\frac{\max(Re_\Omega - C_{TS,crit}, 0)^2}{A_{TS}}\right)$$

$$f_{\tau,l} = 1 - \exp\left[-C_{\tau,l}\frac{k_{T,l}}{\lambda_{eff}^2\Omega^2}\right]$$

The anisotropic (near-wall) dissipation terms for k_T and k_L take the common form:

$$D_T = \nu \frac{\partial\sqrt{k_T}}{\partial x_j} \frac{\partial\sqrt{k_T}}{\partial x_j}$$

$$D_L = \nu \frac{\partial\sqrt{k_L}}{\partial x_j} \frac{\partial\sqrt{k_L}}{\partial x_j}$$

while the effective diffusivity and the boundary layer production term, including a proper damping function, are formulated as:

$$\alpha_T = f_\nu C_{\mu,std} \sqrt{k_{T,s}} \lambda_{eff}$$

$$f_\omega = 1 - \exp \left[-0.41 \cdot \left(\frac{\lambda_{eff}}{\lambda_T} \right)^4 \right]$$

The remaining terms in the transport equations are related to the laminar-to-turbulent transition mechanisms. Remaining constants are reported in Table 3.3.

$$R_{BP} = \frac{C_R \beta_{BP} k_L \omega}{f_W}; \quad R_{NAT} = C_{R,NAT} \beta_{NAT} k_L \Omega; \quad \beta_{BP} = 1 - \exp \left(-\frac{\phi_{BP}}{A_{BP}} \right);$$

$$\phi_{BP} = \max \left[\left(\frac{k_T}{\nu \Omega} - C_{BP,crit} \right), 0 \right]; \quad \beta_{NAT} = 1 - \exp \left(-\frac{\phi_{NAT}}{A_{NAT}} \right);$$

$$\phi_{NAT} = \max \left[\left(Re_\Omega - \frac{C_{NAT,crit}}{f_{NAT,crit}} \right), 0 \right]; \quad f_{NAT,crit} = 1 - \exp \left(-C_{NC} \frac{\sqrt{k_L d}}{\nu} \right)$$

$$\nu_T = \nu_{T,s} + \nu_{T,l}$$

$A_0 = 4.04$	$C_{INT} = 0.75$	$C_{\omega 2} = 0.92$
$A_s = 2.12$	$C_{TS,crit} = 1000$	$C_{\omega 3} = 0.3$
$A_\nu = 6.75$	$C_{R,NAT} = 0.02$	$C_{\omega R} = 1.5$
$A_{BP} = 0.6$	$C_{l1} = 3.4 \cdot 10^{-6}$	$C_\lambda = 2.495$
$A_{NAT} = 200$	$C_{l2} = 1.0 \cdot 10^{-10}$	$C_{\mu,std} = 0.09$
$A_{TS} = 200$	$C_R = 0.12$	$Pr_g = 0.85$
$C_{BP,crit} = 1.2$	$C_{SS} = 1.5$	$\sigma_k = 1$
$C_{NC} = 0.1$	$C_{\tau,l} = 4360$	$\sigma_\omega = 1.17$
$C_{NAT,crit} = 1250$	$C_{\omega 1} = 0.44$	

Table 3.3 – Model constants.

According to Walters and Lopez in [61], the previously described model is not always able to correctly predict the LKE production in regions far from the wall. In fact, it seems to be more accurate defining $\nu_{T,l}$ as:

$$\nu_{T,l} = \min \left\{ f_{\tau,l} C_{l1} \left(\frac{\Omega \lambda_{eff}^2}{\nu} \right) \sqrt{k_{T,l}} \lambda_{eff} + \beta_{TS} C_{l2} \left(\frac{d_{eff}^2 \Omega}{\nu} \right) d_{eff}^2 \Omega, \frac{0.5 \cdot (k_L + k_{T,l})}{S} \right\}$$

$$d_{eff} = \frac{\lambda_{eff}}{C_\lambda}$$

where all the other parameters are the same as the original formulation. As reported in [61] “*the new version of the model does not appear to affect the transition prediction behaviour of the original model for wall-bounded flows and shows no significant change for prediction of the mean velocity field in separated flow regions*”. However, it must be noted that for the purposes of this work, it was not possible to exploit this particular correction, as it has not been implemented in OpenFOAM yet, as well as in other commercial codes; further studies could be carried on trying to define a self-defined function within the current model and observing the possible differences.

3.2.3 The $k - \omega$ SST eddy-viscosity model

The Menter [55] two-equations $k - \omega$ SST eddy-viscosity model was first developed in 1994, and it is one of the most reliable solutions when dealing with fully turbulent flows, high detachment or recirculation along the boundary layer region. It is mainly based on two other eddy viscosity models, the Wilcox $k - \omega$ [56] and the Launder and Spalding $k - \varepsilon$ [62], and it takes the best of both, avoiding the use of wall functions and directly computing the fluid properties at the boundary layer. The model developed by Wilcox in 1988 is suitable for evaluation of the turbulent boundary layer in the sublayer and logarithmic regions, being it able to correctly predict flows affected by adverse pressure gradients and to directly solve for the turbulent quantities at the wall. As for the free-stream region, since the Wilcox model does not provide a sufficiently accurate solution, the 1974 Launder and Spalding model is considered instead. It exploits though the so called “wall functions”, i.e. it applies an empirical law of the wall and solves for the turbulent quantities sufficiently far from it. Despite economizing computational time and storage and adding empirical information when requested (e.g. roughness of the wall) [62], it does not provide the same robustness and precision in analysing the boundary layer as the $k - \omega$ does. Nevertheless, $k - \varepsilon$ does not suffer of high sensitivity to freestream values ω_f specified for ω outside the boundary layer, so it is preferable in this region when compared to the Wilcox model [56].

Despite mathematical analyses deployed large emphasis on the adverse pressure gradients effects on logarithmic region of boundary layers, Johnson and King [63] demonstrated that the eddy-viscosity in the wake region mainly influenced the prediction of adverse pressure gradients. This is further confirmed by the poor ability of the $k - \omega$ model of correctly evaluating pressure-induced separation, despite its major logarithmic region prevision characteristics.

Johnson and King founded their work on Bradshaw's considerations, which concluded that the principal turbulent shear stress is proportional to the turbulent kinetic energy in the wake region of the boundary layer; this assumption is actually violated by the Wilcox model [56]. With this considerations, the work developed by Menter takes the original $k - \varepsilon$ eddy-viscosity model and transforms it into a $k - \omega$ formulation (being the dissipation ε the inverse of ω); this leads to the addition of a cross-diffusion term in the ω equation multiplied by a blending function $(1 - F_1)$, which is able to switch between the original $k - \omega$ (i.e. adopted in sublayer and logarithmic regions) and $k - \varepsilon$ (i.e. adopted in the wake region). This first step represents the definition of the so called BSL baseline model; the second important step is related to the modification of the eddy-viscosity definition through the function F_2 , in order to account for the principal turbulent shear stress transport (SST) and at the same time try to satisfy the Bradshaw's observation, without developing a complex Reynolds stress model (RSM).

3.2.4 The equations

The BSL model is defined as follows:

$$\frac{D\rho k}{Dt} = \tau_{ij} \frac{\partial u_i}{\partial x_j} - \beta^* \rho \omega k + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_k \mu_t) \frac{\partial k}{\partial x_j} \right]$$

$$\frac{D\rho \omega}{Dt} = \frac{\gamma}{\nu_t} \tau_{ij} \frac{\partial u_i}{\partial x_j} - \beta \rho \omega^2 + \frac{\partial}{\partial x_j} \left[(\mu + \sigma_\omega \mu_t) \frac{\partial \omega}{\partial x_j} \right] + 2(1 - F_1) \rho \sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}$$

The constants ϕ of the model are computed as:

$$\phi = F_1 \phi_1 + (1 - F_1) \phi_2$$

where the constants of set 1 (Wilcox) are defined as:

$$\sigma_{k1} = 0.5; \quad \sigma_{\omega 1} = 0.5; \quad \beta_1 = 0.0750$$

$$\beta^* = 0.09; \quad \kappa = 0.41; \quad \gamma_1 = \frac{\beta_1}{\beta^*} - \frac{\sigma_{\omega 1} \kappa^2}{\sqrt{\beta^*}}$$

and the constants of set 2 (standard $k - \varepsilon$) are:

$$\sigma_{k2} = 1; \quad \sigma_{\omega 2} = 0.856; \quad \beta_2 = 0.0828$$

$$\beta^* = 0.09; \quad \kappa = 0.41; \quad \gamma_2 = \frac{\beta_2}{\beta^*} - \frac{\sigma_{\omega 2} \kappa^2}{\sqrt{\beta^*}}$$

Moreover, it is possible to consider the following definitions:

$$v_t = \frac{k}{\omega}; \quad \tau_{ij} = \mu_t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} - \frac{2}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right) - \frac{2}{3} \rho k \delta_{ij}; \quad F_1 = \tanh(\arg_1^4)$$

$$\arg_1 = \min \left[\max \left(\frac{\sqrt{k}}{0.09\omega y}; \frac{500\nu}{y^2\omega} \right); \frac{4\rho\sigma_{\omega 2}k}{CD_{k\omega}y^2} \right]$$

$$CD_{k\omega} = \max \left(2\rho\sigma_{\omega 2} \frac{1}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}, 10^{-20} \right)$$

where y is the distance from the next surface and $CD_{k\omega}$ is the positive portion of the cross-diffusion term which appeared in the ω transport equation.

The SST model appears identical to the previous formulation, except for the constants ϕ_1 that switch to:

$$\sigma_{k1} = 0.85; \quad \sigma_{\omega 1} = 0.5; \quad \beta_1 = 0.0750; \quad a_1 = 0.31$$

$$\beta^* = 0.09; \quad \kappa = 0.41; \quad \gamma_1 = \frac{\beta_1}{\beta^*} - \frac{\sigma_{\omega 1}\kappa^2}{\sqrt{\beta^*}}$$

The eddy-viscosity is defined as:

$$v_t = \frac{a_1 k}{\max(a_1 \omega; \Omega F_2)}$$

where Ω is the absolute value of the vorticity and F_2 is defined as:

$$F_2 = \tanh(\arg_2^2)$$

$$\arg_2 = \max \left(2 \frac{\sqrt{k}}{0.09\omega y}; \frac{500\nu}{y^2\omega} \right)$$

3.3 CFD solver algorithms

Once all the terms appearing in the conservation equations are modelled and assembled in a matrix, being the resulting system implicit, a solver algorithm is required. As said, the problem is closed and it consists of four unknowns in four equations for each control volume; in fact, considering incompressible fluids, the energy equation is not resolved, as those are not subject to both valuable internal energy changes and temperature variations. Considering those equations, it is possible to notice that pressure appears in all three momentum equations, and at the same time the velocity field must satisfy the continuity equation; the main problem is related to the fact that, despite the system being closed, an explicit equation for the pressure is not obtainable. This is due to the incompressible nature of the flow, which prevents the implementation of an equation of state

linking density and pressure (e.g. ideal gases). Actually, in this case the continuity equation works more as a constraint rather than a real equation to solve.

One of the most adopted solutions in this case is exploiting a pressure-velocity coupling algorithm, which derives a pressure equation from the NS equations. In general, three different algorithms descend from this approach:

- SIMPLE (Semi-Implicit Method for Pressure-Linked Equations)
- PISO (Pressure-Implicit with Splitting of Operators)
- PIMPLE (merged PISO-SIMPLE algorithm)

Starting from the coefficient matrix \mathcal{M} obtained from the discretization of the NS equations, the mentioned algorithms operate guessing a value for pressure and then evaluating the velocity field, which will not satisfy the continuity equation. This phase is called momentum predictor:

$$\mathcal{M}\vec{V} = -\nabla P \quad (3.11)$$

Then, the key step is extracting the diagonal component of \mathcal{M} , denoted as \mathcal{A} , and another matrix \mathcal{H} representing the residuals. Manipulating the previous expression, it becomes:

$$\mathcal{M}\vec{V} = \mathcal{A}\vec{V} - \mathcal{H} = -\nabla P \quad (3.12)$$

Next, they derive an expression for the velocity to substitute in the continuity equation, in order to evaluate the corrected pressure field; this is the so-called pressure correction step (3.14).

$$\vec{V} = \mathcal{A}^{-1}\mathcal{H} - \mathcal{A}^{-1}\nabla P \quad (3.13)$$

$$\nabla \cdot (\mathcal{A}^{-1}\nabla P) = \nabla \cdot (\mathcal{A}^{-1}\mathcal{H}) \quad (3.14)$$

Finally, the velocity expression (3.13) is reconsidered in order to explicitly compute the corrected velocity field. However, as \mathcal{H} depends on the previously guessed value \vec{V} , the solution obtained at last cannot be correct. The SIMPLE algorithm solves this problem by starting the so-called “outer corrector” loop, repeating all the procedure from the momentum predictor phase (3.11) and using the latest velocity and pressure values until convergence is reached. This algorithm though is conceived for steady state flows: in fact, it exploits under-relaxation factors to artificially increase the diagonal dominance of \mathcal{M} against the non-linear terms, consequently dumping any unsteady fluctuation. Hence, it would not be suitable for this work.

It is now mandatory to define the necessary condition for CFD simulation stability, known as CFL (Courant – Friedrichs – Lewy) condition, which finds its expression in the Courant number Co :

$$Co = \frac{u \cdot \Delta t}{\Delta x} < Co_{MAX} \quad (3.15)$$

The CFL condition states that the distance travelled through the mesh by any information, within a considered time step, must be lower than the mesh dimension times a Co_{MAX} factor. If Co_{MAX} has unitary value, this means that information starting from a given cell, within a given time step, must travel in the neighbouring cell only. Explicit solvers actually employ $Co_{MAX} = 1$, whilst implicit ones for different reasons generally use $Co_{MAX} > 1$.

Back to the algorithms, PISO performs the momentum predictor stage once and takes advantage of a “inner corrector” loop, used only for the pressure correction step. In fact, as this algorithm solves transient cases, by imposing a Courant number $Co < 1$ the fine temporal discretization assures the diagonal dominance of \mathcal{M} against non-linear terms related to convection. It is hence possible to reach a partial convergence, maintaining stability in the analysis and using in general no more than two or three inner loops.

Finally, the PIMPLE algorithm merges together PISO and SIMPLE: it considers every time step as a steady state problem, exploiting outer corrector loops and relaxation factors in order to reach convergence. Once a satisfactory solution is obtained, it moves to the next time step. Due to its nature, it usually works with $Co > 1$ and hence with longer time steps, when compared to PISO.

Both PISO and PIMPLE algorithms were tested in this thesis, and a brief comparison will be provided in Section 5.2.

4 OpenFOAM

OpenFOAM (Open-source Field Operation And Manipulation) is a Linux based C++ toolbox, firstly developed in the 90s with the task of implementing numerical solvers. The target fields of this software in general concern the solution of continuum mechanics problems, and range from chemistry to solid dynamics and finance, most prominently including fluid dynamics.

The software applications are classified in two different categories [64]: *solvers*, which are specifically designed for different problems characterized by different assumptions, and *utilities*, which are designed to carry out tasks concerning data manipulation. OpenFOAM provides tools for the pre and post processing as well, but it is also supported by third party software such as paraView, which introduces a graphical interface.

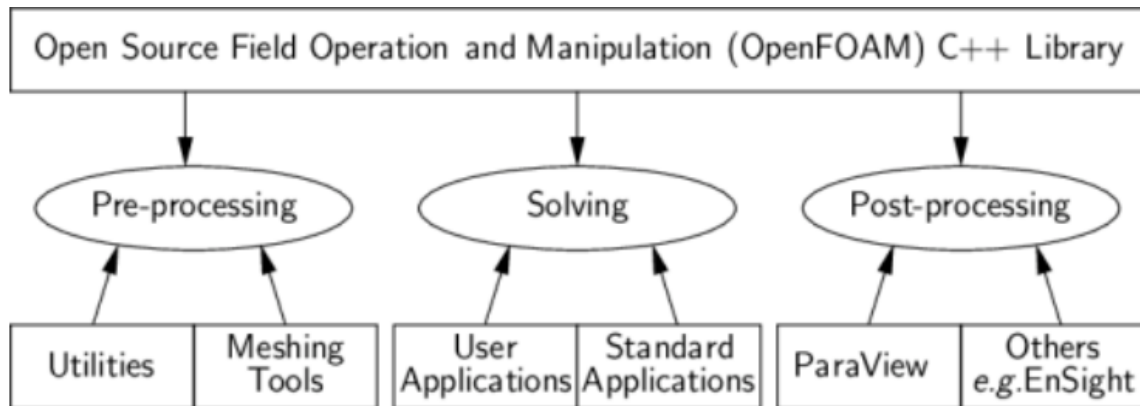


Figure 4.1 - Generic analysis structure in OpenFOAM

A typical OpenFOAM case folder contains a minimum number of specific files, organized in the *system* and *constant* directories, which are required in order to run any simulation; additionally, depending on the case task, accessory dictionaries can be implemented, in order to allow the correspondent utility to be executed. Time directories are a fundamental component of the analysis as well, since, according to the settings of the study, those are periodically created and the simulation results are stored there. A particular mention is due to the first time directory '0', which has to be manually created and includes all the initial and boundary conditions of the problem. Figure 4.2 schematically shows a generic OpenFOAM case folder.

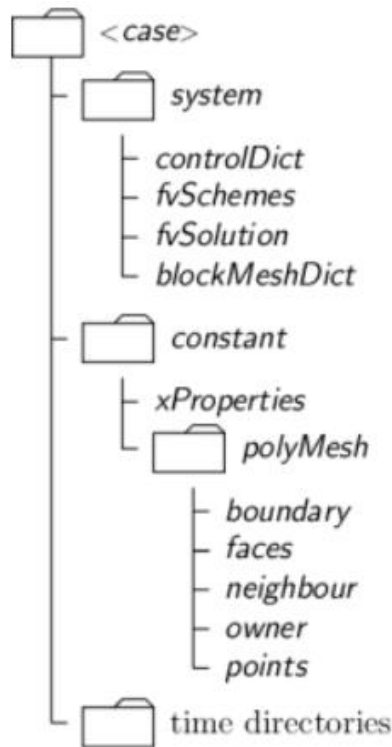


Figure 4.2 - Typical folder tree for OpenFOAM simulations

The *system* folder stores different important files:

- the fundamental *controlDict* dictionary, in which all the time parameters are specified, as well as all the input/output data formats are handled. Also, post-processing data generation is controlled by this dictionary.
- the *fvSchemes* file, where the numerical integration and interpolation methods used to solve the specific terms of PDE are specified.
- the *fvSolution* file, which is used to set and control the solver algorithm parameters, and in particular to set its tolerances.
- all the mesh-building tool dictionaries, such as *blockMeshDict*, which will be discussed in depth further ahead in this work.

The *constant* folder stores instead:

- the *polyMesh* folder, where the whole case-specific mesh is stored. This folder can be either created with OpenFOAM itself or can be written by third party software.
- files containing fundamental properties of the considered case. This thesis task required the implementation of a *transportProperties* file, describing the physical properties of the considered fluid, and a *turbulenceProperties* file, where the adopted turbulence model was specified.

- The dictionary *dynamicMeshDict*, required in order to define the feature motion; depending on the selected technique, this dictionary can be implemented very differently.

As previously mentioned, the first time directory ‘0’ stores the initial and boundary conditions for the whole problem, whether those concern volume scalar fields such as pressure (*volScalarField* in OpenFOAM) or volume vector field such as velocity (*volVectorField*). As any OpenFOAM domain composes of *patches*, a boundary condition for each of those is required and, depending on what the *patch* physically represents, it must be carefully selected. As for this topic, the software offers a wide choice of already built-in conditions which cover the great majority of possibilities; nevertheless, being OpenFOAM an open-source package, it is always possible to either implement a customized feature or modify an existing one. Of course, this holds for every OpenFOAM application type, ranging from boundary conditions to solvers and dictionaries. The chance of implementing a user defined boundary condition will be exploited further ahead in this thesis, according to the guide in [51].

Finally, one huge advantage offered by OpenFOAM is the chance of running cases in parallel: this expression refers to the capability of the software of splitting the computational effort on multiple processors at the same time, drastically reducing the time required for a simulation to be completed. This procedure requires a specific dictionary in the *system* folder, named *decomposeParDict*, in which the number of processors and the decomposition method are defined. Then, when running the application, a ‘*-parallel*’ option has to be specified, along with the number of used processors (i.e. ‘*mpirun -np 2 pimpleDyMFoam -parallel*’). At last, once the computation is over, the ‘*reconstructPar*’ command has to be run, in order to assemble the data coming from different processors and to make them readable for post-processing. This feature was extensively employed in this thesis, both for mesh building processes and simulations; for explicative purposes, moving from a 2 processors decomposition to a 16 processor decomposition reduced the computational time to about 1/7.

5 Case setup

Being this thesis inspired by [39] and [49], the most favourable approach to the topic was to critically consider the authors' choices and evaluate if those would properly fit the considered cases. Firstly, in the paper the authors presented the experimental setup required to evaluate some experimental quantities, such as the reference freestream velocity and the turbulence intensity. This setup was approximately the same as the one used in [48], except for the measurement devices, which in [48] were focused on capturing the torque power output of the turbine. The main core of the reference article addressed then the issues of the turbulence modelling and the meshing strategy. As for the turbulence modelling, the Spalart-Allmaras (one transport equation of ν_T), the $\kappa - \omega$ SST (two transport equations of κ and ω) and the $k_T - k_L - \omega$ model (three transport equations of k_T , k_L and ω) were the considered options. Regarding the meshing strategy, the authors, other than the classical rotating mesh technique, took into account different possibilities: a time deforming mesh, a separated overset mesh for each blade on top of a background mesh and finally a reproduction of the blades motion by moving immersed boundaries.

Both 2D and 3D simulations of the single turbine configuration were tested and, due to computational effort and convergence reasons, the final choice of the authors was the Spalart-Allmaras turbulence model and the classical rotating mesh strategy. However, one final consideration was made concerning the alternative meshing techniques: when dealing with a more geometrically complex case, i.e. two closely spaced counter-rotating turbines where the chosen technique would not be suitable, also the time deforming and the overset meshing techniques would represent solid alternatives, despite the additional computational requirements.

The purpose of the present thesis is investigating the reciprocal effects of the flow fields generated by two closely spaced H-Darrieus turbines. Firstly, in order to validate the model, a single turbine configuration is investigated and then, for scientific and research purposes, two different counter-rotating configurations are tested. All the CFD simulations are carried out using an OpenFOAM environment and, since the blade profile does not vary along the span, in a 2-dimensional domain. The obtained results are then representative of the machines behaviour approximately at mid-span, where the three-dimensional effects are negligible. As an objective of a future work, the 3D model of the turbines would be able to capture the effects of the finite length of the blades, the blockage effect given by the free surface and the riverbed and also the three-dimensional nature of the vortices, but, on the other hand, it would require much more computational power.

As presented in [48], the blades shape is based on the NACA0012 profile and the considered dimensions are the same as the ones presented in the article.

Geometrical and mechanical characteristics of the single turbine rotor are listed in Table 5.1.

Diameter [m]	<i>0.0683</i>
Chord [m]	<i>0.0254</i>
Pitch angle [°]	<i>15</i>
N. blades	<i>3</i>
Solidity	<i>1.1</i>

Table 5.1 - Single turbine fundamental characteristics

The blades STL files provided to carry out this analysis though, being a very accurate representation of the actual experimental model, presented a very sharp trailing edge, which classically introduces one of the biggest issues for the meshing procedure. The solution adopted in this thesis was cutting from the STL files 2% of the blade chord length from the very same trailing edge, in order to both increase the geometrical angle which the mesh had to accommodate, and to remove that portion of the blade which would be distorted by the meshing algorithm. Thanks to the fact that a separation would always occur in the trailing edge region, such a procedure concerned a dead water area, hence granting irrelevant effects on the simulations' solution. Moreover, as long as the cut portion would be negligible, the new alternative profile could be selected at will. Hence, since different meshing techniques were adopted in different configurations of the turbine, as will be discussed further ahead, the trailing edge blade profile chosen for the closely spaced counter rotating turbines is actually different from the one used in the single and the regular counter rotating configurations.

In order to be able to compare the results with the conclusions from [48] and [49], the selected computational domain for the single turbine configuration has the same dimensions of the channel used in laboratory. At the opposite, concerning the counter rotating setups, the choice was to keep the same blockage ratio as the previous case, hence accordingly increasing the domain width. Each study case is illustrated in its relative domain in Figure 5.1.

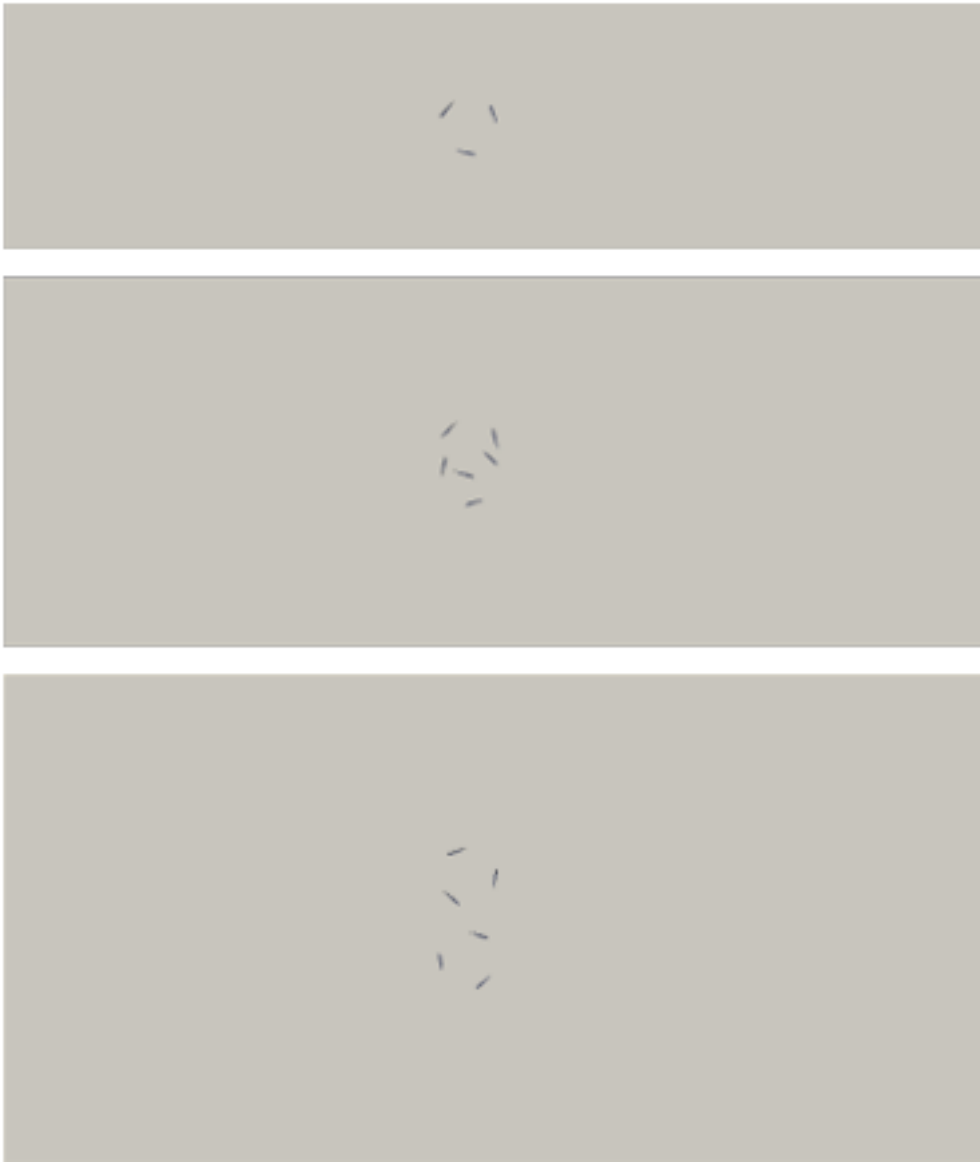


Figure 5.1 - Computational domains for the single turbine (a), engaging (b), non-engaging (c) configurations

The meshing procedure was different among the different study cases: regarding the single turbine and non-engaging turbines, in accordance with [49], the classical rotating mesh technique was selected; as for the engaging turbines instead, due to the complex motion of the blades, it was necessary to implement a time-deforming mesh. The details of these strategies will be provided further ahead.

As previously discussed in Section 3.2, despite the conclusions reached in [49], the investigated turbulence models are the two equations $\kappa - \omega$ SST and the three equations $k_T - k_L - \omega$ models. The Spalart-Allmaras model is in fact a very time-efficient solution, being it built on just one balance equation and being able to

work even with a coarse mesh, but it was mainly conceived for aerospace applications and, among its drawbacks, it gives relatively poor predictions of shear flows, separation and isotropic turbulence decay. Moreover, since the meshes were built with great precision especially along the blade profiles, more complex models would represent a better fit for the study cases. Actually, in [49] one of the main reasons why the $k_T - k_L - \omega$ model was discarded was the high computational effort required, also considering the mesh contribution; considering the two dimensional nature of the analysis and the computational resources available for this thesis, it was possible to aim for a more accurate solution, applying the two selected models to a fine mesh.

5.1 Domain general features and boundary conditions

The computational domain selected for this analysis is a close representation of the one used in [48] and [49], and the different configurations are displayed in Figure 5.1. In order to save some computational time and at the same time avoid the influence of domain boundaries in the streamwise direction, the channel length was reduced from the original $3.5m$ to $1.2m$, and the turbine centre was positioned in such a way to have about seven machine diameters upstream and about nine downstream. The asymmetry of the domain was chosen in order to provide enough physical space for the wake to completely develop. Considering the cross-stream direction, the operative choice was to keep the geometrical blockage ratio constant and equal to the experimental one for every setup, resulting then in a proportional increment of the channel width. The driving thought is that, given an existing channel with fixed dimensions, it is possible to select the most efficient turbine configuration to extract the flow energy. Finally, as the analysis was carried out in 2D, the dimension in z direction did not influence the results and it was arbitrarily set to $0.01m$.

The domain dimensions are presented in Table 5.2.

	Single Turbine	Engaging turbines	Non-engaging turbines
Length [m]	<i>1.2</i>	<i>1.2</i>	<i>1.2</i>
Width [m]	<i>0.3</i>	<i>0.45</i>	<i>0.6</i>
Blockage [%]	<i>22.77</i>	<i>22.77</i>	<i>22.77</i>
Thickness [m]	<i>0.01</i>	<i>0.01</i>	<i>0.01</i>

Table 5.2 - Domains dimensions

The OpenFOAM environment then identifies the domain boundaries as *patches*, which have to be defined and named at the very beginning of each study and where the boundary conditions are imposed. Since all the domains have rectangular shape, the following patches differ just in dimension among the different study cases, but always present the same name and boundary condition:

- in streamwise direction there are the *inlet* and *outlet* patches;
- in cross-stream direction there are the *top* and *down* patches;
- along the machine axis there are the *front* and *back* patches.

Additional *patches* are also defined for each blade in the studied configuration, and in case of the rotating mesh technique.

As for the boundary conditions, in order not to over-constrain the problem, at the *inlet* it was imposed a fixed value for the velocity and a zero-gradient condition in normal direction for the pressure, whilst at the *outlet* it was imposed a zero-gradient in normal direction for the velocity and a fixed value for the pressure. The *top* and *down patches* instead required an adherence condition concerning the velocity and a zero-gradient in normal direction for the pressure, simulating this way the actual behaviour of physical walls in the channel. A special mention is due to the *front* and *back patches*, which are actually used to mimic the 3D nature of the solid bodies: the boundary conditions for every considered quantity on those patches are set to *empty*, which signals to the OpenFOAM software that the analysis is two-dimensional, and hence all the calculated parameters are periodically repeated along the third dimension. Finally, the boundary conditions for velocity and pressure on the blades are respectively set to an adherence condition and a zero-gradient in normal direction.

Table 5.3 sums up the velocity and pressure boundary conditions for the presented patches. Notice that the velocity at the inlet depends on the considered flow regime and the pressure value is the relative to the atmospheric conditions.

Patch	Velocity [m/s]	Pressure [atm]
<i>inlet</i>	<i>0.25271</i>	<i>zeroGradient</i>
<i>outlet</i>	<i>zeroGradient</i>	<i>0</i>
<i>top</i>	<i>noSlip</i>	<i>zeroGradient</i>
<i>down</i>	<i>noSlip</i>	<i>zeroGradient</i>
<i>front</i>	<i>empty</i>	<i>empty</i>
<i>back</i>	<i>empty</i>	<i>empty</i>
<i>blade#</i>	<i>movingWallVelocity</i>	<i>zeroGradient</i>

Table 5.3 - Velocity and pressure boundary conditions

According to the selected turbulence model then, some additional boundary conditions have to be set for every turbulence parameter.

Concerning the two-equation fully turbulent model $\kappa - \omega$ SST, boundary conditions are required for the turbulent kinetic energy κ , the specific dissipation rate ω and the kinematic eddy viscosity ν_T . The proper conditions for ω are prescribed in [55] and the same values are used in every flow regime; the following equations were used in order to respectively compute the freestream value and the wall values:

$$\omega_\infty = \frac{11 U_\infty}{2 L} \quad (5.1)$$

$$\omega = 10 * \frac{6 * \nu}{\beta_1 * (\Delta y_1)^2} \quad (5.2)$$

Where U_∞ is the freestream velocity, L is the domain length, ν is the fluid viscosity, $\beta_1 = 0.075$ and Δy_1 is the distance of the first cell centre from the wall.

The turbulent kinetic energy values are set to 0 at the walls, a zero-gradient in normal direction is imposed at the *outlet*, while at the *inlet* κ is evaluated for each flow regime with the following formula:

$$\kappa_T = 3/2 * (I * U_\infty)^2 \quad (5.3)$$

Where the parameter I represents the turbulence intensity, which for this thesis was assumed to be 5%. The kinematic eddy viscosity is instead computed from the previous two, according to the definition given in [55]:

$$\nu_t = \frac{a_1 k}{\max(a_1 \omega; \Omega F_2)} \quad (5.4)$$

Table 5.4 reports the boundary conditions used in the $\kappa - \omega$ SST simulations.

Patch	κ	ω	ν_T
<i>inlet</i>	<i>0.0002395</i>	<i>1.16</i>	<i>calculated</i>
<i>outlet</i>	<i>zeroGradient</i>	<i>zeroGradient</i>	<i>calculated</i>
<i>top</i>	<i>0</i>	<i>250</i>	<i>calculated</i>
<i>down</i>	<i>0</i>	<i>250</i>	<i>calculated</i>
<i>front</i>	<i>empty</i>	<i>empty</i>	<i>empty</i>
<i>back</i>	<i>empty</i>	<i>empty</i>	<i>empty</i>
<i>blade#</i>	<i>0</i>	<i>8800000</i>	<i>calculated</i>

Table 5.4 – κ , ω and ν_T boundary conditions for the κ - ω SST model

Considering now the transitional model based on three balance equations $k_T - k_L - \omega$, boundary conditions are again required for the turbulent kinetic energy k_T , the laminar kinetic energy k_L , the specific dissipation rate ω and the kinematic eddy viscosity ν_T . In [54] the authors described the adequate boundary conditions to implement for the model to work correctly. The turbulent kinetic energy required the same set up as the previous model, with a specific value for the *inlet* patch for every flow regime. The laminar kinetic energy instead was set to 0 on every patch except for the *outlet*, where a zero-gradient condition was imposed in order not to over-constrain the problem. As for ω , the zero-gradient condition is required for every physical wall as well as for the *outlet*; the inlet value was instead computed according to

$$\omega = \sqrt{\frac{k_T}{C_\mu * 0.007 * D_h}} \quad (5.5)$$

where k_T is the previously computed turbulent kinetic energy, $C_\mu = 0.09$ and D_h is the equivalent hydraulic diameter.

Finally, similarly to the previous model, the kinematic eddy viscosity is computed from the previous parameters as:

$$\nu_T = \nu_{TS} + \nu_{TL} \quad (5.6)$$

Where ν_{TS} and ν_{TL} are respectively the kinematic eddy viscosity contribution from the small-scale and the large-scale turbulent eddies.

Table 5.5 reports the boundary conditions used in the $k_T - k_L - \omega$ simulations.

Patch	k_T	k_L	ω	ν_T
<i>inlet</i>	<i>0.0002395</i>	<i>0</i>	<i>2.662</i>	<i>calculated</i>
<i>outlet</i>	<i>zeroGradient</i>	<i>zeroGradient</i>	<i>zeroGradient</i>	<i>calculated</i>
<i>top</i>	<i>0</i>	<i>0</i>	<i>zeroGradient</i>	<i>calculated</i>
<i>down</i>	<i>0</i>	<i>0</i>	<i>zeroGradient</i>	<i>calculated</i>
<i>front</i>	<i>empty</i>	<i>empty</i>	<i>empty</i>	<i>empty</i>
<i>back</i>	<i>empty</i>	<i>empty</i>	<i>empty</i>	<i>empty</i>
<i>blade#</i>	<i>0</i>	<i>0</i>	<i>zeroGradient</i>	<i>calculated</i>

Table 5.5 - k_T , k_L , ω and ν_T boundary conditions for the $k_T - k_L - \omega$ model

5.2 Solver and numerical schemes

The OpenFOAM software provides different alternatives concerning the algorithm used to solve the RANS equations, and its choice is based on some macro-characteristics such as the compressibility and the steadiness of the investigated problem. The *PimpleDyMFoam* solver deals with incompressible and unsteady phenomena, which perfectly fit the purpose of this thesis.

Thanks to its various options, it was possible to use *PimpleDyMFoam* either with a PIMPLE or a PISO algorithm simply changing the number of iterations required to solve one time-step, and both implementations provided advantages. The PIMPLE algorithm is in general more stable, faster and it allows for much larger time-steps (higher Courant number), due to the fact that it solves each step multiple times using relaxation factors; on the other hand, the usage of a high Courant number implies the risk of information loss for a very fine mesh.

The PISO algorithm instead uses smaller time-steps (Courant number smaller than unity) and a single iteration for each of those, not requiring the definition of relaxation factors. The main advantage of this method is the certainty of not losing any flow information because of a badly calibrated time advancement.

Both implementations were tested for a reference flow regime and no significant difference was reported in either the flow fluctuations or the final evaluated power coefficient. Nevertheless, the PIMPLE algorithm consistently showed divergent behaviour using a relatively high Courant number: it stabilized for Co lower than 2. Table 5.6 shows the *PimpleDyMFoam* parameters and some significant data from the simulations:

Parameter	PIMPLE	PISO
<i>nOuterCorrectors</i>	1000	1
<i>nCorrectors</i>	1	3
<i>nNOCorrectors</i>	0	0
<i>Max Co</i>	2	0.5
ΔT	$\sim 1.5 \div 2 * 10^{-4}$	$\sim 6 \div 9 * 10^{-5}$

Table 5.6 - Solvers settings in OpenFOAM

Considering that the adjustable time-step option was used for the simulations, notice how the Courant number is defined through its maximum admissible value and, consequently, the time-step is not constant throughout the simulation. Moreover, it must be considered that the *nOuterCorrector* parameter represents the maximum admissible number of iterations per time-step, and not the

effective one; when converging, the algorithm required about 20-22 iterations per time-step.

As clearly emerges from this data, for the purpose of this thesis the PISO mode was the obvious choice, being more time-efficient than PIMPLE, mostly due to the poor time advancement improvement showed by the algorithm.

OpenFOAM requires then the definition of numerical schemes for each investigated quantity, in order to carry out the solution of the discretized PDE. Numerical schemes classify according to their order: first order schemes provide a stable and less accurate solution, whilst second order schemes give a much more accurate behaviour, despite their characteristic oscillations which result in general instability. Most of the times in fact, in order to improve stability, second order schemes either benefit from some corrective factors which help bounding the solution or are not used as pure second order schemes: a blending factor between a first and a second order scheme is hence defined, which helps reducing the oscillating behaviour of the solution, though also reducing its accuracy.

The numerical schemes selected for this thesis are equally used in each analysis; for accuracy's sake, second order schemes were preferred and no stability issues occurred.

Table 5.7 shows the complete list of the employed numerical schemes:

PDE term	Numerical scheme
<i>ddtSchemes</i>	<i>CrankNicolson 0.9</i>
<i>gradSchemes</i>	<i>Gauss linear</i>
<i>divSchemes</i>	<i>Gauss linear</i>
<i>laplacianSchemes</i>	<i>Gauss linear corrected</i>

Table 5.7 - Adopted numerical schemes

5.3 Mesh single turbine

The single turbine configuration, as said, represents the foundation of this thesis, as it was used to validate the numerical model. The computational domain is described in the previous Section and it is showed in Figure 5.1a.

The meshing procedure required the implementation of different OpenFOAM utilities, first of which was *blockMesh*. Referring to a cartesian coordinate system, this dictionary defines the domain dimensions, its boundary *patches* and, most importantly, the first raw spatial discretization made of hexahedral cells. Being the analysis two-dimensional, and hence the third dimension irrelevant, the background mesh was designed with squared-section cells and, according to the analysis carried out in [39], the side dimension was selected to be *6mm*. Table 5.8 shows the dimensions and the raw discretization of the domain, while Figure 5.2 represents its discretization.

Direction	Dimension [m]	N. of cells
<i>x</i>	<i>1.2</i>	<i>200</i>
<i>y</i>	<i>0.3</i>	<i>50</i>
<i>z</i>	<i>0.01</i>	<i>1</i>

Table 5.8 - Single turbine domain discretization

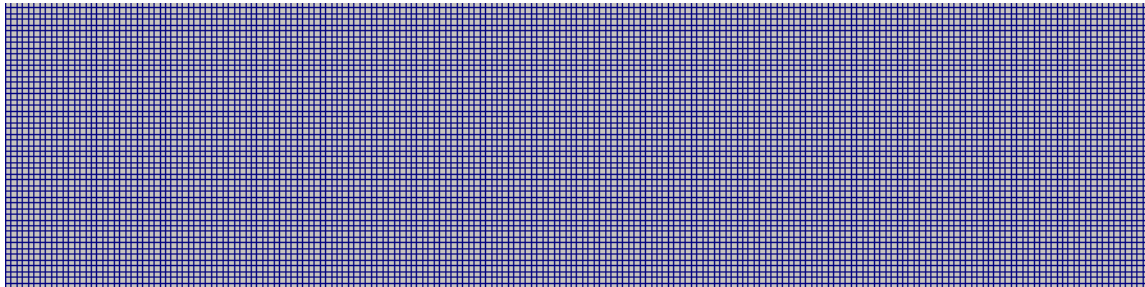


Figure 5.2 - Single turbine blockMesh discretization

The *surfaceFeatureExtract* utility was then required in order to extract and write the blades, blades refinement regions and rotating region features to an OpenFOAM readable file.

The next important step in the mesh generation procedure exploited the *snappyHexMesh* utility. This extremely useful tool provides the possibility of defining sub-regions of the domain, refining selected areas, introducing external geometries and manipulate their adjacent cells in order to create the most fitting mesh for the considered case. As first, the dictionary requires the definition of the

blades, blades refinement regions and rotating region geometries; three additional refinement regions were defined, respectively named ‘*refinementRotor*’, ‘*refinementWake*’ and ‘*refinementGeneral*’, with the purpose of obtaining a better insight of the flow behaviour respectively in the machine rotor, in the wake region and on the channel walls downstream the turbine. The *snappyHexMesh* utility is then organized in three different sections, which are here singularly analysed:

- **castellatedMesh:** this section dictates the refinement level for each defined region and for the indicated features. In general, in terms of computational time, it is the most expensive operation.

Two refinement levels are defined for each feature: the first represents the minimum imposed level, the second one instead is used in case a cell sees multiple intersection with an angle higher than a specified *resolveFeatureAngle*, always set to 20°. The refinement process consists of splitting an existing cell in two in each direction for every specified refinement level (i.e. a cell refined at level 1 turns in 8 smaller cells). Table 5.9 lists the refinement levels chosen for each surface (S) and region (R), while Figures 5.3 and 5.4 shows the computational domain after the *castellatedMesh* step.

Surface/Region	Refinement level
<i>blade# (S)</i>	(5 5)
<i>rotor (S)</i>	(4 4)
<i>refinementRotor (R)</i>	(4 4)
<i>refinementWake (R)</i>	(2 2)
<i>refinementGeneral (R)</i>	(1 1)
<i>blade#Ref (R)</i>	(6 6)

Table 5.9 - Refinement level for surfaces (S) and regions (R)

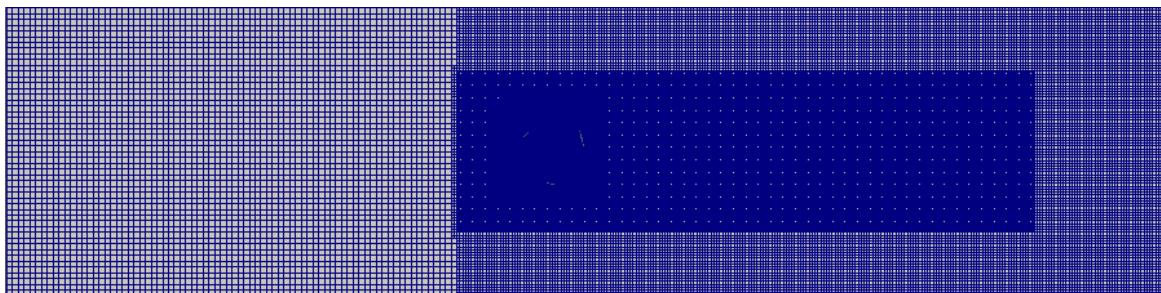


Figure 5.3 - Castellated mesh of the single turbine domain

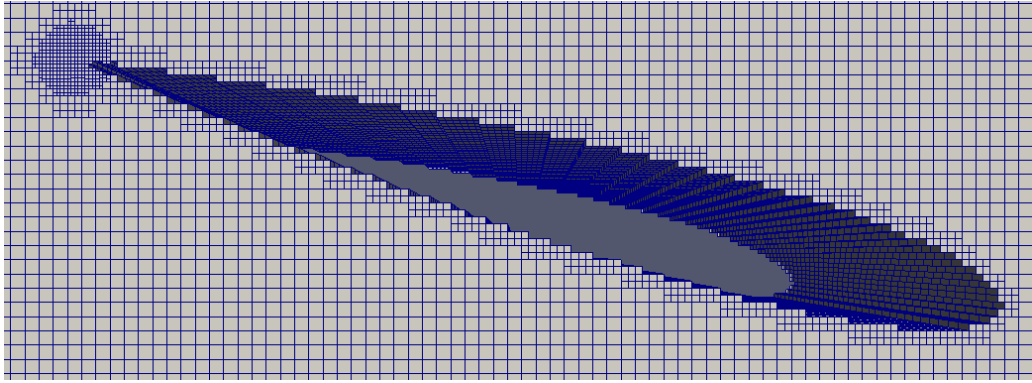


Figure 5.4 - Blade profile in castellated mesh

- **snap:** the ‘snap’ process consists of the modifications of boundary cells which intersect the geometries previously introduced; it involves an iterative process which ends when set tolerances are met. The result should be the negative representation of the considered geometry in the grid, but in general some accuracy is lost close to sharp edges and corners.

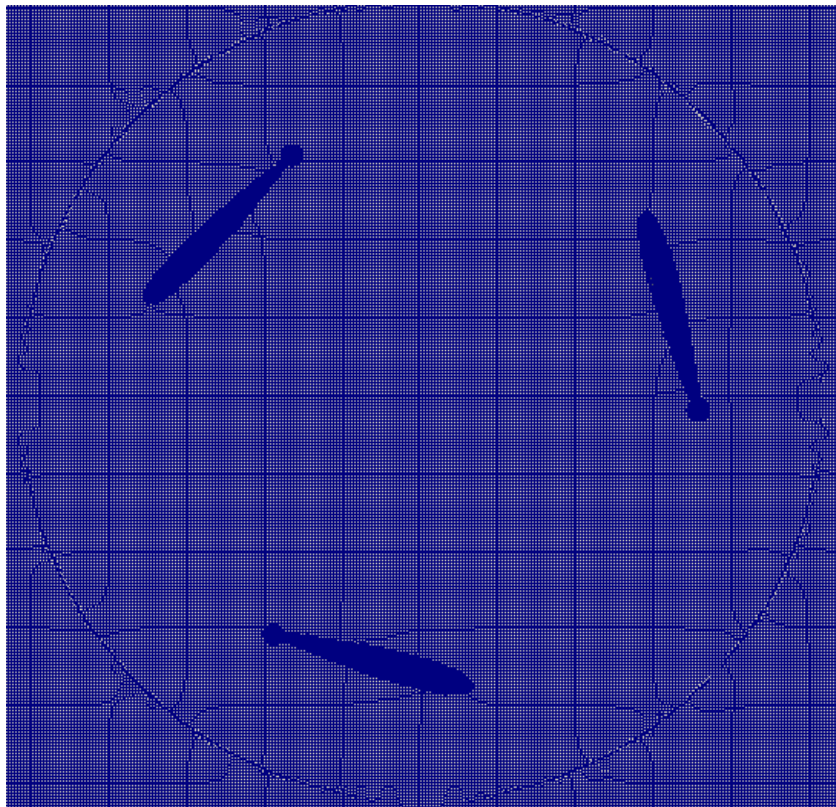


Figure 5.5 - Meshed rotor region after snapping

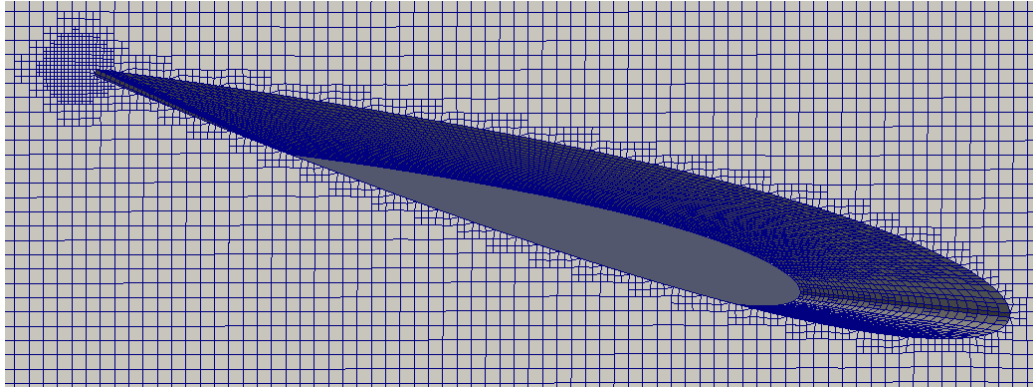


Figure 5.6 - Blade profile after snapping

- addLayers:** the last section of this utility concerns the layer addition process. Layers are very thin cells built adjacently to the reference surface, and their function is allowing the resolution of the boundary layer equations, necessary in order to gain insight into the flow-wall interaction. Some turbulence models actually do not require such a fine refinement close to walls, as those make use of general ‘*wall functions*’ to predict the quantities behaviour; these functions, despite finding a good correspondence with experiments, are not case-specific and hence are not considered accurate enough for this analysis. In order to obtain a proper viscous boundary layer resolution then, the Y^+ parameter is considered and, since wall functions were discarded, to be acceptable its value is imposed to be lower than five. Y^+ is defined as:

$$Y^+ = \frac{\Delta Y * \sqrt{\frac{1}{2} * 0.0576 * (Re)^{-\frac{1}{5}} * U_\infty^2}}{\nu} \quad (5.7)$$

Where ΔY represents the distance from the wall at which Y^+ is evaluated, Re is the Reynolds number, U_∞ is the freestream velocity and ν is the fluid kinematic viscosity. Through this formula, it was possible to obtain an estimation of the required first layer thickness. Results are reported in Table 5.10.

U_∞ [m/s]	0.25271
Re	~6000
ν [m ² /s]	$1.002 * 10^{-6}$
Y^+	0.3~0.4
ΔY [m]	0.00002

Table 5.10 - First layer thickness evaluation

Ten layers were added in the single turbine configuration. It is important to notice that the layering process strongly relies on the local background mesh and, in order to obtain a sufficiently refined grid, a refinement region for the trailing edge of each blade was required. Figures 5.7 and 5.8 show the final look of the computational domain in the vicinity of the blades.

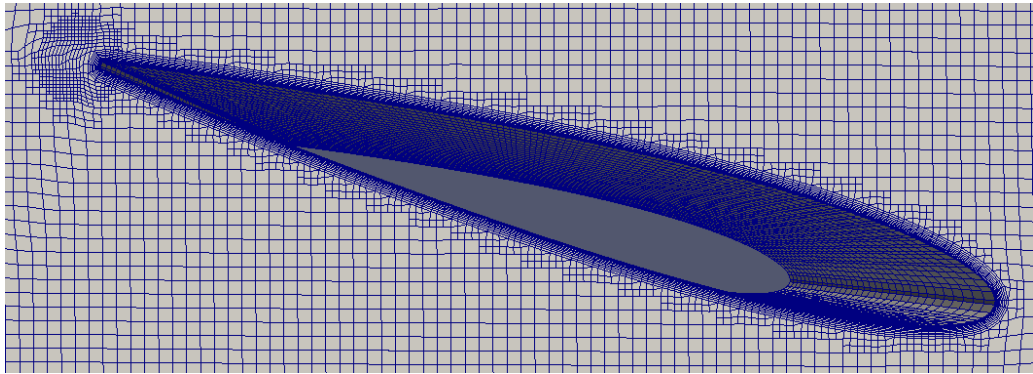


Figure 5.7 - Blade profile after layering

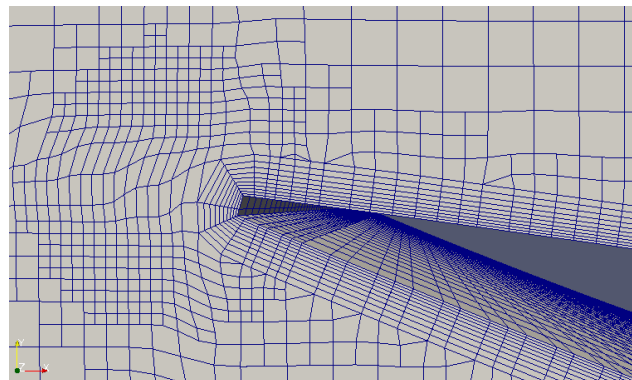


Figure 5.8 - Blade trailing edge with layers

After building the complete background mesh, the blade motion had to be set. For this purpose, first the *createBaffles* utility was employed to create two coincident cyclicAMI-type patches, respectively named ‘*rot-master*’ and ‘*rot-slave*’, referring to the cell-zone contoured by the *rotor* feature; then, through the command ‘*mergeOrSplitBaffles -split*’, the two patches were released from each other, in order to have the possibility of applying the proper relative motion to the internal *rotor* cell-zone. The boundary conditions required for the two newly defined sliding interfaces are named ‘*cyclicAMI*’: these are the same for each fluid-dynamic quantity and do not depend on the turbulence model. With these conditions, each quantity is interpolated from the stationary region to the rotating one according to its azimuthal position.

Since in general OpenFOAM deals with 3D cases, all its utilities build cells along every direction. The final step of the mesh-building process then consists of the removal of the unnecessary cells through the *extrudeMesh* dictionary, which considers the grid projection on one patch, i.e. the *back* patch, and extrudes it of an arbitrary quantity. Notice that the extrusion thickness is not relevant overall, as the power output is computed per height unit of the turbine. Figure 5.9 shows the final aspect of the mesh.

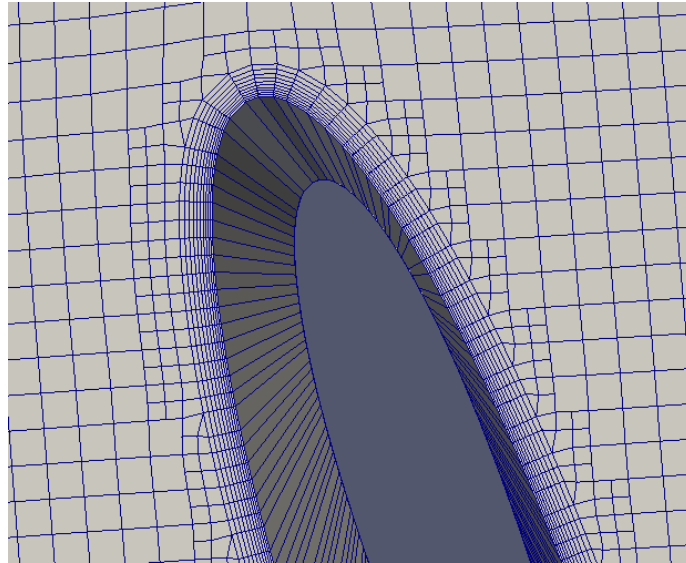


Figure 5.9 - Layered blade leading edge

Finally, the mesh motion is prescribed in the *dynamicMeshDict* dictionary. Table 5.11 reports the parameters used in this file.

Origin	<i>(0.05 0.05793 0)</i>
Axis	<i>(0 0 1)</i>
Omega	<i>7.4</i>
Rotating region	<i>rotor</i>
Motion	<i>rotatingMotion</i>

Table 5.11 - *dynamicMeshDict* settings

5.4 Mesh non-engaging counter rotating turbines

As second study case, this thesis focuses on the counter-rotating non-engaging turbine configuration. Similar analyses have already been carried out for VAWT as in [35, 36, 37] and, concerning hydrokinetic turbines, in [48], despite the usage of a different turbulence model and meshing accuracy. Considering two closely spaced turbines, the two different possibilities showed in Figure 5.10 were repetitively investigated, and the common result was the better efficiency of configuration A over B.

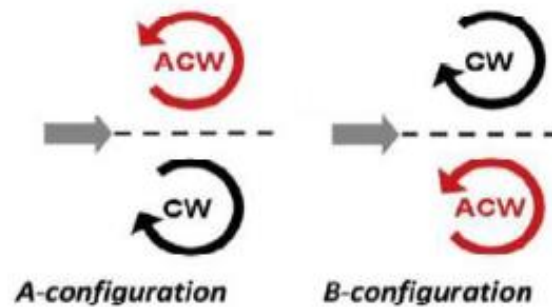


Figure 5.10 - Counter rotating configurations

The investigated turbine configuration consists then of two turbines positioned in a water channel with a distance of $1.5D$ between the axes of rotation, having the respective angular speed analogous to configuration A in Figure 5.10. A domain sketch is represented in Figure 5.1b and the domain dimensions are reported in Table 5.2. As already mentioned, the operative choice was to keep the same blockage ratio for every study case, in order to be able to compare the results; for this configuration though, because of the aisle between the turbines, the distance from each turbine and the respective wall is proportionally slightly lower than the other study cases, hence causing an enhanced blockage on the turbine. For a more accurate estimation of this effect, an evaluation of the cross-stream velocity profile between the turbines and the walls was carried out; as explained in Section 6, results showed that the same blockage conditions are in fact achieved as these profiles from different study cases match.

The STL files used for the analysis are obtained through the `'surfaceTransformPoints'` command in OpenFOAM, starting from the original STL files used in the single turbine configuration. In order to be consistent with the following case, a phase angle difference of 60° was introduced between the turbine angular positions; according to [48] though, no significant difference has been detected varying the phase difference.

The meshing and moving procedures were analogous to the single turbine, as no geometrical complexities were introduced and the classical rotating mesh technique could be employed.

In order to maintain the same cell dimensions in a wider domain, the *blockMesh* discretization was adapted accordingly. Table 5.12 reports the cell number adopted for each direction.

Direction	Dimension [m]	N. of cells
<i>x</i>	<i>1.2</i>	<i>200</i>
<i>y</i>	<i>0.6</i>	<i>100</i>
<i>z</i>	<i>0.01</i>	<i>1</i>

Table 5.12 - Non-engaging turbines domain discretization

All the other utilities employed were not significantly modified, except of course for the introduction of patches for the second turbine blades, the relative *rotor* cell zone rotating in the opposite direction and the proportional adjustment of the refinement regions. Figures 5.11, 5.12 and 5.13 show the final grid and its details.

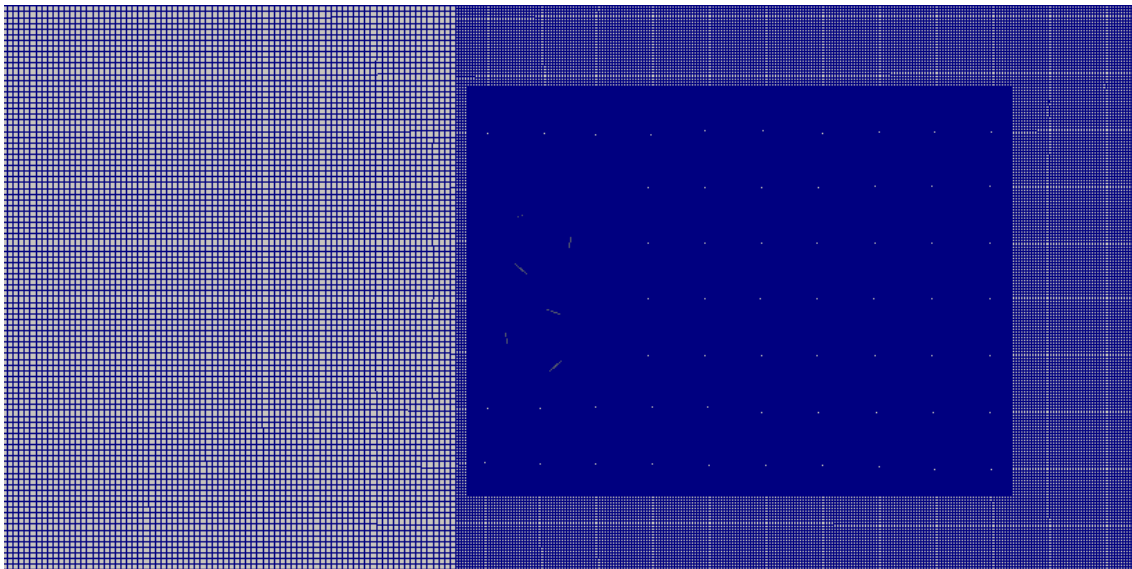


Figure 5.11 - Castellated mesh of the non-engaging turbines domain

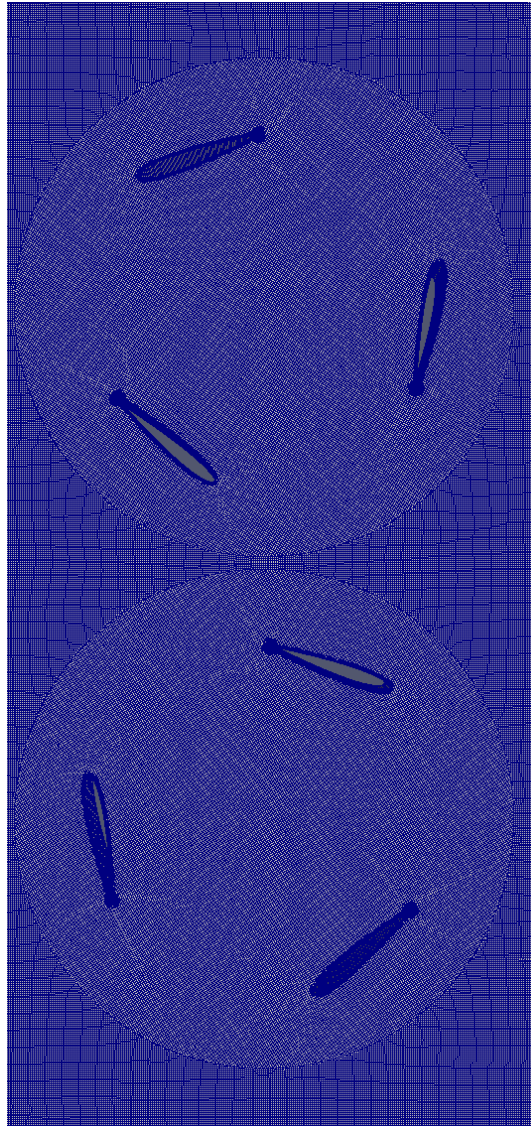


Figure 5.12 - Meshed rotors region

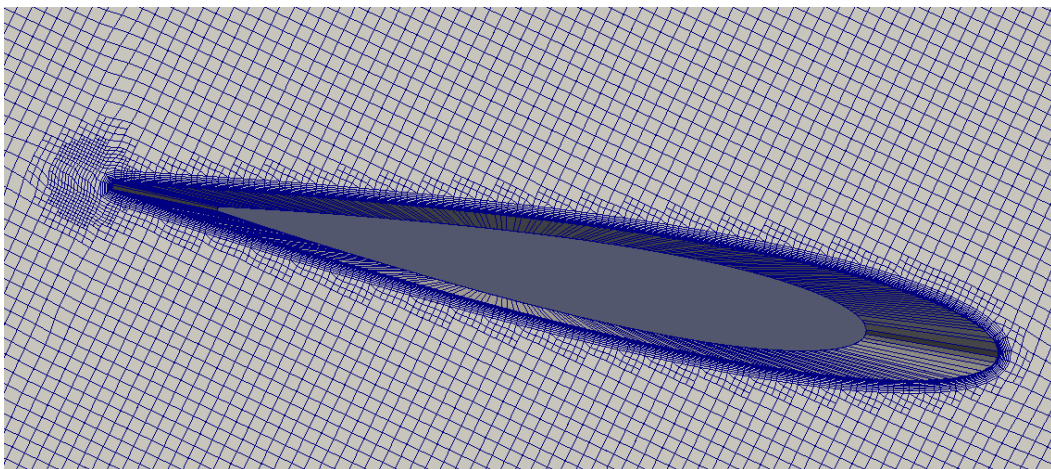


Figure 5.13 - Meshed blade profile

5.5 Mesh engaging counter rotating turbines

The final case investigated in this thesis involved two engaging counter rotating turbines and it was inspired by [48] as an extremization of the previous configuration. The turbines were set with a $0.5D$ distance between the axes of rotation and with a 60° phase difference in the angular position, mandatory to prevent the blades from collapsing on each other; moreover, as already mentioned, the rotational speed was implemented according to configuration A in Figure 5.10, which had proven to give better results in [35, 36, 37]. A domain sketch is represented in 5.1c and the domain dimensions are reported in Table 5.2, established according to the constant blockage ratio.

Contrarily to the previous analyses, since the blades movement patterns required the two rotating regions to overlap, the rotating mesh technique was not suitable for this machine. Among the alternatives presented in [49], the deforming mesh technique was selected and implemented according to [51]. The main idea behind this method is deforming the mesh until it reaches predetermined acceptability thresholds and then building a whole new mesh for the reached angular position; this process is repeated until the complete cycle of the turbines is covered, and of course it requires the definition of thresholds and a mapping tool to transfer the fluid-dynamic quantities from one mesh to the following.

The mesh parameters were very similar to the previous cases, except for the absence of rotating regions and the *blockMesh* data, which are reported in Table 5.13; additionally, the number of layers was reduced to 6 in the *snappyHexMesh* utility, as it was very difficult to obtain a valid setup resulting in properly built layers for each azimuthal position of the turbines.

Direction	Dimension [m]	N. of cells
<i>x</i>	1.2	200
<i>y</i>	0.45	75
<i>z</i>	0.01	1

Table 5.13 - Engaging turbines domain discretization

Since the meshing procedure had to be run multiple times and the *castellatedMesh* in *snappyHexMesh* required a high computational effort, it was run in parallel; two different utilities had to be implemented then, where the one operating the *castellatedMesh* options followed the *surfaceFeatureExtract* as before, while the second one was run after the *extrudeMesh* utility in order to carry out the snapping and layering processes in the already two-dimensional domain. For every mesh, the STL files were rotated into the proper position

through the *surfaceTransformPoints* command: first the features were translated towards the axes origin, then rotated of a precise quantity around the z axis and finally translated back towards the turbine axis. Figures 5.14 and 5.15 show the undeformed grid.

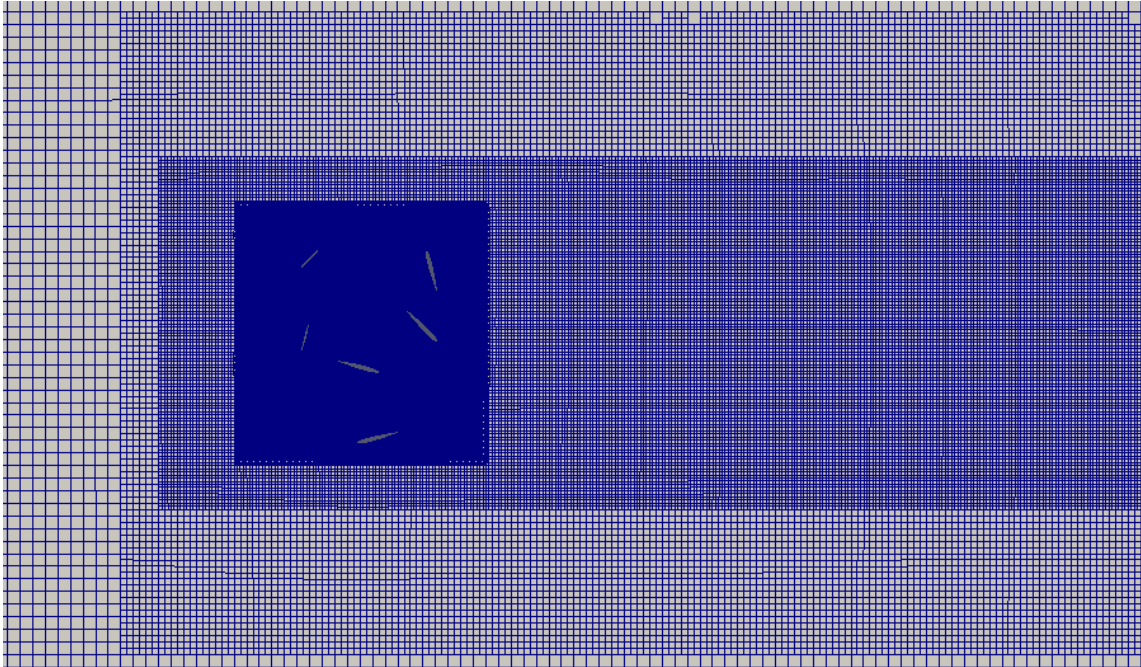


Figure 5.14 - Undeformed engaging turbines mesh

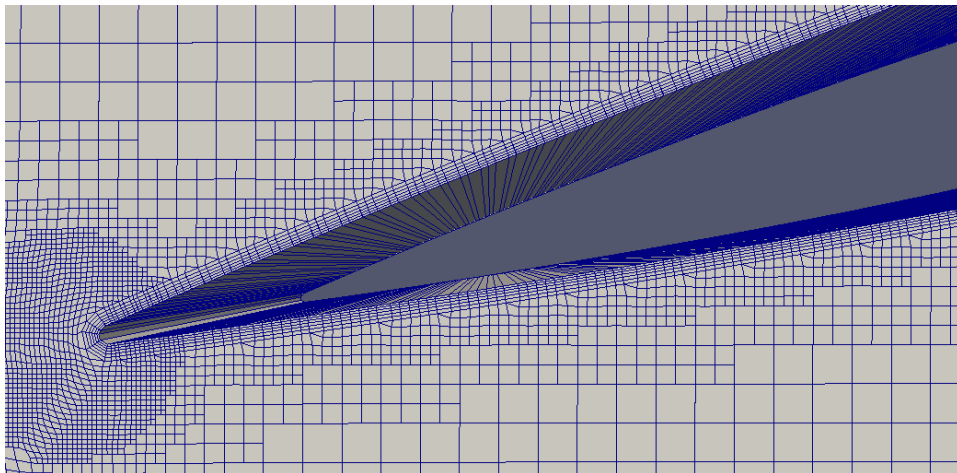


Figure 5.15 - Undeformed blade profile mesh

The grid deformation was instead prescribed in the *dynamicMeshDict* dictionary by setting *velocityLaplacian* as *motionSolver*: this solver can compute the motion of selected patches and requires the definition of a deforming criterion and an additional boundary condition. In order to preserve the mesh quality as

much as possible close to the blades, the deforming law was chosen to be *inverseDistance*, which in fact strains more the cells furthest away from the indicated patches. The additional boundary condition is called *pointMotionU* and its function is defining the law of motion of the defined patches, whether these are moving or not. OpenFOAM though does not provide a boundary condition which sets a rotating motion with constant angular speed and hence it had to be implemented manually, modifying the source code of a similar boundary condition named *oscillatingAngularVelocity*. As is explained in detail in [51], a new folder *UserBoundary* was created, containing among others the modified “.C” and “.H” source files in which the required law of motion was specified. The command ‘*wmake libso*’ was used to compile the new boundary condition *angularVelocity* and allow it to be implemented in the *pointMotionU* file. This user-defined boundary condition required also the definition of the rotation axis, the centre of rotation and the angular velocity value. Table 5.14 reports the specific boundary conditions selected for each patch, while Figures 5.16 and 5.17 show a detail of the mesh, both undeformed and deformed.

Patch	pointMotionU
<i>inlet</i>	<i>fixedValue (0 0 0)</i>
<i>outlet</i>	<i>fixedValue (0 0 0)</i>
<i>top</i>	<i>slip</i>
<i>down</i>	<i>slip</i>
<i>front</i>	<i>empty</i>
<i>back</i>	<i>empty</i>
<i>blade#Main</i>	<i>angularVelocity</i>
<i>blade#Second</i>	<i>angularVelocity</i>

Table 5.14 - pointMotionU boundary conditions

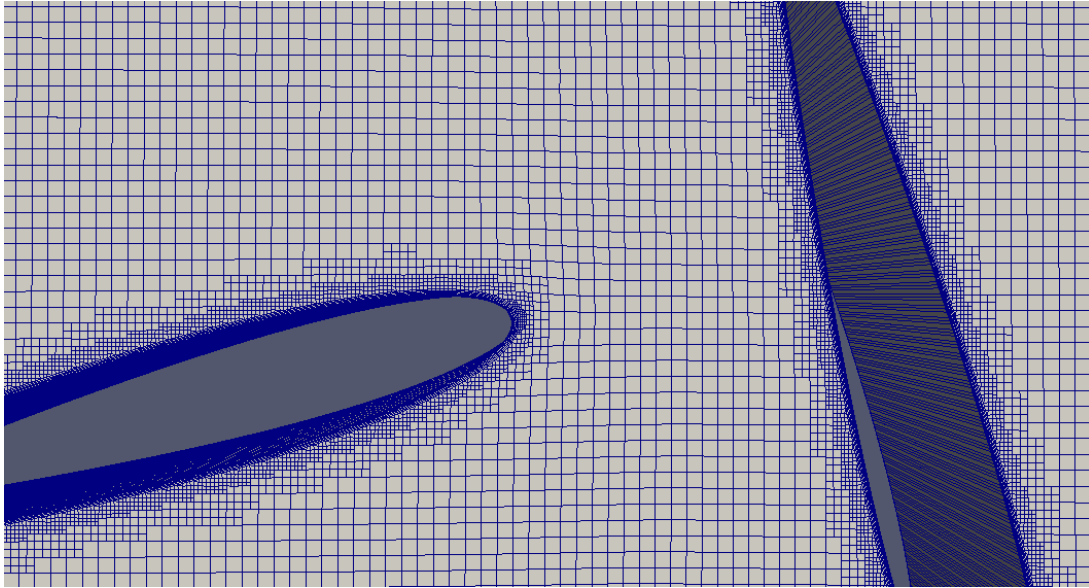


Figure 5.16 - Undeformed mesh detail

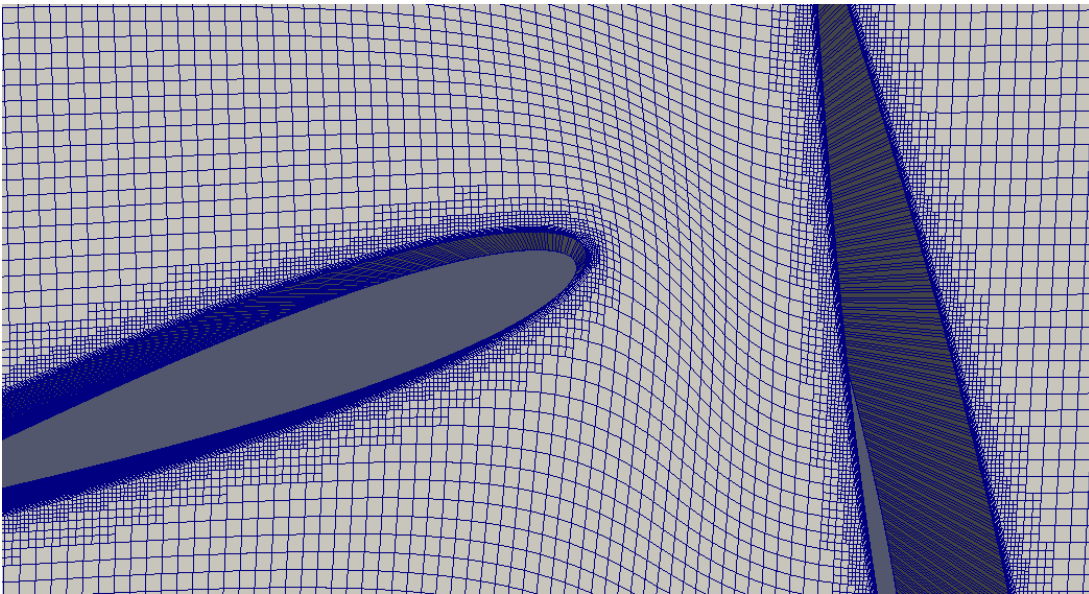


Figure 5.17 - Deformed mesh detail

The utility *checkMesh* was used to monitor the quality of the mesh during the deforming process, quantified by the *non-orthogonality* and *skewness* parameters. Selecting a maximum respective value of 65 and 3, a maximum allowable angular displacement of $3^{\circ}48'57''$ in 0.009 s was determined, resulting in 95 different starting meshes required to cover the whole turbine cycle; being the period of rotation $T = 0.84908$ s, the last mesh was deformed for a shorter time interval of 0.00308 s. All the meshes were built beforehand, named from 0 to 94 and stored in the *TurbineMeshes* folder, from where the solver algorithm would pick the proper mesh for each time instant, repeating this process for each

simulation cycle. The computations results were stored in a *TurbineResults* folder under an '*n.m*' name, where *n* represented the cycle number and *m* the mesh number; this convention was necessary in order to carry out all the post processing calculations. The *mapFields* utility was instead used in order to map the computed quantities from the deformed mesh to the following undeformed one, passing through a temporary folder named *solverPrevious*.

6 Results

Considering all the premises posed in the previous sections, this chapter now focuses on the results obtained from the numerical analysis. Several OpenFOAM simulations were carried out concerning the cases introduced in Section 5, and an additional upscaled configuration was investigated in order to have a glimpse of how a realistically scaled machine would behave.

The first analysis concerned the validation of the CFD model of the considered H-Darrieus turbine; for this purpose, both the transitional $k_T - k_L - \omega$ and the fully turbulent $k - \omega SST$ models were tested and compared with the experimental results from [48]. After selecting the most fitting turbulence model, the two different counter-rotating configurations were investigated, in order to observe how the closely spaced arrangement would impact the turbines performances. The best configuration was finally selected and upscaled to a realistic machine size.

Through the implementation of case specific Matlab scripts, the post-processing data generated by the *controlDict* dictionary were elaborated, in order to evaluate some key characterizing parameters. The most relevant are:

- A normalized convergence parameter, defined as

$$\frac{RMS(\tau^i(t) - \tau^{i-1}(t))}{\bar{\tau}^{i-1}} * 100$$

where RMS represents the Root Mean Square function, $\tau(t)$ the torque evolution in time, $\bar{\tau}$ the average torque and the apices refer to the considered laps. The aim of this parameter, evaluated for every complete lap with respect to the previous one, is firstly assessing the simulation convergence by quantifying the evolution of torque fluctuations, and secondly providing an estimation of the steadiness of the torque output, a fundamental requirement for the power extraction through an electrically driven motor.

- The average torque at the turbine shaft, used to compute the mechanical power extracted from the flow. The net torque exerted on the turbine shaft composes of two different terms, one related to pressure forces \vec{F}_p and the other to viscous forces \vec{F}_v applied on the blades, which are computed as:

$$\vec{F}_p = \sum_i \rho_i \vec{s}_{f,i} (p_i - p_{ref})$$

$$\vec{F}_v = \sum_i \vec{s}_{f,i} \cdot (\mu \overline{R_{dev}})$$

where ρ is the density, $\vec{s}_{f,i}$ the face area vector, p the pressure, μ the dynamic viscosity and $\overline{\overline{R_{dev}}}$ the deviatoric stress tensor. Pressure forces represent the useful energy contribution extracted from the flow, whilst viscous forces represent the dissipative term.

- The dimensionless power coefficient C_p of the machine, computed as

$$C_p = \frac{dP}{\frac{1}{2}\rho v_{flow}^3 dA_{front}}$$

where dA_{front} is the case specific frontal area of the machine. This parameter is computed considering the mesh thickness parameter, specified in the *extrudeMeshDict*, and the cross-stream section of the considered configuration, being $1*D$ for the single turbine, $2*D$ for the non-engaging configuration and $1.5*D$ for the engaging configuration.

- The transversal velocity profile at the turbine centre, in order to evaluate the percentual acceleration due to blockage.

In order to build a dimensionless characteristic curve revealing the optimal working condition, all the study cases were tested for several of those varying the Blade Speed Ratio λ (BSR), and hence the flow velocity accordingly. BSR is defined as

$$\lambda = \frac{\omega R}{v_{flow}}$$

6.1 Single Turbine

As previously mentioned, the single turbine case was exploited as testing for the turbulence model to apply to the other configurations. Considering the experimental curve presented in [48], seven simulations for each turbulent model were carried out in order to cover the whole BSR range, and the results are showed in Figure 6.1.

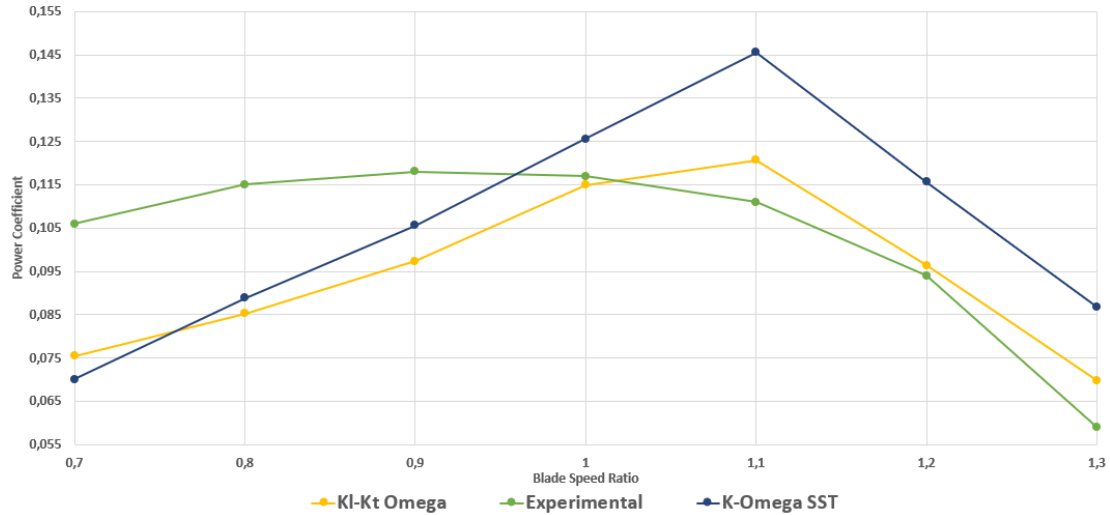


Figure 6.1 – Single turbine power coefficient curves comparison

The first consideration concerns the difference between the turbulence models: the $k - \omega$ SST curve stands above the transitional model curve for the majority of the investigated flow velocities, and $BSR = 0.7$ represents the only exception; interestingly then, moving towards lower flow velocities (higher BSR), the difference between the curves increases. This is a direct consequence of the $k_T - k_L - \omega$ structure which models the flow as transitional for higher BRSs, hence reducing the estimation of the pressure torque at the shaft and the power coefficient accordingly; on the other hand, when the flow velocity is higher, its results as expected get closer to the fully turbulent ones.

As clearly appears from the plot, considering that three dimensional effects were not accounted for, a very good correspondence was found between experimental values and the $k_T - k_L - \omega$ model for BSRs higher than unity, whilst for lower values it provided an underestimation of the power coefficient of about 10% to 25%. In [49] though, the same authors from [48] presented several simulation results concerning a $BSR \approx 0.85$ clearly showing that, despite using a different mesh and a different turbulence model, the final computed power coefficient lied approximately between 0.088 and 0.100. Firstly the considerations by the authors themselves, and then the results obtained in this work, suggest that the

experimental curve presented in [48] might be slightly offset due to some measurement systematic error, or alternatively a very high uncertainty, most likely due to the turbulent nature of the flow and to the relatively small scale of the experimental setup. However, the fully turbulent model resulted in a clear overestimation of the generated power on the right side of the plot, and in a slightly lower underestimation on the left side; for this reasons, in agreement with the considerations from Section 3.2, the most fitting model was selected to be the transitional $k_T - k_L - \omega$.

As for the simulations convergence and fluctuations, Table 6.1 reports the behaviour of $BSR = 0.8$ and $BSR = 1.2$.

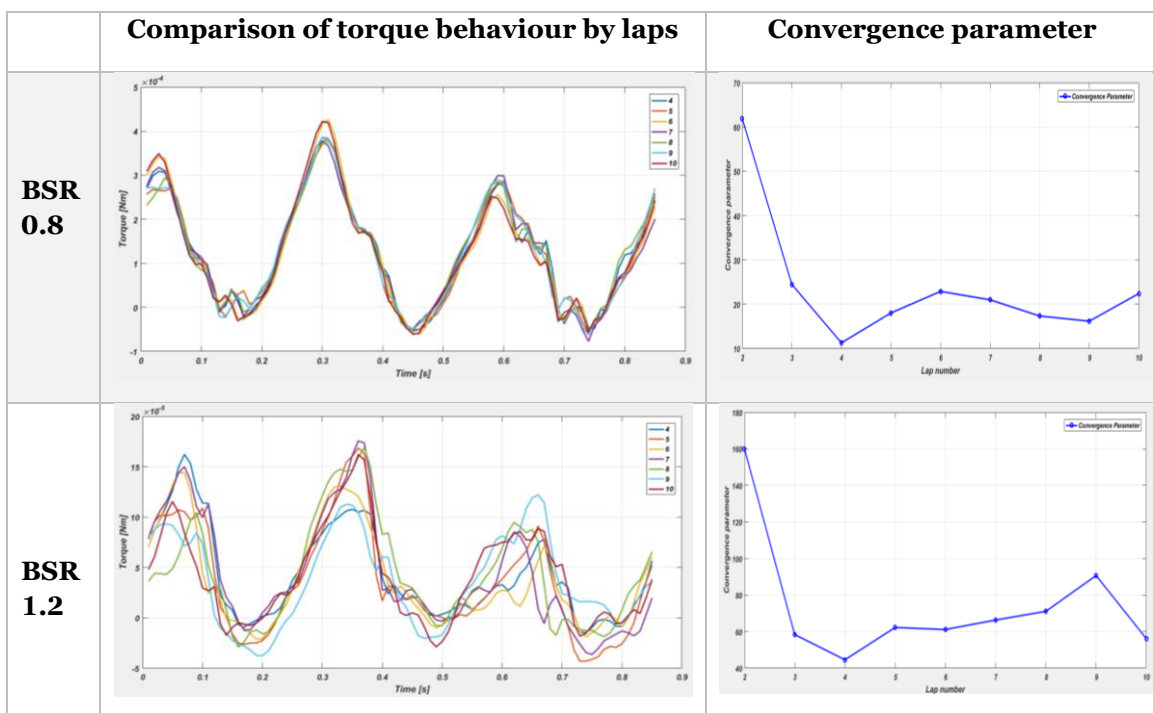


Table 6.1 - Torque and convergence behaviour for different flow regimens

These plots clearly highlight the stability difference encountered for different flow velocities: considering the transitional regime, despite the overall torque signal being approximately the same over different laps, fluctuations up to about 80% of the average torque value were observed, whilst for the fully turbulent regime the same parameter stabilized around 10% in most cases.

6.2 Counter rotating non-engaging turbines

The second study case considered two turbines arranged as described in Section 5.4, with equal rotational speed in module, but opposite in direction. As concluded in the previous section, the adopted turbulence model was $k_T - k_L - \omega$ and, since the best efficiency point for the single turbine was $BSR = 1.1$, this analysis focused on a flow velocity range from $BSR = 0.9$ to $BSR = 1.3$. Less simulations were carried out because of the higher computational effort required. Figure 6.2 compares the obtained curve with the one from the single turbine case.

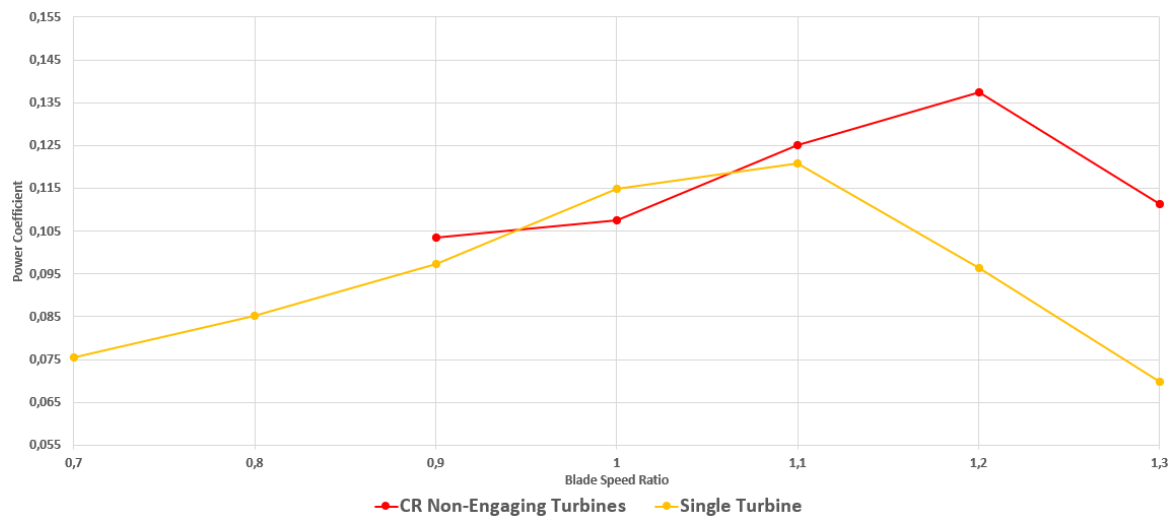


Figure 6.2 - Single turbine and non-engaging turbines power coefficient curves comparison

In the chart the best efficiency point for this turbines arrangement resulted to be for a higher blade speed ratio, in particular for $BSR = 1.2$. Considering the peak values of the two curves, a performance increment of about 13.9% was registered, hence proving the beneficial effects that the turbines exert on each other. Considering $BSR = 1.2$ as a reference, these effects are particularly evident when the torque behaviour in time is considered. The curves in Figure 6.3 represent the torque evolution in time, averaged on 7 turbine laps, with the relative confidence interval at 50% in order to account for its fluctuations.

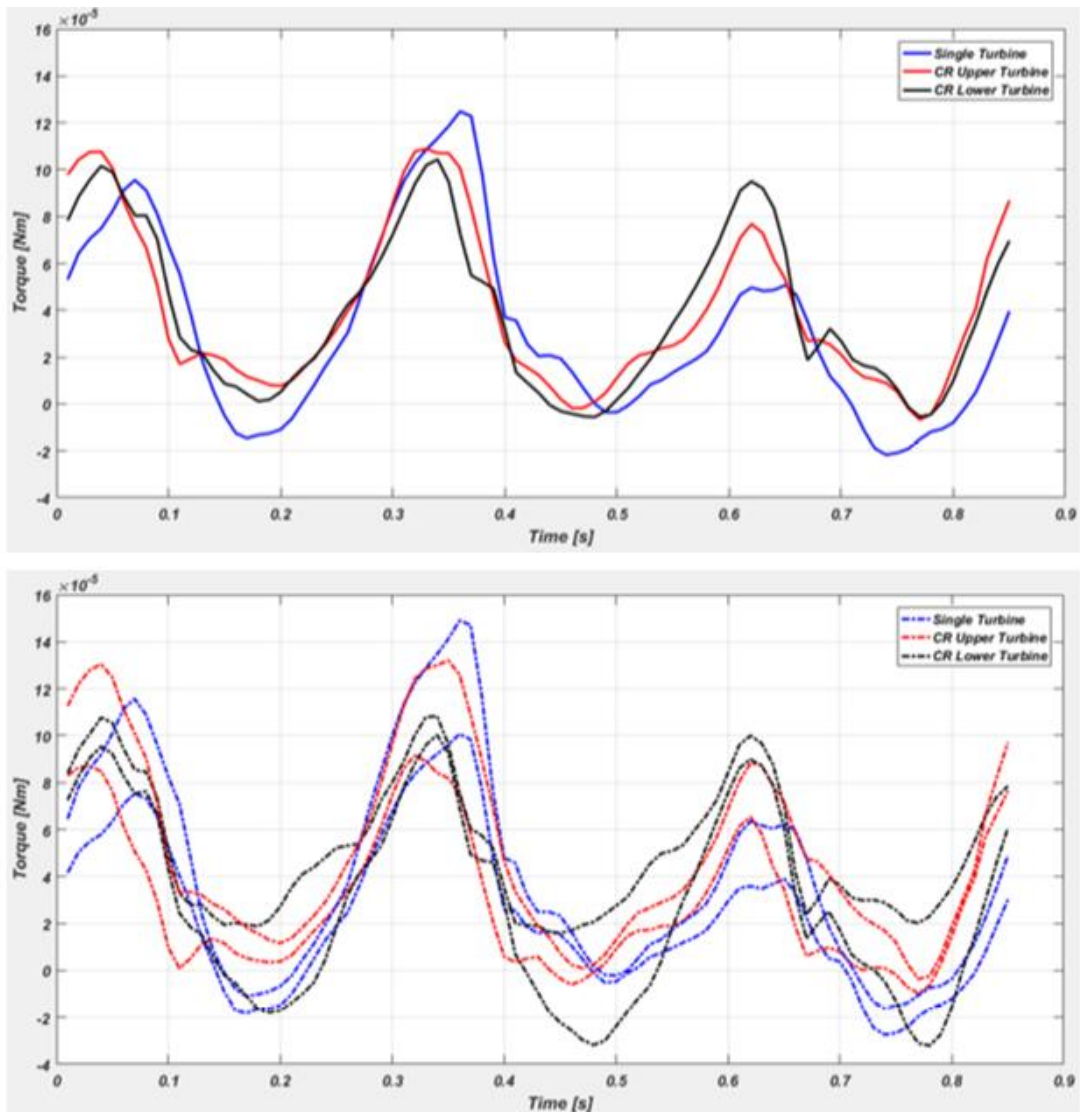


Figure 6.3 – Average torque behaviour (a) and 50% confidence interval (b) for different configurations

Comparing the single turbine curve with the torques from the CR configuration, it appears that the benefits mainly come from an almost complete reduction of the negative torque region, whilst the positive peaks change less significantly. This is due to the fact that, as the negative torque region matches the blades passage in the channel between the two machines, exactly where the flow is accelerated by the turbines motion, the adverse pressure gradient, which generated negative torque in the single turbine case, is almost completely cancelled.

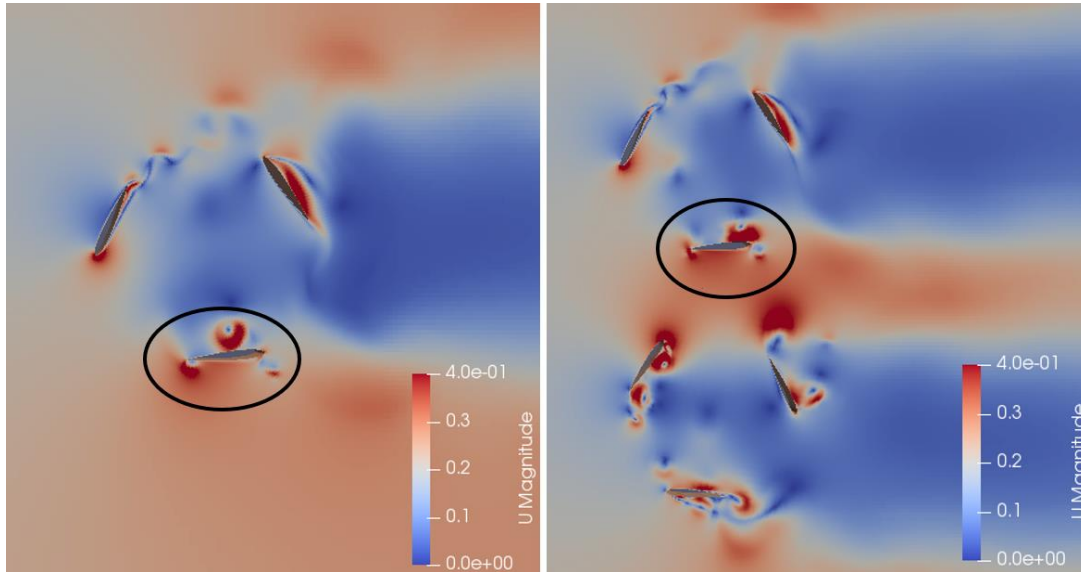


Figure 6.4 - Velocity field comparison: single turbine (a) vs non engaging turbines (b)

In Figure 6.4b it is possible to appreciate for the blade passing in the inner channel how the velocity is higher and more uniform with respect to Figure 6.4a, especially around the trailing and leading edge areas. These considerations find their match in the average torque evaluation at the shaft, which do not show any significant variation concerning the dissipative viscous contribution but show a relevant increment in the positive pressure contribution. Table 6.2 reports as example the values computed for the discussed case.

	\mathbf{pM}_z [Nm]	\mathbf{vM}_z [Nm]	\mathbf{M}_z [Nm]	\mathbf{C}_p
Single T.	$5.608 * 10^{-5}$	$-1.596 * 10^{-5}$	$4.012 * 10^{-5}$	<i>0.0963</i>
CR Upper T.	$7.555 * 10^{-5}$	$-1.532 * 10^{-5}$	$6.023 * 10^{-5}$	<i>0.1375</i>
CR Lower T.	$-7.367 * 10^{-5}$	$1.534 * 10^{-5}$	$-5.833 * 10^{-5}$	<i>0.1375</i>

Table 6.2 - Pressure torque, viscous torque, net torque and power coefficient comparison between single and non-engaging turbines

6.3 Counter rotating engaging turbines

The last investigated arrangement, already described in Section 5.5, involves two counter rotating turbines whose blades paths cross each other. The applied turbulence model was again $k_T - k_L - \omega$, and the power coefficient results concerning the $BSR = 1.1$ are presented in Table 6.3.

	pM_z [Nm]	vM_z [Nm]	M_z [Nm]	C_p
Single T.	$8.233 * 10^{-5}$	$-1.561 * 10^{-5}$	$6.671 * 10^{-5}$	<i>0.1192</i>
CR Upper T.	$3.302 * 10^{-5}$	$-2.167 * 10^{-5}$	$1.135 * 10^{-5}$	<i>0.0080</i>
CR Lower T.	$-1.865 * 10^{-5}$	$2.328 * 10^{-5}$	$0.463 * 10^{-5}$	<i>0.0080</i>

Table 6.3 - Pressure torque, viscous torque, net torque and power coefficient comparison between single and engaging turbines

By looking at the power coefficient values, it appears that not only this configuration does not improve the performances of the turbines, but it also almost completely erases the generated power, bringing the power coefficient below 1%; in fact, as the reported torque values show, there are both a relevant decrease concerning the average pressure torque and a non-negligible increment of the dissipative viscous torque. Comparing the overall torque behaviour in time from Figure 6.5 with Figure 6.6, it emerges that the performance reduction is due more to the larger relevance of the negative torque regions, rather than to a decrement of the torque peak values.

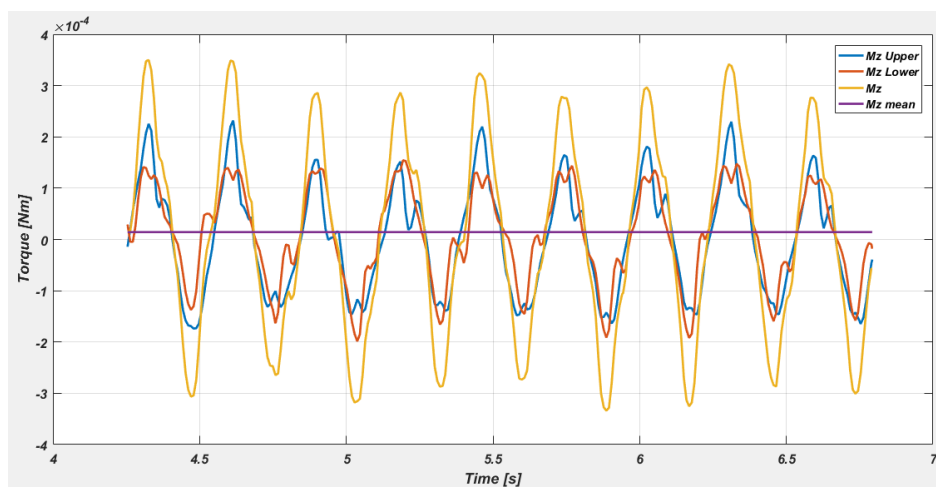


Figure 6.5 - Overall torque behaviour from engaging turbines configuration

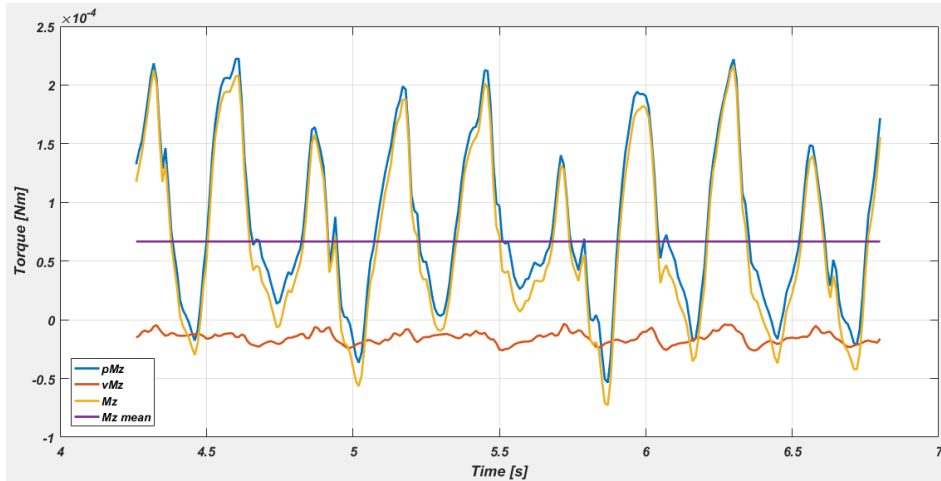


Figure 6.6 - Overall torque behaviour from single turbine configuration

For a deeper understanding of the phenomenon, the torques acting on the single blades were compared with the ones from the single turbine configuration and, as confirmed by Figure 6.7, while the torque peaks are approximately constant in each configuration, in the negative torque region the single turbine curves stand almost constantly above the others.

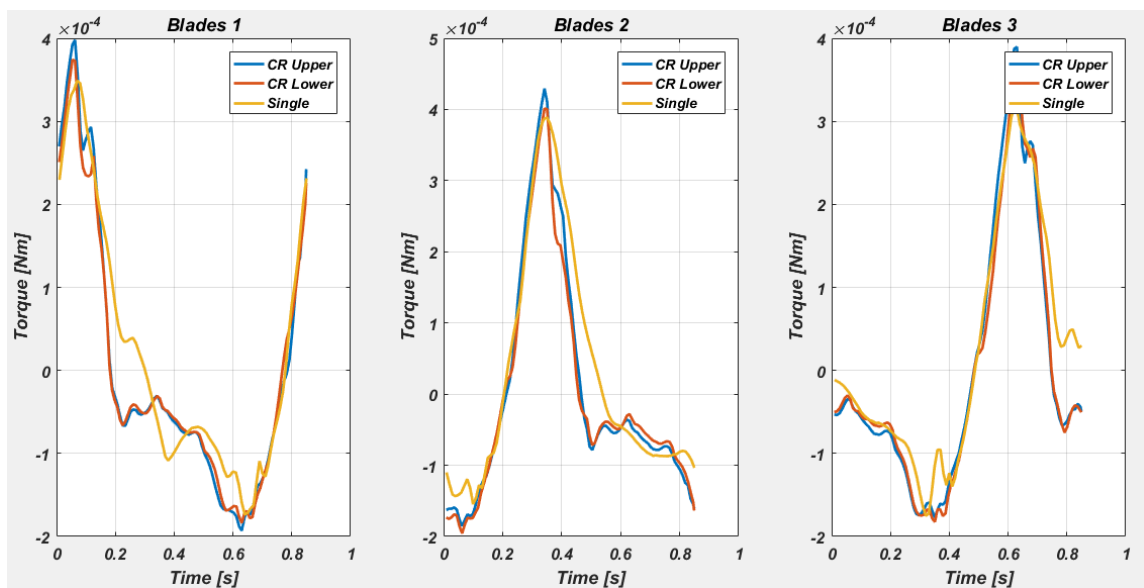


Figure 6.7 - Single blade torque behaviour comparison

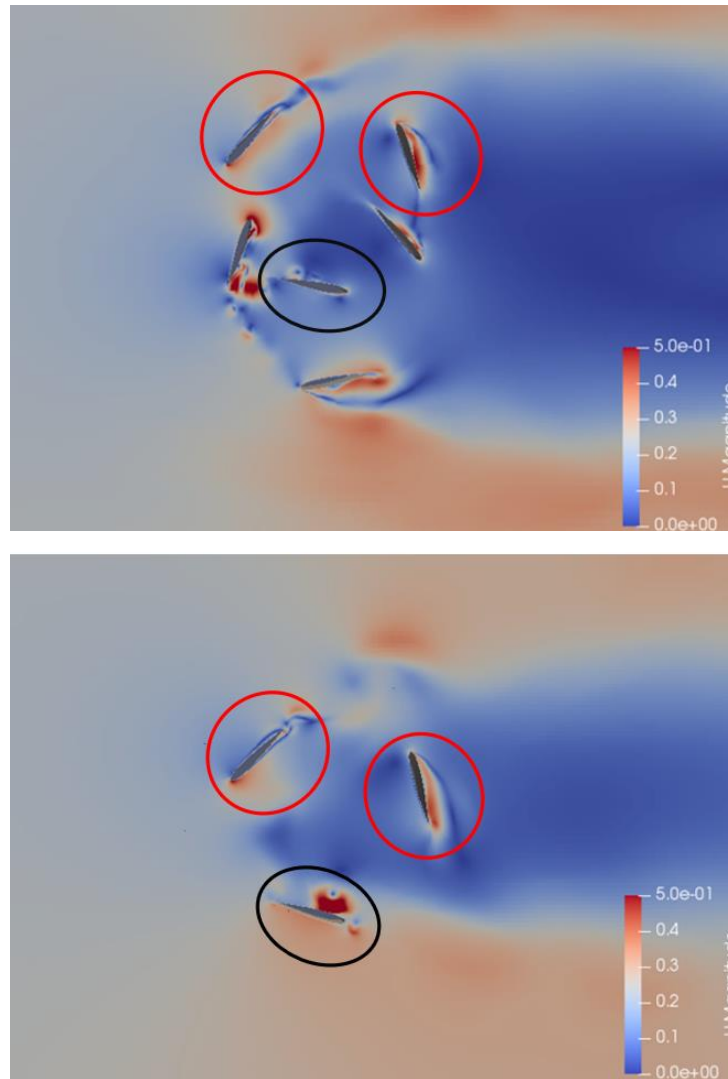


Figure 6.8 - Velocity fields comparison: engaging turbines (a) vs single turbine (b)

Finally, Figure 6.8 shows the differences concerning the velocity field between the engaging counter rotating (a) and the single turbine configuration (b): the red circles point at the velocity field similarities in the positive torque azimuthal positions, responsible for similar torque peaks, whilst the black one highlights the great disturbance generated by the lower turbine, which worsen the efficiency of the counter rotating engaging configuration.

In conclusion it results that, despite providing a reduced usage of the available space, the turbines performances are heavily affected by the detrimental overlap of the blades wakes in the rotors region, hence making the engaging counter rotating configuration a not viable option.

6.4 Upscaled Turbine

The final analysis carried out in this thesis aimed at proposing an engineering relevant turbine configuration able to efficiently generate power. Considering the results from previous analysis, the counter rotating non-engaging turbines configuration was selected for the test and upscaled considering a geometrically similar structure: the diameter was increased to 1 meter, hence determining a scaling factor $\alpha \approx 14.64$, and all the other dimensions were scaled accordingly. Table 6.4 reports the most relevant parameters.

	Small scale	Large scale
Diameter [m]	<i>0.0683</i>	<i>1</i>
Chord [m]	<i>0.0254</i>	<i>0.3719</i>
Domain Length [m]	<i>1.2</i>	<i>17.5695</i>
Domain Width [m]	<i>0.45</i>	<i>6.5886</i>

Table 6.4 - Upscaled non-engaging turbines fundamental characteristics

Assuming a realistic value, the water flow velocity was imposed at 2.5 m/s and it was used in order to establish the new rotational velocity of the turbines. According to the BSR definition in (2.13) and using its peak value for the considered small scale configuration $\lambda = 1.2$, the rotational speed was computed as:

$$\Omega = \frac{\lambda * U_{\infty}}{R} = 6 \frac{\text{rad}}{\text{s}}$$

As for the turbulence model, the Reynolds number on the blade chord was evaluated as in Section 3.2:

$$Re = \frac{U_{\infty} * C}{\nu} \approx 8.45 * 10^5$$

Along with the previously made considerations, such value suggested that, being it higher than the $2 * 10^5$ threshold, a fully turbulent model would best fit this analysis and provide the most accurate results; for this reason, the $k - \omega \text{ SST}$ model was implemented, and the boundary conditions were adapted accordingly.

The meshing process was analogous to the small scale turbine, except of course for the cells dimensions which were upscaled as well; maintaining the same *blockMesh* discretization, the base cell was $\Delta X \approx 0.088\text{m}$. In order to obtain a properly accurate solution in the boundary layer region, 12 layers were built, and the first layer thickness was re-evaluated aiming at $Y^+ \approx 3$. Other more refined meshes were tested trying to reach lower Y^+ values, but the high non-

orthogonality levels would not allow an acceptable computational robustness of the problem. Nevertheless, being the adopted Y^+ value for this analysis characteristic of the viscous sublayer region of the boundary layer, it holds as a reasonable choice. Table 6.5 reports the results.

U_∞ [m/s]	2.5
Re	$8.45 * 10^5$
ν [m ² /s]	$1.1 * 10^{-6}$
Y^+	~ 3
ΔY [m]	0.000032

Table 6.5 - First layer thickness evaluation

Figures 6.9 and 6.10 show some details concerning the final mesh and its layers on the blades profile.

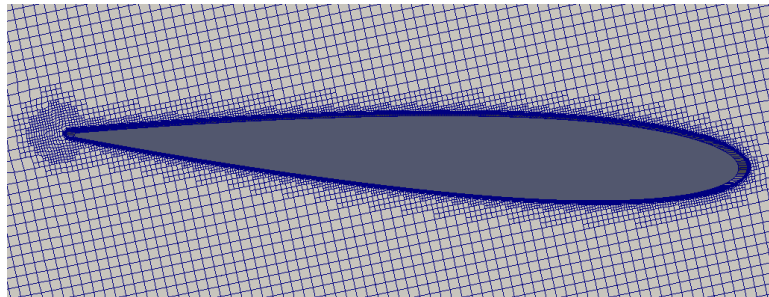


Figure 6.9 - Meshed blade profile

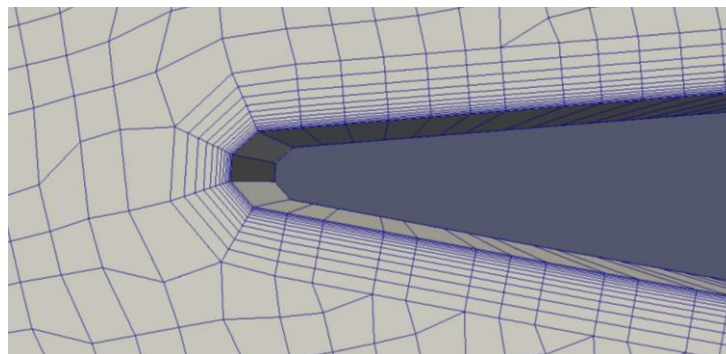


Figure 6.10 - Layered blade trailing edge

The large scale configuration was tested for $BSR = 1.2$, in order to evaluate the Reynolds effect with respect to the small scale turbines case; the results are compared in Table 6.6. Notice that the torque terms refer to the contributions of both turbines at the shafts, which are approximately similar.

	pM_z [Nm]	vM_z [Nm]	M_z [Nm]	C_p
Small Scale	$14.922 * 10^{-5}$	$-3.066 * 10^{-5}$	$11.856 * 10^{-5}$	<i>0.1375</i>
Large Scale	10.514	-0.251	10.263	<i>0.3941</i>

Table 6.6 - Large scale machine performance evaluation

As clearly emerges from the power coefficient evaluation, the upscaling process resulted in a great improvement of the turbines efficiency and, as expected, of the net torque at the shaft. The main reason for the C_p increment is the relevance of the viscous torque with respect to the pressure torque: in fact, while in the small scale configuration vM_z represents about 20.6% of pM_z , the large scale turbine term vM_z weights about ten times less, being about 2.4% of the pressure torque. This great difference is due to the fact that the blades boundary layers, where the viscous torques vM_z are computed, even though the whole domain was upscaled, do not grow proportionally: considering in fact the cell thicknesses of the first layer for the two cases, the scaling factor is $\alpha_{BL} = 0.000032/0.00002 = 1.6$, much lower than the scaling factor α . Considering that this analysis did not account for three dimensional effects, the result matches the values presented in [24] as general expectation from the performances point of view.

As for the analysis convergence, Figure 6.11 shows the convergence parameter behaviour throughout the simulation laps for both the upper and lower turbine.

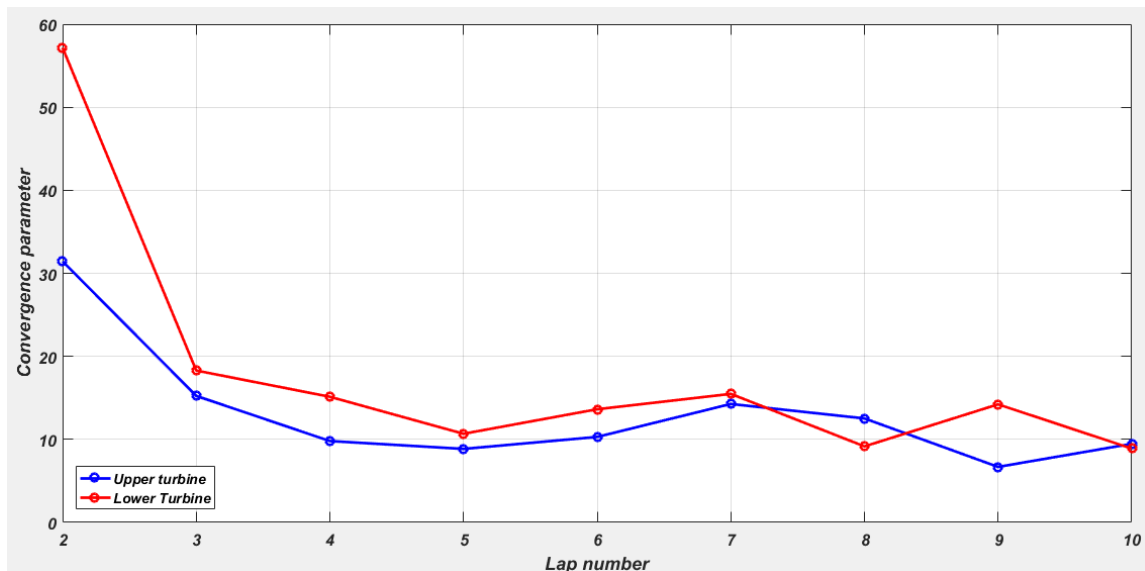


Figure 6.11 - Upscaled machine convergence behaviour

For both turbines, the convergence parameter stabilized around 10% of the average torque value, proving both the analysis to have reached convergence and a relatively regular torque output.

6.5 Blockage Analysis

As previously explained, all the analysis presented in this thesis were carried out keeping a constant geometrical blockage, equal to one adopted in [48] for the experimental study; this condition was achieved by proportionally enlarging the water channel according to the cross-stream dimension of the considered machine. As final validation of the work done, this section proposes to compare the velocity cross-stream profiles among the four different studied configurations, in order to confirm that the flow acceleration on the machine sides is the same for each of them. In order to avoid the influence of the wake on the velocity profile, it was evaluated in correspondence of the machine centre and then normalized with respect to the undisturbed flow velocity. Figure 6.12 reports the velocity profile from the four considered configurations.

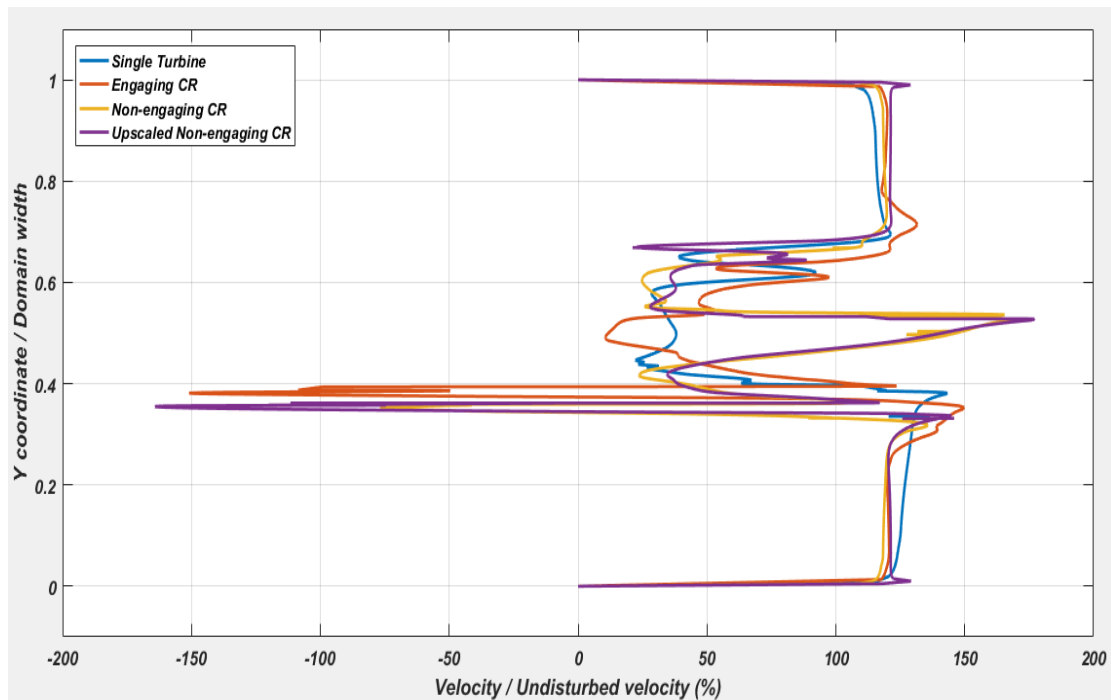


Figure 6.12 - Blockage analysis of the different turbines configurations

As appears from the plot, the velocity profiles in the rotors regions are very different among the configurations, due to the peculiar turbulent structures generated by different blade motions; on the other hand, considering the profiles on the machines sides, it emerges that in fact the flow is equally accelerated among the configurations, reaching a velocity value for each case of about 120% of the undisturbed flow velocity. Being the only asymmetric case though, when compared to the other profiles, the single turbine curve presents a slightly higher velocity between the machine and the bottom wall, and a slightly lower velocity between the machine and the top wall. The main reason for this is actually the

counter-clockwise turbine rotating motion itself, which lowers the flow inertia in the upper region and increases it below the turbine. Of course, as the other configurations are symmetric with respect to the machine centre, this phenomenon does not emerge from the other curves, but the profiles appear symmetric as well.

Overall, Figure 6.12 proves that the studied cases were actually carried out with the same blockage conditions, further confirming that the non-engaging counter rotating configuration would be the most efficient arrangement in a fixed channel.

7 Conclusions

This thesis proposed and analysed innovative and alternative configurations for the considered hydrokinetic H-Darrieus turbine, by means of CFD modelling of an existing experimental setup.

Through the CFD opensource software OpenFOAM, the first step was reproducing and meshing the physical model described in [48], consisting of an isolated hydrokinetic H-Darrieus turbine, posed in a confined water channel and rotating with fixed angular velocity. The basic grid parameters, such as cell dimensions, layering features and refinement levels were inspired by the detailed analysis carried out by Padricelli in [39]. As part of the modelling process then, the turbulence model choice proved to be critical: both the fully turbulent $k - \omega SST$ and the transitional $k_T - k_L - \omega$ models were considered as valid candidates, but, as showed in Figure 6.1, the second turned out to be the best fit for the studied cases.

Once the numerical model validation reached satisfactory precision levels, the multiple turbine configurations were investigated. Great attention was given to the machines working conditions, as, in order to find the best approach to exploit a given waterflow, it was mandatory to compare the alternatives on an even field; for this reason, all the configurations were tested within the same flow velocity range and with the same blockage factor.

Firstly, the analysis focused on a counter rotating turbines configuration where the two turbines were arranged with 1.5D distance between the axes of rotation; as no particular geometrical issues arose due to the non-engaging blades paths, the meshing procedure was analogous to the single turbine case. Opposingly, the secondly studied counter rotating configuration was designed with 0.5D distance between the rotational axes, in such a way that the blades paths would cross each other. This arrangement therefore required a different meshing technique, which consisted of deforming a regularly built grid until some acceptance thresholds were met; once every mesh had reached its deforming limit, a new undeformed one was built, and the computed fields were mapped from one to the other. This process required several iterative procedures, along with the implementation of a user-defined boundary condition for imposing the rotational blade motion.

The CFD simulations results showed that the non-engaging counter rotating configuration actually improved the turbine performances by about 13.9%, because of the peculiar velocity fields interaction; as for the engaging counter rotating turbine though, the computed efficiencies clearly demonstrated that such an arrangement would be highly detrimental for the overall machine performances.

The final analysis concerned an upscaled version of the counter rotating non-engaging configuration, studied in what would be, according to literature, a more realistic environment. As reported in Table 6.6, the simulation result presented a power coefficient of 39.41%, about three times higher than the small-scale configuration, resulting from a much lower impact of the viscous forces on the net torque at the shafts.

For a better understanding of the H-Darrieus turbines behaviour in hydrokinetic applications, further studies and developments should firstly focus on evaluating the C_p curve of the upscaled machine over a wider BSR range, in order to find the most performing velocity interval for this machine type. An interesting challenge, which could increase the competitiveness, the feasibility and the viability of this technology, concerns then the blades shape optimization, which could greatly improve both the efficiencies and the power output of the turbines. Finally, an experimental validation of the proposed solution would be required in order to further confirm the achieved results.

Appendix A

This conclusive section collects some representative OpenFOAM dictionaries, implemented for both the mesh generation processes and the CFD simulations. The following scripts refer to the single turbine configuration, which was always considered as the fundamental reference case; nevertheless, the parameters choice for every configuration is explained in depth in Section 5.

Dictionaries for the mesh generation process

The firstly reported dictionary is the *blockMeshDict*, used to set the first raw discretization of the computational domain.

```
/*-----*- C++ -*-----*\
|=====|
|  \ \ /  /  F i e l d      | OpenFOAM: The Open Source CFD Toolbox
|  \ \ /  /  O peration    | Version: 2.3.0
|  \ \ /  /  A nd          | Web: www.OpenFOAM.org
|  \ \ /  /  M anipulation |
\*-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       blockMeshDict;
}

// *****

convertToMeters 1;

vertices
(
    (-0.51715 -0.09207 0.01)
    ( 0.68285 -0.09207 0.01)
    ( 0.68285  0.20793 0.01)
    (-0.51715  0.20793 0.01)
    (-0.51715 -0.09207 0.03)
    ( 0.68285 -0.09207 0.03)
    ( 0.68285  0.20793 0.03)
    (-0.51715  0.20793 0.03)
);

blocks
(
    hex (0 1 2 3 4 5 6 7) domain (200 50 1) simpleGrading (1 1 1)
);

edges
(
);

boundary
```

```

(
  top
  {
    type wall;
    faces
    (
      (3 7 6 2)
    );
  }
  down
  {
    type wall;
    faces
    (
      (1 5 4 0)
    );
  }
  inlet
  {
    type patch;
    faces
    (
      (0 4 7 3)
    );
  }
  outlet
  {
    type patch;
    faces
    (
      (2 6 5 1)
    );
  }
  front
  {
    type empty;
    faces
    (
      (4 5 6 7)
    );
  }
  back
  {
    type empty;
    faces
    (
      (0 3 2 1)
    );
  }
);

// ***** //

```

Secondly the *snappyHexMeshDict* dictionary is presented, where the castellated mesh is built and snapping and layering processes are carried out.

```

/*-----*- C++ -*-----*/
|=====|
|  \ \ /  /  F i e l d      | OpenFOAM: The Open Source CFD Toolbox
|  \ \ /  /  O p e r a t i o n  | Version:  2.3.0
|  \ \ /  /  A n d              | Web:      www.OpenFOAM.org
|  \ \ /  /  M a n i p u l a t i o n  |
|-----|
/*-----*- C++ -*-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       snappyHexMeshDict;
}
// * * * * *

castellatedMesh true;
snap            true;
addLayers       true;

// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    blade1.stl
    {
        type triSurfaceMesh;
        name blade1;
    }

    blade2.stl
    {
        type triSurfaceMesh;
        name blade2;
    }

    blade3.stl
    {
        type triSurfaceMesh;
        name blade3;
    }

    blade1ref.stl
    {
        type triSurfaceMesh;
        name blade1ref;
    }

    blade2ref.stl
    {

```

```

        type triSurfaceMesh;
        name blade2ref;
    }

blade3ref.stl
{
    type triSurfaceMesh;
    name blade3ref;
}

rotor.stl
{
    type triSurfaceMesh;
    name rotor;
}

refinementRotor
{
    type searchableBox;
    min (-0.005    0.00293 0.01);
    max ( 0.105    0.11293 0.03);
}

refinementWake
{
    type searchableBox;
    min (-0.05    -0.02707 0.01);
    max ( 0.550    0.14293 0.03);
}

refinementGeneral
{
    type searchableBox;
    min (-0.05    -0.09707 0.01);
    max ( 0.68285    0.20793 0.03);
}
};

// Settings for the castellatedMesh generation.
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 100000;

    // Overall cell limit (approximately). Refinement will stop immediately
    // upon reaching this number so a refinement level might not complete.
    // Note that this is the number of cells before removing the part which
    // is not 'visible' from the keepPoint. The final number of cells might
    // actually be a lot less.
    maxGlobalCells 200000;

    // The surface refinement loop might spend lots iterations refining just a
    // few cells. This setting will cause refinement to stop if <= minimumRefine

```

```

// are selected for refinement. Note: it will at least do one iteration
// (unless the number of cells to refine is 0)
minRefinementCells 10;

// Allow a certain level of imbalance during refining
// (since balancing is quite expensive)
// Expressed as fraction of perfect balance (= overall number of cells /
// nProcs). 0=balance always.
maxLoadUnbalance 0.10;

// Number of buffer layers between different levels.
// 1 means normal 2:1 refinement restriction, larger means slower
// refinement.
nCellsBetweenLevels 2;

// Explicit feature edge refinement
// ~~~~~

// Specifies a level for any cell intersected by its edges.
// This is a featureEdgeMesh, read from constant/triSurface for now.
features
(
  {
    file      "rotor.eMesh";
    level     4;
  }

  {
    file      "blade1.eMesh";
    level     4;
  }

  {
    file      "blade2.eMesh";
    level     4;
  }

  {
    file      "blade3.eMesh";
    level     4;
  }
);

// Surface based refinement
// ~~~~~

// Specifies two levels for every surface. The first is the minimum level,
// every cell intersecting a surface gets refined up to the minimum level.
// The second level is the maximum level. Cells that 'see' multiple
// intersections where the intersections make an
// angle > resolveFeatureAngle get refined up to the maximum level.

refinementSurfaces
{
  blade1

```

```

{
    // Surface-wise min and max refinement level
    level (5 5);
    // Optional specification of patch type (default is wall). No
    // constraint types (cyclic, symmetry) etc. are allowed.
    patchInfo
    {
        type wall;
        inGroups (walls);
    }
}

blade2
{
    // Surface-wise min and max refinement level
    level (5 5);

    // Optional specification of patch type (default is wall). No
    // constraint types (cyclic, symmetry) etc. are allowed.
    patchInfo
    {
        type wall;
        inGroups (walls);
    }
}

blade3
{
    // Surface-wise min and max refinement level
    level (5 5);

    // Optional specification of patch type (default is wall). No
    // constraint types (cyclic, symmetry) etc. are allowed.
    patchInfo
    {
        type wall;
        inGroups (walls);
    }
}

rotor
{
    level      (4 4);
    cellZone   rot-ext;
    faceZone   rot-ext;
    cellZoneInside  inside;
}
}

// Resolve sharp angles
resolveFeatureAngle 20;

// Region-wise refinement
// ~~~~~

// Specifies refinement level for cells in relation to a surface. One of
// three modes

```



```

// - distance. 'levels' specifies per distance to the surface the
// wanted refinement level. The distances need to be specified in
// descending order.
// - inside. 'levels' is only one entry and only the level is used. All
// cells inside the surface get refined up to the level. The surface
// needs to be closed for this to be possible.
// - outside. Same but cells outside.

refinementRegions
{
    refinementRotor
    {
        mode inside;
        levels ((4 4));
    }

    refinementWake
    {
        mode inside;
        levels ((2 2));
    }

    refinementGeneral
    {
        mode inside;
        levels ((1 1));
    }

    blade1ref
    {
        mode inside;
        levels ((6 6));
    }

    blade2ref
    {
        mode inside;
        levels ((6 6));
    }

    blade3ref
    {
        mode inside;
        levels ((6 6));
    }
}

// Mesh selection
// ~~~~~

// After refinement patches get added for all refinementSurfaces and
// all cells intersecting the surfaces get put into these patches. The
// section reachable from the locationInMesh is kept.
// NOTE: This point should never be on a face, always inside a cell, even
// after refinement.
locationInMesh (-0.1 0.0 0.02);

```

```

    // Whether any faceZones (as specified in the refinementSurfaces)
    // are only on the boundary of corresponding cellZones or also allow
    // free-standing zone faces. Not used if there are no faceZones.
    allowFreeStandingZoneFaces true;
}

// Settings for the snapping.
snapControls
{
    //- Number of patch smoothing iterations before finding correspondence
    // to surface
    nSmoothPatch 3;

    //- Relative distance for points to be attracted by surface feature point
    // or edge. True distance is this factor times local
    // maximum edge length.
    tolerance 2.0;

    //- Number of mesh displacement relaxation iterations.
    nSolveIter 30;

    //- Maximum number of snapping relaxation iterations. Should stop
    // before upon reaching a correct mesh.
    nRelaxIter 5;

    // Feature snapping

    //- Number of feature edge snapping iterations.
    // Leave out altogether to disable.
    nFeatureSnapIter 10;

    //- Detect (geometric only) features by sampling the surface
    // (default=false).
    implicitFeatureSnap true; //false;

    //- Use castellatedMeshControls::features (default = true)
    explicitFeatureSnap true;

    //- Detect points on multiple surfaces (only for explicitFeatureSnap)
    multiRegionFeatureSnap true; //false;
}

// Settings for the layer addition.
addLayersControls
{
    // Are the thickness parameters below relative to the undistorted
    // size of the refined cell outside layer (true) or absolute sizes (false).
    relativeSizes false;

    // Per final patch (so not geometry!) the layer information
    layers
    {
        blade1
        {

```

```

        nSurfaceLayers 10;
    }

    blade2
    {
        nSurfaceLayers 10;
    }

    blade3
    {
        nSurfaceLayers 10;
    }
}

// Expansion factor for layer mesh
expansionRatio 1.12;

// Wanted thickness of final added cell layer. If multiple layers
// is the thickness of the layer furthest away from the wall.
// Relative to undistorted size of cell outside layer.
// See relativeSizes parameter.
firstLayerThickness 0.00002;

// Minimum thickness of cell layer. If for any reason layer
// cannot be above minThickness do not add layer.
// Relative to undistorted size of cell outside layer.
minThickness 0.00000005;

// If points get not extruded do nGrow layers of connected faces that are
// also not grown. This helps convergence of the layer addition process
// close to features.
// Note: changed(corrected) w.r.t 17x! (didn't do anything in 17x)
nGrow 0;

// Advanced settings

// When not to extrude surface. 0 is flat surface, 90 is when two faces
// are perpendicular
featureAngle 120;

// At non-patched sides allow mesh to slip if extrusion direction makes
// angle larger than slipFeatureAngle.
slipFeatureAngle 30;

// Maximum number of snapping relaxation iterations. Should stop
// before upon reaching a correct mesh.
nRelaxIter 3;

// Number of smoothing iterations of surface normals
nSmoothSurfaceNormals 1;

// Number of smoothing iterations of interior mesh movement direction
nSmoothNormals 3;

// Smooth layer thickness over surface patches
nSmoothThickness 10;

// Stop layer growth on highly warped cells
maxFaceThicknessRatio 0.5;

```

```

// Reduce layer growth where ratio thickness to medial
// distance is large
maxThicknessToMedialRatio 0.3;

// Angle used to pick up medial axis points
// Note: corrected w.r.t 17x! 90 degrees corresponds to 130 in 17x.
minMedianAxisAngle 90;

// Create buffer region for new layer terminations
nBufferCellsNoExtrude 0;

// Overall max number of layer addition iterations. The mesher will exit
// if it reaches this number of iterations; possibly with an illegal
// mesh.
nLayerIter 50;
}

// Generic mesh quality settings. At any undoable phase these determine
// where to undo.
meshQualityControls
{
    //- Maximum non-orthogonality allowed. Set to 180 to disable.
    maxNonOrtho 65;

    //- Max skewness allowed. Set to <0 to disable.
    maxBoundarySkewness 20;
    maxInternalSkewness 3;

    //- Max concaveness allowed. Is angle (in degrees) below which concavity
    // is allowed. 0 is straight face, <0 would be convex face.
    // Set to 180 to disable.
    maxConcave 80;

    //- Minimum pyramid volume. Is absolute volume of cell pyramid.
    // Set to a sensible fraction of the smallest cell volume expected.
    // Set to very negative number (e.g. -1E30) to disable.
    minVol -1E30; //1e-13;

    //- Minimum quality of the tet formed by the face-centre
    // and variable base point minimum decomposition triangles and
    // the cell centre. This has to be a positive number for tracking
    // to work. Set to very negative number (e.g. -1E30) to
    // disable.
    // <0 = inside out tet,
    // 0 = flat tet
    // 1 = regular tet
    minTetQuality -1E30; //1e-15;

    //- Minimum face area. Set to <0 to disable.
    minArea -1;

    //- Minimum face twist. Set to <-1 to disable. dot product of face normal
    // and face centre triangles normal
    minTwist -1; //0.02;

```

```

//- Minimum normalised cell determinant. This is the determinant of all
//- the areas of internal faces. It is a measure of how much of the
//- outside area of the cell is to other cells. The idea is that if all
//- outside faces of the cell are 'floating' (zeroGradient) the
//- 'fixedness' of the cell is determined by area of the internal faces.
//- 1 = hex, <= 0 = folded or flattened illegal cell
minDeterminant 0.001;

//- Relative position of face w.r.t. to cell centres (0.5 for orthogonal
//- mesh) (0 -> 0.5)
minFaceWeight 0.05;

//- Volume ratio of neighbouring cells (0 -> 1)
minVolRatio 0.01;

//- Per triangle normal compared to average normal. Like face twist
//- but now per (face-centre decomposition) triangle. Must be >0 for Fluent
//- compatibility
minTriangleTwist -1;

//- if >0 : preserve cells with all points on the surface if the
//- resulting volume after snapping (by approximation) is larger than
//- minVolCollapseRatio times old volume (i.e. not collapsed to flat cell).
//- If <0 : delete always.
//minVolCollapseRatio 0.1;

// Advanced

//- Number of error distribution iterations
nSmoothScale 4;
//- amount to scale back displacement at error points
errorReduction 0.75;
}

// Advanced

// Write flags
writeFlags
(
    scalarLevels
    layerSets
    layerFields // write volScalarField for layer coverage
);

// Merge tolerance. Is fraction of overall bounding box of initial mesh.
// Note: the write tolerance needs to be higher than this.
mergeTolerance 1e-6;

// ***** //

```

Finally, the *createBafflesDict* is used to create the AMI patches, required for the motion definition of the rotating region.

```

/*-----* C++ *-----*\
|=====|
| \\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox
| \\ / O p e r a t i o n | Version: 2.3.0
| \\ / A n d | Web: www.OpenFOAM.org
|  \\ M a n i p u l a t i o n |
\*-----*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       createBafflesDict;
}
// ***** //

// Whether to convert internal faces only (so leave boundary faces intact).
// This is only relevant if your face selection type can pick up boundary
// faces.
internalFacesOnly false;

// Baffles to create.
baffles
{
    rotor
    {
        //- Use predefined faceZone to select faces and orientation.
        type          faceZone;
        zoneName      rot-ext;

        patches
        {
            master
            {
                name          rot-master;
                type           cyclicAMI;
                inGroups      (cyclicAMI);
                matchTolerance 0.0001;
                neighbourPatch rot-slave;
                transform      noOrdering;
            }
            slave
            {
                name          rot-slave;
                type           cyclicAMI;
                inGroups      (cyclicAMI);
                matchTolerance 0.0001;
                neighbourPatch rot-master;
                transform      noOrdering;
            }
        }
    }
}
// ***** //

```

Dictionaries for the simulation settings definition

As for the simulation settings definition, the *fvSchemes* dictionary contains all the integration and interpolation schemes employed by the solver.

```
/*-----*- C++ -*-----*\
| ===== |
| \\ / F i e l d | OpenFOAM: The Open Source CFD Toolbox
| \\ / O p e r a t i o n | Version: 2.2.2
| \\ / A n d | Web: www.OpenFOAM.org
| \\ / M a n i p u l a t i o n |
\*-----*\
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "system";
    object       fvSchemes;
}
// ***** //

ddtSchemes
{
    default      CrankNicolson 0.9;
}

gradSchemes
{
    default      Gauss linear;
}

divSchemes
{
    default      none;
    div(phi,U)   Gauss linearUpwind grad(U);
    div(phi,k1)  Gauss linear;
    div(phi,kt)  Gauss linear;
    div(phi,omega) Gauss linear;
    div((nuEff*dev2(T(grad(U)))) Gauss linear;
}

laplacianSchemes
{
    default      Gauss linear corrected;
    laplacian(nuEff,U) Gauss linear corrected;
    laplacian((1|A(U)),p) Gauss linear corrected;
    laplacian(DomegaEff,omega) Gauss linear corrected;
    laplacian(DkEff,kt) Gauss linear corrected;
    laplacian(nu,k1) Gauss linear corrected;
}

interpolationSchemes
{
    default      linear;
    interpolate(U) linear;
}

```

```

snGradSchemes
{
    default          corrected;
}

fluxRequired
{
    default          no;
    p                ;
}

wallDist
{
    method meshWave ;
}

// ***** //

```

Finally, the *fvSolution* dictionary defines the solver's parameters employed for the resolution of the selected algorithm.

```

/*-----*- C++ -*-----*/
|=====|
|  \ \ /  /  F i e l d      | OpenFOAM: The Open Source CFD Toolbox
|  \ \ /  /  O p e r a t i o n | Version: 2.3.0
|  \ \ /  /  A n d             | Web:      www.OpenFOAM.org
|  \ \ /  /  M a n i p u l a t i o n |
/*-----*-*/
FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       fvSolution;
}
// ***** //

solvers
{
    p
    {
        solver          GAMG;
        tolerance       1e-05;
        relTol          0.05;
        smoother        GaussSeidel;
        cacheAgglomeration true;
        nCellsInCoarsestLevel 20;
        agglomerator     faceAreaPair;
        mergeLevels      1;
    }

    pFinal
    {
        $p;
        tolerance       1e-05;
        relTol          0;
    }
}

```



```

"(U|k|k1|kt|omega)"
{
    solver          smoothSolver;
    smoother        GaussSeidel;
    nSweeps         2;
    tolerance       1e-05;
    relTol          0.1;
}

"(U|k|k1|kt|omega)Final"
{
    $U
    tolerance       1e-05;
    relTol          0;
}
}

PIMPLE
{
    nOuterCorrectors 1;

    // NO < 65:
    nCorrectors      3;
    nNonOrthogonalCorrectors 0;

    // NO > 65:
    //nCorrectors     2;
    //nNonOrthogonalCorrectors 1;

    pRefCell         0;
    pRefValue        0;
}

// ***** //

```

References

- [1] IRENA (2018), “Renewable capacity statistics 2018”, International Renewable Energy Agency (IRENA), Abu Dhabi.
- [2] Bilgili M., Bilirgen H., Ozbek A., Ekinçi F., Demirdelen T., 2018, “The role of hydropower installations for sustainable energy development in Turkey and the world”.
- [3] C.M. Niebuhr, M. van Dijk, V.S. Neary, J.N. Bhagwan., 2019, “A review of hydrokinetic turbines and enhancement techniques for canal installations: Technology, applicability and potential”, Renewable and Sustainable Energy Reviews 113 (2019).
- [4] NREL, 2012, “Renewable electricity futures study”, https://www.nrel.gov/analysis/re_futures/.
- [5] Dinesh Kumar, Shibayan Sarkar., 2016, “A review on the technology, performance, design optimization, reliability, techno-economics and environmental impacts of hydrokinetic energy conversion systems”, Renewable and Sustainable Energy Reviews 58 (2016).
- [6] Kusakana K., Vermaak H. J., 2013, “Hydrokinetic power generation for rural electricity supply: case of South Africa”, Renew Energy 55 (2013).
- [7] M.J. Khan, G. Bhuyan, M.T. Iqbal, J.E. Quaiçoe., 2009, “Hydrokinetic energy conversion systems and assessment of horizontal and vertical axis turbines for river and tidal applications: A technology status review”, Applied Energy 86 (2009).
- [8] Verdant Power, “Free Flow System”, 2020, <https://www.verdantpower.com/free-flow-system>.
- [9] Instream Energy Systems, “Yakima Hydrokinetic Project”, August 2013, <https://www.instreamenergy.com/yakima-washington>.
- [10] Atlantisstrom Project, 2014. http://atlantisstrom.de/news_english.html.
- [11] HydroVenturi Ltd, 2018, “HydroVenturi,”.
- [12] Zotlöterer Plant, <http://www.zotloeterer.com/welcome/gravitation-water-vortex-power-plants/reference-plants/>.
- [13] Arnold Energy Systems, 2008, [Online].
- [14] Taylor George W., Burns Joseph R., Kammann Sean M., Powers William B., Welsh Thomas R., 2001, “The energy harvesting eel: a small subsurface ocean/river power generator”, IEEE J Ocean Eng 2001.
- [15] Vortex Hydro Energy, “VIVACE” Project <https://www.vortexhydroenergy.com/>.
- [16] BioPower System Ltd, “BPS Ocean Energy,” 2019, <https://www.bps.energy>.
- [17] Minesto, Deep Green Project, 2019, <https://www.minesto.com>.
- [18] US Department of Energy, Energy Efficiency and Renewable Energy, January 2006 , “Wind and Hydropower Technologies, Feasibility

- Assessment of the Water Energy Resources of the United States for New Low Power and Small Hydro Classes of Hydroelectric Plants”, Tech. Rep. DOE-ID-11263.
- [19] Khan MJ, Iqbal MT, Quaiocoe JE, 2008, “River current energy conversion systems: progress prospects and challenges”, *Renew Sustain Energy Rev* 2008.
- [20] Ponta F, Shankar Dutt G., 2000, “An improved vertical-axis water-current turbine incorporating a channelling device”, *Renew Energy* 2000.
- [21] Hannes Riegler, 2003, “HAWT versus VAWT”, REFOCUS.
- [22] Mertens Sander, Kuik Gijs van, Bussel Gerard van., 2003, “Performance of an H-Darrieus wind turbine in the skewed flow on a roof”, *J Solar Energy Eng ASME* 2003.
- [23] Radkey RL, Hibbs BD, 1981, “Definition of Cost Effective River Turbine Designs”, Tech. Rep. AV-FR-81/595 (DE82010972), Report for US Department of Energy, Aerovironment Inc., Pasadena, California.
- [24] Anyi M., 2013, “Water current energy for remote community: design and testing of a clog-free horizontal axis hydrokinetic turbine system”, PhD Thesis University of South Australia.
- [25] Anyi M, Kirke B., 2015, “Tests on a non-clogging hydrokinetic turbine”, *Energy Sustain Dev* 2015.
- [26] CADDET, 1988, “Water current turbines pump drinking water”, <http://www.caddet-re.org/assets/no83.pdf>.
- [27] Swenson WJ., 1996, “The specification, design and development of a kinetic energy tidal power generator”, The Darwin summit, Australia: National Engineering Conference.
- [28] Shannon R., 1996, “Waterwheel engineering”, Sixth international permaculture conference & convergence, perth, Australia.
- [29] Kumar A. Sachendra, 2017, “Development in augmented turbine technology”, *Int J Trends in Research Dev* 2017.
- [30] Elbatran AH, Yaakob OB, Yasser MA, Firdaus BA, 2015, “Augmented diffuser for horizontal Axis marine current turbine”, 1st international conference on innovation in science and technology, Kaula Lumpur, Malaysia.
- [31] Fleming CF, Willden RH, 2016, “Analysis of bi-directional ducted tidal turbine performance”, *Int J Mar Energy* 2016.
- [32] Belloni CSK, Willden RHJ, 2017, “An Investigation of ducted and Open-centre tidal turbines employing CFD embedded BEM”, *Renew Energy* 2017.
- [33] SHP Smart hydro power, 2016, “Munich: Smart Hydro Power”, <http://www.smart-hydro.de/decentralized-rural-electrification-projectsworldwide/nigeria-rural-electrification/#project>.

- [34] Gaden DLF, 2007, “An investigation of river kinetic turbines: performance enhancements, turbine modelling techniques and an assessment of turbulence models”, Master Thesis Winnipeg, Canada, University of Manitoba.
- [35] N. Parneix, R. Fuchs, A. Immas and F. Silvert, 2016, “Efficiency improvement of vertical-axis wind turbine with counter-rotating lay-out,” in *European Wind Energy Association*, Hamburg.
- [36] A. Vergaerde, T. De Troyer, J. Kluczevska-Bordier, N. Parneix, F. Silvert and M. C. Runacres, , 2018 , “Wind tunnel experiments of a pair of interacting vertical-axis wind turbines,” in *The Science of Making Torque from Wind (TORQUE 2018)*, Milano.
- [37] S. Zanforlin and N. Takafumi, , 2016, “Fluid dynamic mechanisms of enhanced power generation by closely spaced vertical axis wind turbines,” *Renewable Energy*, vol. 99, pp. 1213-1226.
- [38] Windside WS-4, 2020, <https://www.windside.com/products/ws-4>.
- [39] Padricelli C., 2019, “CFD investigation of counter rotating H-type Darrieus turbines in marine environment”, Master Thesis, Politecnico di Milano, Milano.
- [40] O. L. Hansen M., 2015, “Aerodynamics of Wind Turbines”, 3rd ed., Routledge, London, UK. Chap. 4.
- [41] Dixon S.L. and C.A. Hall, 2014, “Fluid Mechanics and Thermodynamics of Turbomachinery”, 7th ed., Elsevier Butterworth-Heinemann, Oxford, UK. Chap. 10.
- [42] Kirke B.K., Lazauskas L., 2011, “Limitations of fixed pitch Darrieus hydrokinetic turbines and the challenge of variable pitch”, *Renewable Energy* 36.
- [43] Claessens MC, 2006, “The design and testing of airfoils for application in small vertical axis wind turbines”, MS Thesis, TU Delft.
- [44] Kirke B.K., Lazauskas L., 1991, “Enhancing the performance of vertical axis wind turbine using a simple variable pitch system”, *Wind Eng*.
- [45] Baker JR, 1983, “Features to aid or enable self starting of fixed pitch low solidity vertical axis wind turbines”, *J Wind Eng Ind Aerodyn*.
- [46] Kirke B., 2019, “Hydrokinetic and ultra-low head turbines in rivers: A reality check”, *Energy for Sustainable Development* 52.
- [47] Persico G., 2018, “Lift-driven Darrieus turbines”, Politecnico di Milano, Milano.
- [48] Doan M. N., Alayeto I. H., Padricelli C., Obi S., Totsuka Y., 2018, “Experimental and computational fluid dynamic analysis of laboratory scaled counter-rotating cross-flow turbines in marine environment”, Joint US-European Fluids Engineering Summer Conference, Montreal.
- [49] Doan M. N., Alayeto I. H., Kumazawa K., Obi S., 2019, “Fluid dynamic analysis of a marine hydrokinetic crossflow turbine in low Reynolds number flow”, ASME – JSME – KSME Joint Fluids Engineering conference, San Francisco.

- [50] Ull A. R., 2012, “Study of mesh deformation features of an open source CFD package and application to a gear pump simulation”, MS Thesis, Universitat Politècnica de Catalunya, Barcelona.
- [51] Persico G., 2019, “Formulation of conservation laws”, Energy department, Politecnico di Milano, Milano.
- [52] Kundu P. K., Cohen I. M., Dowling D. R., 2015, “Fluid Mechanics”, 6th ed., Elsevier Academic Press, Cambridge, US. Chap. 12.
- [53] Walters D. K., Cokljat D., 2008, “A Three-Equation Eddy-Viscosity Model for Reynolds-Averaged Navier-Stokes Simulations of Transitional Flow”, *Journal of Fluids Engineering* 130.
- [54] Menter F. R., 1994, “Two-Equations Eddy-Viscosity Turbulence Models for Engineering Applications”, *AIAA Journal* 32.
- [55] Wilcox, D. C., 1988, “Reassessment of the Scale-Determining Equation for Advanced Turbulence Models”, *AIAA Journal*, Vol. 26, No. 11, pp. 1299-1310.
- [56] Klebanoff, P. S., 1971, “Effects of Free-Stream Turbulence on a Laminar Boundary Layer”, *Bull. Am. Phys. Soc.*, 16, p. 1323.
- [57] Mayle, R. E., Schulz A., 1997, “The Path to Predicting Bypass Transition”, *ASME J. Turbomach.*, 119, pp. 405–411.
- [58] Jacobs, R. G., and Durbin, P. A., 1998, “Shear Sheltering and the Continuous Spectrum of the Orr-Sommerfeld Equation”, *Phys. Fluids*, 10, pp. 2006–2011.
- [59] Fürst J., 2013, “Numerical simulation of transitional flows with laminar kinetic energy”, *Engineering Mechanics*, Vol. 20, No. 5, p. 379–388.
- [60] Lopez M., Walters D. K., 2017, “A Recommended Correction to the $k_T - k_L - \omega$ Transition-Sensitive Eddy-Viscosity Model”, *Journal of Fluids Engineering* 139.
- [61] Launder B. E., Spalding D. B., 1974, “The numerical computation of turbulent flows”, *Computer Methods in Applied Mechanics and Engineering* 3.
- [62] Johnson, D. A., and King, L. S., 1985, “Mathematically Simple Turbulence Closure Model for Attached and Separated Turbulent Boundary Layers”, *AIAA Journal*, Vol. 23, No. 11, pp. 1684-1692.
- [63] CFD Direct, 2020, “OpenFOAM user guide”, [Online] <https://cfd.direct/openfoam/user-guide/>.
- [64] Sheldahl RE, Klimas PC, 1981, “Aerodynamic characteristics of seven symmetrical airfoil sections through 180-degree angle of attack for use in aerodynamic analysis of vertical axis wind turbines”, Sandia National Laboratories Report SAND80-2114.
- [65] Selig MS, Guglielmo JJ, Broeren AP, Giguere P., 1995, “Summary of low-speed airfoil data, vol. 1”, Virginia Beach, Virginia: SoarTech Publications.