

MASTER THESIS
ENERGY FOR SMART CITIES, KICINNOENERGY

CFD analyses and performance
comparison of micro-hydropower

plants

Author: MARCO GRIPPA

Directors: PROF. A. SUMPER, DR. G. SLACHMUYLDERS

Call: SEPTEMBER 2016

ESCOLA TÈCNICA SUPERIOR

D'ENGINYERIA INDUSTRIAL DE BARCELONA



Abstract

The project concerns the hydropower renewable energy technology at its micro scale: thanks to an internship performed with the Belgian startup *TurbulentHydro*, the purpose of this Master Thesis is to evaluate the energy performance of 2 different models currently under investigation: the so-called Flatblades and Streamlines configurations. These layouts are similar in shape but different both in dimension and for the turbine used.

After a little introduction on renewable energies and hydropower technology, the selected CFD simulation procedure and all its options have been explained as well as the choice of the turbulence model to apply, computing the meaningful parameters to add in the model. This dissertation highlights the fluid dynamics behaviour by means of suitable softwares for this purpose: Autodesk Inventor, the 3D CAD mechanical design software in order to build, and edit when necessary, the geometries of the models considered, MeshMixer, a state-of-art software for locally adjusting the mesh of the starting model and OpenFOAM, a free and open source CFD program in order to run and evaluate any details of the analysis, simulating the operation conditions by means of its components and tools. Eventually, the most important CFD results are presented.

Different configurations bring different results. Regarding Flatblades model, the scope was refined the CFD initial setup in order to achieve results as close to the real case validation as possible whereas, for Streamlines model, an additional new component has been added for improving the current design in terms of energy output at the turbine level. At the end, conclusions stated possible refinements and improvements for these simulations as well as uncertainties arose from the results that might be avoided for the next stages.

Acknowledgements

I would like to express my personal gratitude to KIC Innoenergy and all its crew, I have been part of an amazing experience throughout the past 2 years in 3 different European countries and challenge myself in a really ambitious way, which makes me feeling like acting importantly in the sustainable and innovative path that anybody should follow and be direct to.

Special regards go to Geert and all Turbulent team for let me take this huge opportunity on taking part of this outstanding project they are developing and I am truly thankful to Prakriteesh for his endless patience.



Contents

Abstract	i
Acknowledgements	iii
List of Figures	x
List of Tables	xi
1 Introduction	1
1.1 Hydroelectricity, an overview	2
1.2 Turbulent concept and technology	5
2 Computational Fluid Dynamics	9
2.1 OpenFOAM overview	12
2.1.1 <i>Constant</i> folder	14
2.1.2 <i>System</i> folder	15
2.1.3 θ folder	17
2.2 Meshing	18
2.2.1 snappyHexMesh	20
2.3 Rotating region	22
2.4 Solvers: <i>simpleFOAM</i> and <i>interFoam</i>	22
2.5 Boundary conditions	23
2.5.1 Inlet/Outlet Velocities	23
2.5.2 Pressure field	24
3 Turbulence modelling	25
3.1 Theory	25

3.1.1	Near-wall treatment	26
3.1.2	Reynolds averaged equations (RANS)	27
3.1.3	Linear eddy viscosity model	30
3.2	SST k- ω model	30
3.3	Computation of the parameters	31
4	Turbulent models	37
4.1	Power calculations	41
4.2	Flatblades	44
4.2.1	Second simulation	47
4.2.2	Fourth simulation	49
4.2.3	Fifth simulation	52
4.2.4	Assessment of the results	55
4.3	Streamlines	58
4.3.1	Diffuser	60
4.3.2	First simulation	61
5	Conclusions	66
List of Symbols		70
References		73
A	Initial setup	74
B	Qualitative 2D CAD drawings	75
C	1st and 3rd Flatblades simulations	76
C.1	First simulation	76
C.2	Third simulation	78
D	Important commands on Ubuntu terminal	80
E	Other meaningful files for OpenFOAM	82
E.1	<i>transportProperties</i>	82
E.2	<i>snappyHexMeshDict</i>	83

E.3	<i>fvOptions</i>	86
E.4	<i>fvSchemes</i>	87
E.5	<i>blockMeshDict</i>	87
E.6	<i>controlDict</i>	88
E.7	<i>topoSet</i>	89
F	Autodesk CFD Simulations results	90
G	Model .stl files	92
H	Meshing and running simpleFoam in parallel with snappyHexMesh tool	94

List of Figures

1.1	World electricity production in 2014	2
1.2	Typical hydroelectric power plant	3
1.3	Run-of-the-river system based in Muzaffarabad area	5
1.4	Turbulent solution working system	6
1.5	On-site Kleerbeck plant	8
2.1	CFD procedure	12
2.2	Case directories in OpenFOAM	13
2.3	blockMesh benchmark	18
3.1	Water flow over a plate	26
3.2	Wall function	27
3.3	Velocity fluctuations	29
4.1	Origin and axis on the model	37
4.2	Patches of the model	38
4.3	paraView home	38
4.4	Heights in runner and diffuser zones	41
4.5	Flatblades model view	45
4.6	Turbines' blades for Flatblades	46
4.7	Flatblades meshing, particular views	46
4.8	Initial conditions for the 2 nd simulation	47
4.9	Velocities results for the 2 nd simulation	48
4.10	Pressures results for the 2 nd simulation	48
4.11	k and ω results for the 2 nd simulation	49
4.12	Forces and Torque values for the 2 nd simulation	49

4.13 <i>setFields</i> refined areas for the 4 th simulation	50
4.14 Turbulence parameters initial conditions for the 4 th simulation	50
4.15 Velocities results for the 4 th simulation	51
4.16 Pressures results for the 4 th simulation	51
4.17 k and ω results for the 4 th simulation	52
4.18 Forces and Torque values for the 4 th simulation	52
4.19 <i>setFields</i> average values refined for the 5 th simulation	52
4.20 Turbulence parameters initial conditions for the 5 th simulation	53
4.21 Velocities results for the 5 th simulation	53
4.22 Pressures results for the 5 th simulation	54
4.23 k and ω results for the 5 th simulation	54
4.24 Forces and Torque values for the 5 th simulation	55
4.25 Streamlines turbine	58
4.26 Streamlines model views	59
4.27 Streamlines meshing, particular views	60
4.28 Standard diffuser for Streamlines	61
4.29 Initial conditions for the simulation	62
4.30 <i>setFields</i> refined areas for the simulation	62
4.31 Turbulence parameters initial conditions for the simulation	63
4.32 Velocities results for the simulation	63
4.33 Pressures results for the simulation	64
4.34 k and ω results for the simulation	64
4.35 Forces and Torque values for the simulation	64

List of Tables

1.1	Pros and cons of hydroelectric power	4
1.2	Classification of hydroelectric power plants	4
1.3	Technical data	7
2.1	SI dimension units in OpenFOAM	14
2.2	snappyHexMesh procedure	21
2.3	Velocities boundary conditions	23
2.4	Pressures boundary conditions	24
3.1	Models to solve Reynolds stresses	29
3.2	Turbulence parameters' region	32
3.3	Turbulence parameters	33
3.4	Turbulence parameters results for Streamlines	34
3.5	Average values for k and ω for Streamlines	34
3.6	Turbulence parameters results for Flatblades	35
3.7	Average values for k and ω for Flatblades	35
4.1	Results analysis	40
4.2	Forces and Torques syntax results	41
4.3	Particular points in the models	42
4.4	Constant parameters for power calculations	44
4.5	Comparison between experimental data and simulations results	57
4.6	Power results for Flatblades	57

Chapter 1

Introduction

During the past decades, the power generation and its environmental impact arose as a global issue that any single country has to face, renewable energies are clean energy resources that have way lower or absent environmental impact than whichever traditional energy technologies. These sources will never run out, they are slowly and constantly taking over non-renewable energies such as oil or gas, even though conventional technologies are the most energy production used globally, e.g. coal source, which is the biggest air polluter, has still the highest share, 39% on the global electricity generation in 2014 [1]. A glance on the world electricity generation is issued in Figure 1.1.

A shift from fossil-fuels based to renewable energies technologies is necessary to face this problem worldwide.

Nowadays, various renewable energy technologies have been developed: the solar energy is coming directly from the sunlight, it can be used for different purposes such as heating or lighting buildings, for hot sanitary domestic water and, for generating electricity thanks to photovoltaics effect; this is surely the most known among renewable energies and it has a big potential to be exploited. Many efforts are undertaken to reach acceptable efficiencies, especially regarding PV modules.

Wind energy, captured by wind turbines with propeller-like blades, generates electricity according to where they are mounted and at which height they are operating.

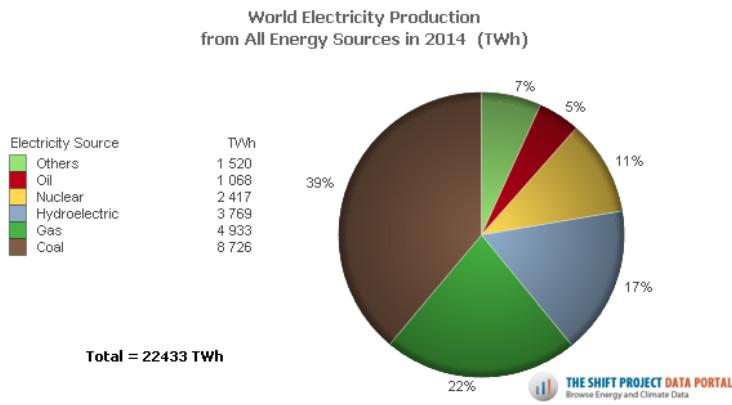


Figure 1.1: World electricity production in 2014 [1]

The last two technologies mentioned are pretty volatile, they strongly depend on the sun and wind availabilities at the place taken into account.

From organic matter, new technology can extract energy out of it, it is so-called biomass energy; it is quite recent compared to the others but it can be utilised for replacing transportation fuels, chemicals or produce electricity as well. Geothermal usage takes advantages from the internal Earth heat for disparate uses like air heating and/or cooling for building and electric power production; the most abundant element on the planet is hydrogen, which can be also used as an energy carriers, allowing transporting and storing energy. In Figure 1.1, it is easy to figure out the predominance of conventional energy generation technology in terms of shares.

One of the most impactful green energy is the electricity generated from water, better known as *hydroelectric* power. This technology is currently the most present one among renewable energies, it produces 17% of the total power generated globally. In 2014, EU27 countries produce around 349 TWh of electricity generation from hydroelectric power sources, which account for the 11% of the total electricity production, which makes, even in this case, in the most exploited renewable energy in the Old continent. [1]

1.1 Hydroelectricity, an overview

The hydroelectric power is an old method to produce energy but, especially during the XX century, it enhanced new knowledges and improvements that reach quite important role in the power generation benchmark; first power plants were installed in North America

and UK at the end of XX century and, afterward, this technology has been spread around the world.

The top 5 greatest market in terms of hydropower capacity are, respectively, China, Brazil, US, Russia and Canada but China has a far overcoming any other country with the amount of 249 GW; to complete the top 10, the list contains India, Norway, Japan, France and Turkey. Various countries account hydroelectricity in their land with a share of electricity generation for over 50% like Canada or Brazil but also the “outsiders” Mozambique and Nepal. [2]

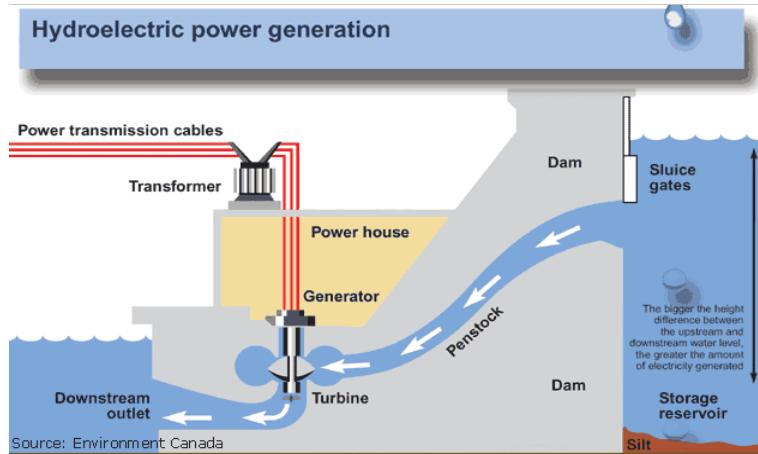


Figure 1.2: Typical hydroelectric power plant [3]

The working principle is quite simple, it produces electricity in a similar way as coal-fired power plants: a falling water from the reservoir, where it has potential energy thanks to the accumulated water, rush down at a penstock and it used to run a water turbine, which can be either an impulse or reaction type, striking and creating forces on the blades of it. Here the potential energy is transformed in mechanical energy. Consequently, it turns a metal shaft of an electric generator which is designed to produce current in stationary coils of wire and, thanks to power houses and transmission cables located into, the power generated is transferred to the electricity grid. [3]

In the Figure 1.2 a general layout of a dam hydroelectric power plant.

The capacity to generate energy is highly dependent on the height from which water falls and on the available flow, a suitable gate can adjust the mass flow rate according

to the energy needs by the grid network. The turbine configuration has to be chosen depending on the head and on the flow rate usable at the power plant, whereas the shape of the blades are designed in accordance with type of impeller selected and water pressures. Hydropower offers many advantages compared to other energy resources and, at the same time, tackling the environmental problems that fossil fuel-wise technologies largely have.

In Table 1.1 is shown the main pros and cons of hydroelectricity:

Advantages	<ul style="list-style-type: none"> • Renewable energy, rainfall renews water and no pollution created • Flexible, thanks to adjusting water inlet flow and power output • Reliable energy, very little fluctuations in electricity • Relatively low O&M costs
Disadvantages	<ul style="list-style-type: none"> • Dependent on local precipitations, droughts may affect the system <ul style="list-style-type: none"> • It can modify the fish habitat • High investment costs

Table 1.1: Pros and cons of hydroelectric power [3, 4]

One of the classification of this technology is made regarding the capacity size of the power plants, although the definitions vary a bit between each others, the Table 1.2 highlights the different categories:

Size	Capacity range
Large	> 50 MW
Medium	10 MW ÷ 50 MW
Small	0.1 MW ÷ 10 MW
Micro	< 0.1 MW

Table 1.2: Classification of hydroelectric power plants [5, 6]

Recently, the nano-hydroelectric power plants are released, it consists in applications of up to few kW but this technology is still under development and, so far, it has very little usage.

The main concern of this project is entirely focused on *micro-hydropower* plants. This technology, usually, does not have large storage reservoir but it exploits a portion of

moving water that is diverted from the standard river stream to a suitable channel or pressurised pipeline which link directly the river to the water turbine. This is the so called *run-of-the-river* system. The next picture shown an example of this system in Pakistan.

Various systems use an additional inverter in order to convert the low-voltage DC current provided from the generator into either 120 or 240 V of AC electricity current. As any other electricity generation system, it can be either off-grid or connected to electricity network.



Figure 1.3: Run-of-the-river system based in Muzaffarabad area [7]

Most of the installations are used by home or small business-owners because, even if they are small power plants, it is good enough to provide energy to run the entire energy appliances of a big house or a small resort [8]. In the next section, it has been analysed the concept of Turbulent and all the features concerning the innovation and sustainability behind it.

1.2 Turbulent concept and technology

TurbulentHydro is a start-up from Belgium found in January 2015. The team started prototyping vortex water turbine and all the turnkey based on that since summer 2014, after successes on hydro-power projects; it is located both in Belgium and Chile. It has been developed considering biomimicry, which is the imitation of system and elements of

the nature for the purpose of solving human issues.

The Turbulent solution proposes an affordable and reliable way to generate energy from water stream to even the most remote locations in the world, thus increasing the global electrification rate, in small scale and toward the path of *prosumer*.

This can be done thanks to a smart and decentralised energy production together with ease to install turnkey product. It generates electricity in the form of a single turbine or a network of multiple turbines. Turbulent aims both to sell single products and to collaborate with energy distributor and issuer, as well as installation companies to deliver the technology as simple as possible. Furthermore, the target is to electrify remote locations at which standard electricity grid is complicate either to install or to reach.

[9]

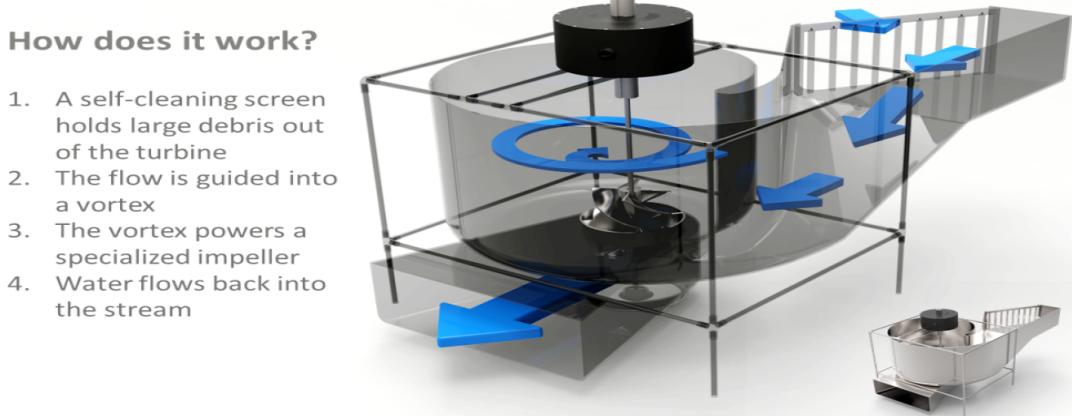


Figure 1.4: Turbulent solution working system [9]

This new technology is inspired by the natural shape of a vortex and enable to deliver affordable, renewable energy without damaging the local ecosystem. These hydropower plants range from 3 kW to 200 kW and are built using standard components combined with our specifically modelled propeller and control software.

The main idea is revolutionary: these hydro-power plants operate at a high efficiency, using the run-of-the-river technology, with a low height difference and, by combining few of them, it can produce the same amount of energy as a dam. The benefits are even more important, they are listed below [9]:

- **Fish friendly:** A low shear, low impeller RPM and low change of pressure make sure that fishes that get in the turbine, exit not harmed.

- **Low maintenance:** Self-cleaning trash rack and good quality components result in a quite low required maintenance.
- **Long operating life:** A solid design and rugged materials make sure that the turbine can easily achieve 20 years of regular production.
- **No flood risk:** The power plant works together with nature because it does not clog the flow of the river up.
- **Remote monitoring:** From anywhere at any time, the vortex is outfitted with a remote monitoring software.
- **Turnkey and mobile:** The vortex turbine is pre-manufactured, easily transportable and cost-effective as well as smallest in its type.

Turbulent took the design tools of a classic Kaplan water turbine and modified it to the extreme. The turbine has a low specific speed that can be made with a cheaper production process and lower tolerances. The main advantage of this modified turbine lies in its low heads range. This turbine produces 10 kW at 1.5 meters, and as the river conditions change (incl. height and flow), the runner efficiency diminishes by only a little amount due to a wide efficiency band. Another benefit it leaded from the low rotational speed of the turbine, which mean fish-friendliness by design. It has neither high shear speeds, nor large pressure changes, hence the additional benefit of allowing sediment and sand to pass without being abraded. All over the large scale design of 10 to 30 kW is easy to install and eco-friendly, while at the same time producing clean energy for the grid, or for the remote communities that need it.

Electrical power	Height difference	Yearly energy production
2.25 kW	2 m	12 ÷ 15 MWh
Efficiency	Flow	Yearly saved energy costs
55 %	230 lps	2400 ÷ 3000 €

Table 1.3: Technical data [9]



Figure 1.5: On-site Kleerbeck plant

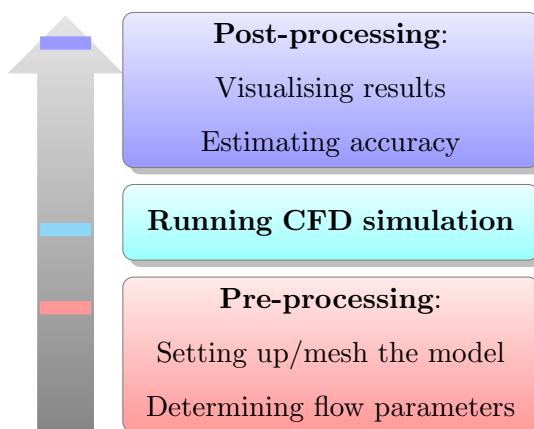
Most of the research team is developing and mapping potential sites in Chile, although the Turbulent pilot site is in Kleerbeck, Belgium; this is the first sample of an installed Turbulent product, the first real tests of a Turbulent power plant have been performed here both from the mechanical and electronic sides, the set-up is depicted in the Figure 1.5 and the features of the plant are shown in the Table 1.3.

Chapter 2

Computational Fluid Dynamics

The computational fluid dynamics, hereafter called CFD, is an interdisciplinary matter between numerical analysis, fluid mechanics and computer science in order to watch not only how a liquid or a gas streams flows but also how those affect the surrounded environment.

The values and parameters took into account are, usually, velocity, density, heat transfer, velocity and pressure and the theory behind it is based in Navier-Stokes equations. The latter rules the fluids motion and it can be considered as the Second Newtonian Law of motion for fluids, it also concerns the conservation of momentum and mass, together with the continuity equation [10]. It is pivotal in this analysis because, by solving these equations for a set of boundary conditions as elucidated in the next paragraphs, it predicts fluid characteristics for a given geometry.



The softwares involved in the analysis require informations about content, size and layout of the control volume considered; they are used for realising a 3D mathematical model on a mesh that can be handled by view and rotated from various perspective. The main 3 steps of the analysis are indicated in the above diagram.

First of all, it needs to decide which kind of simulation to perform, there are various types of [11]:

- **2D, quasi 3D or 3D:** 2D simulation is used for initial design phase such as section of a vane or blades, quasi 3-D simulation is an upgrade of the 2D where additional terms about the acceleration/deceleration caused by whatever cause are considered, many CFD codes need to activate this feature since they require hidden commands. 3D is the most complete one, at which also secondary flows are obtained correctly, it works very well in case of boundary layers grow very fast and interact with a huge part of flow field.
- **Viscid or inviscid:** an inviscid Euler simulation is suitable for flows close to the design point and it does not take into account boundary layer whereas inviscid one is necessary to predict losses and separation; nowadays, the latter configuration does not require so much time as back in time.
- **Transient or stationary:** transient simulations are performed when it is needed to know the behaviour of a transient flow which has a strong influence on the entire flow field; else, a stationary simulation fits better the case.

In this project, stationary 3D simulations are implemented, as it can be seen in the next chapters. By changing variables and parameters, the user can set and visualise how water stream clashes the blades of a turbine under a number of different conditions, for example.

There are different softwares whose run CFD simulations. During the initial period of Turbulent experience, it has been adopted a programme called *FlowSimulation*, add-in of the basic SolidWorks, which allow a successful performance to simulate gas and liquid flows in real conditions, either for internal or external analysis. Unfortunately, in FlowSimulation, it is not possible to perform a free-surface simulation, which is the involvement of air/water interface, a key point for the analyses needed in this project. Hence, when this characteristic

was not possible to apply, a migration to another software was needed. The second software used to perform CFD analysis was *Autodesk CFD Simulation*, this software allows to make free-surface simulation, analogous to FlowSimulation, they have many similarity like performing the 3 steps of the CFD analysis (drawing, meshing and simulation) in the same in-built software which take much short time to setup the simulation, compared to other softwares.

On the other hand, they do not allow to change all the variables involved in the theory, especially concerning the turbulence model, the most important topics to reach result as close to the experimental data as possible, especially regarding the first configuration undertaken. Beside that, the results from this software were quite good and consistent with the prediction, except for the specific value about the torque applied on the runner which is a truly meaningful value for the analysis and, after a long period to figure out why that value was wrong, also this software has been discarded for the project.

Finally, after quite some weeks, it has been decided to perform the CFD simulations with the most powerful software available online: *OpenFOAM*. This software is quite different to the ones just mentioned because it is working on C++ environment as base language. To do so, it has been needed some time to understand how this language works in Ubuntu, especially using the terminal of the new operating system and, in turn, it took quite long time to learn how to use that. That is why in the Appendix D are highlighted the main commands to know in order to work properly under this language and environment.

In the Figure 2.1, the entire final procedure to make the CFD analysis is illustrated.

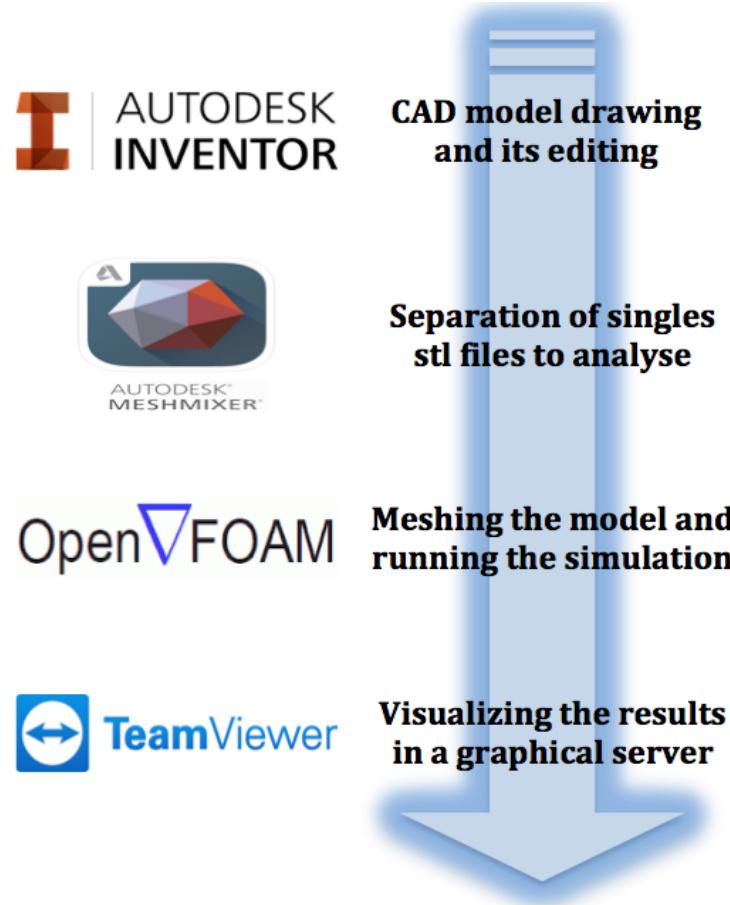


Figure 2.1: CFD procedure

2.1 OpenFOAM overview

OpenFOAM is an open-source, free and state-of-art CFD software since 2004. It has a huge customer base across most areas of engineering and science, from both commercial and academic organisations.



OpenFOAM has a wide range of features to solve even very complex fluid flows involving turbulence, heat transfer and chemical reactions to solid mechanics, acoustics and magnetics. Quality performance is based on rigorous testing. The code evaluation, verification and validation process includes various daily unit tests, test battery on a weekly basis, and large industry test battery anytime a new version releases. Tests are designed to assess regression behaviour, scalability, code performance and memory usage. It is a collection of nearly 250 applications built on another collection

of more than 100 software libraries. [12]

The version used in the project is 2.04 OpenFOAM over 14.04.5 Ubuntu operating system. The User Guide examines any single utilities enabled to use, various reader and write data modules. Its releases are scheduled every six months in June and December and it works only in the Ubuntu ambient, so a Linux partition, by a free virtual machine additionally installed, has been addressed on the local computer used. [13]

The original structure of any cases ran in OpenFOAM are shown in the Figure 2.2:

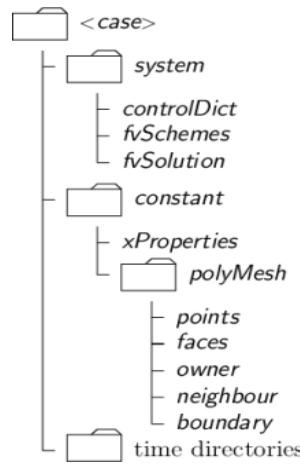


Figure 2.2: Case directories in OpenFOAM

The mandatory three directories, those had to be implemented, are *system*, *constant* and *time directories*, the latter usually contain the initial time step at 0 sec, as it can be seen in the next chapters and the name is based on the simulated time taken into account, so normally, the initial time directory is called *0*.

For both models considered in the project, the directories are unchanged, even though their contents vary according to the purpose, geometry and many other variables. Setting up these folders is part of the pre-processing procedure. An example of how these files look like, it delineated in Appendix A. The unit dimensions are expressed in this form:

$$[0 \ 1 \ -2 \ 0 \ 0 \ 0]$$

Number	Property	SI unit
1	Mass	kilogram (kg)
2	Length	metre (m)
3	Time	second (s)
4	Temperature	kelvin (K)
5	Quantity	mole (mol)
6	Current	ampere (A)
7	Luminous intensity	candela (cd)

Table 2.1: SI dimension units in OpenFOAM

In the example written above, the numbers represent the position within the brackets and it is the the unit of measure of acceleration m/s^2 .

2.1.1 *Constant* folder

This directory regards all the constant values by which the thorough case description is made as well as specifying the physical quantities, concerning the application in case.

The sub-folder *triSurface* contains all the .stl files of the geometry taken into account, throughout the various setups, it has been made files named as the separated patch considered as shown in Appendix G, the number of stls files vary according to the configuration. The sub-folder *polyMesh* is necessary to connect and link cells and boundary faces around cells and boundary patches, each face refers to the owner and the neighbour cells; by doing so, the connectivity is arranged. After this, it listed the points, faces, owner and neighbour cells and boundaries. This folder contains *blockMeshDict* which specify the vertices, blocks, edges and boundaries within the model thanks to blockMesh utility explained in Appendix E.5 as well as in Section 2.2.

Others files present in this folder are [13]:

- *dynamicMeshDict*: it is the dictionary for solving systems thanks to moving meshes, it acts during the simulations itself, it controls morphing and deformation of the mesh. The dictionary allows to specify four types of mesh motion: it has chosen the staticFvMesh one which basically has no mesh motion, so the mesh is fixed as

static; this utility varies a lot according to the version of OpenFOAM.

- *g*: it is to define the gravity function and its value, it has clearly been chosen the constant gravitational acceleration across the domain.
- *turbulenceProperties*: it selected for any case which include turbulence modelling, the choice can be between 3 different kinds of modelling: laminar, LES or RAS. It chose the last one for the analysis.
- *RASProperties*: it specifies which Reynolds-averaged stress turbulence model to apply (name and switch it on/off) and to whether print the coefficient or not, it enables the turbulence model to do. Optional features of the dictionary are not taken into account within the project.
- *transportProperties*: it defines the density and viscosity for the all means of transport to consider, generally, water and air and, even in this case, the standard values has been constantly set in this file as outlined in Appendix E.1 .

2.1.2 *System* folder

The directory *system* contains the parameters linked to the procedure to get the solution itself. The files contained in this folder for this project purposes are the following:

- *controlDict*: it defines the data concerning the time control as well as the reading and writing of the solution. Time steps, Start/End time, format and precision, Courant number are set here, OpenFOAM provides great flexibility, as it partially reproduced in the Appendix E.6; it also regards which useful parameters for the analysis to print at the end of the simulation; in this case, the forces acting on the blades faces by water stream passing through the runner and, eventually the torque; within its features, it is also established the writing control of the steps: adjustableRunTime has been stated.
- *decomposeParDict*: it uses to decompose mesh and fields. The objective is to break the entire domain, accordingly to certain parameters and specified in this dictionary; there are different methods of decomposition, the user has to choose between simple, hierarchical, manual and scotch as well as define the number of subdomains (of CPU cores) to have. Simple mode is the one selected in the dictionary.



- *fvOptions*: throughout this project, fvOptions is used to define the exact region of rotation for the runner, the velocity in magnitude and direction, at what time it starts and how long it lasts. An example of this file is in Appendix E.3.
- *fvScheme*: it discretises, by schemes, the solution during the run-time. It sets the numerical schemes for terms like derivatives in system of equations as well as gradient or laplacian operators. The items to define have to be chosen among the common numerical models: Euler, Gauss linear, Gauss upwind, etc. The file is in Appendix E.4.
- *fvSolution*: over here, tolerances, relaxation factors, algorithms and equations solvers are controlled; it contains various sub-dictionaries, such as solver values to run the simulation; for pressures PCG with GAMG preconditioner and for U, k and omega the smoothSolver has chose.
- *meshQualityDict*: it gets back about the mesh, only controlling some feature like maximum non-orthogonality allowed, described in snappyHexMeshDict. It includes defaults values from master dictionary.
- *setFieldsDict*: This is done in order to set initial condition within the domain, in case a non-uniform ones it would be set up here; as it sees later on, this is a quite important file because it allows to specify the turbulence parameters and the initial condition for the height of the water initially within the configuration. This file varies for any simulation.
- *surfaceFeatureExtractDict*: it extracts the edge features and save it to a new .eMesh files located in the same subdirectory. The extraction method is selected as from surface.
- *topoSetDict*: it performs to enclose the rotating frame for the runner, it sets the geometrical area to consider for the rotation, specifying in this project, the cylinder source for the rotational region, as it has been highlighted in the Appendix E.7.

Furthermore, other 2 files: *snappyHexMeshDict* and *snappyHexMeshDict.org* are present in this folder, they are explained in Section 2.2.1 as well as in Appendix E.2.

2.1.3 *0* folder

The time directory gets the single files for any specific fields like pressure or velocity; the data might be initial values or boundary conditions that user should specify for the simulation or results written by OpenFOAM itself. The following files are involved in this folder:

- *alpha.water*: it defines the fraction of water and air within the volume as initial condition. It is very important file especially for the first 3 Flatblades simulations, because the first parameters needed to be calculated was the initial height of the water within the volume. Furthermore, the scalar value alpha was set, punctually:

$\alpha = 0$, the cell is considered 100% of air fraction

$\alpha = 1$, the cell is considered 100% of water fraction

- *k*: it is part of the turbulence model. It is the turbulent kinetic energy, computed in Chapter 3. The boundary field sets this amount for any single .stl files defined in *constant/triSurface*.
- *nut*: this file concerns the choice for the wall function specified via viscosity value for the turbulence model and enables to apply different walls functions to different walls regions.
- *omega*: alike *k* regards the parameters for the turbulence model for any .stl files, ω is the specific dissipation rate and it is calculated in Chapter 3.
- *pgh*: it is defined by the equation

$$p - \rho gh$$

which is the dynamic pressure calculated as the static pressure minus the hydrostatic one.

- *U*: it sets the velocity onto inlet patch as the initial condition, important for the incoming flow in the model.

An important consideration about this folder is that these initial conditions, also the ones regarding *k* and ω parameters are changed during the simulation and, referring to that, the default parameters set in *system/setFields* are overwriting these initial conditions as well as for the initial water height.



2.2 Meshing

One of the most important practise to perform CFD analysis as best as possible is surely the meshing of the model. It is an integral part of the numerical solution and have to satisfy certain criteria to observe an accurate and valid solution. Fluid flows, expect for very simple cases, are not suitable for analytic solution, and it makes the analyses harder from the mathematical point of view. Therefore, the entire volume is used to be split into smaller subdomains thanks to geometric shapes applied for that purpose, it might be 3D ones such as hexahedral or tetrahedral one. By doing so, any single amount of the governing fluid equations is discretised and solved within these subdomains. [14]

It should focus the attention on the proper continuity of undergo solution along the portions nearby, so that the approximate solution can be put together to make a exhaustive picture of the fluid flow within the whole domain. As it analysed in this section, usually these subdomains are called *cells*. Various computational and solver techniques have been released recently, refining mesh elements and their connectivity more and more. Many softwares have automatic meshing tool (e.g FlowSimulation or Autodesk CFD Simulation), which can be way faster than using manual meshing, but it would work very well only if the geometry is not very complex or the flow condition are not so complicated. Structured meshes are regular connectivity characterised, it expressed as either a 2 or 3 dimensional array, it makes the mesh conserving space because of a storage arrangement relationship defined on adjacent cells. Hence, multi-block hexahedral structured mesh is the most often used in this case [15].

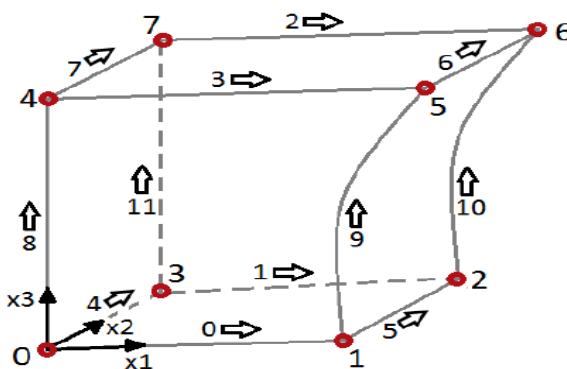


Figure 2.3: blockMesh benchmark [12]

It is tough to define the mesh size in a common frame and, basically, it depends on the purpose of the simulation. OpenFOAM provides different meshing tools: by default, it uses a polyhedral cells in 3-D polygonal faces. This mesh within the software is called *polyMesh*, it enables to generate a huge freedom in meshing and handling, especially, complex geometry or changing over time [12]. In the next list, it highlighted the specifications for the mesh:

- *Point*: location in a 3D space, it is defined by a vector. They are listed and referred to by a label
- *Faces*: it is a list of ordered points, 2 points are linked by an edge, also faces are listed as well as referred to by labels; the direction took into account is defined by right-hand rule. They can be internal faces (which connect 2 cells) or boundary faces (of whom one cell coincide with the boundary domain)
- *Cells*: list of faces in casual order, they must entirely cover the computational domain, convex, closed and its cell center inside the cell.
- *Boundary*: list of patches, of which everyone is associated with a boundary condition.

OpenFOAM provides 2 main utilities to produce a mesh for models: *blockMesh* and *snappyHexMesh*; *blockMesh* is suggested for creating a simple hexahedral structured block mesh, each block consists in 12 edges and 8 vertices in a specified order, as shown in the Figure 2.3. The theory behind this first tool is pretty smooth: it defines the vertices first, then secondly, edges, blocks and patches. The vertices are listed in such a way that each of them can be mentioned by remarking its label, edges can be splines, polylines, arcs or straight lines. Each block is referred to a local coordinate system and 2 vertices are linked with either straight or spline edge lines, it is right-handed and it decompose the entire geometry in multidimensional hexahedral blocks. The blocks are defined as follows:

- Vertex numbering: the blocks are always hexahedra and the vertices are listed and ordered in a certain manner.
- Number of cells: this entry gives the number of cells for each x, y, z direction.
- Cells expansion ratio: it is a useful tool, it gives the expansion ratio for each x, y, z direction enabling refinement in a specified axes if needed; the one used

simpleGrading underlines uniform expansions along the three axes and, the same order of magnitude between them.

The boundary of the mesh is listed under blocks, this boundary is broken into various patches, but for this purpose, only the big bounding box is indicate in this file because another tool of OpenFOAM is used. Hence, the boundary *allBoundary* is written and it includes all the outer patches of the suitable model for the simulation. The *blockMeshDict* file is shown in Appendix E.5.

Nonetheless , in this project, snappyHexMesh is used because of its functionality and, as an initial starting point, the blockMesh tool as described in the next subsection.

2.2.1 snappyHexMesh

It is a 3D meshes including hexahedra and spit-hexahedra from triangulated surface or tri-surface geometries in STL (Stereolithography) format; the procedure is iterative: it constantly refines the initial mesh; the biggest advantage is the flexibility for the mesh refinement throughout the model as well as a pre-specified final mesh. Load balancing steps run in parallel with the mesh generation.

This dictionary is explained in details in Appendix E.2. In Table 2.2, the entire procedure of meshing is resumed and explained, taking into account the OpenFOAM user guide [12].

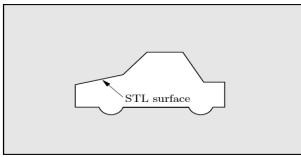
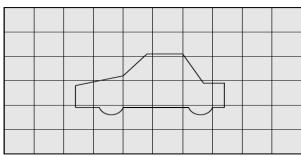
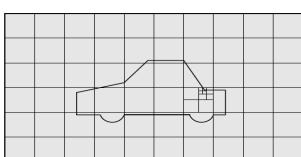
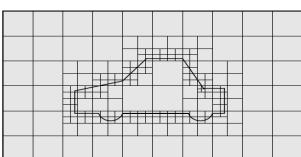
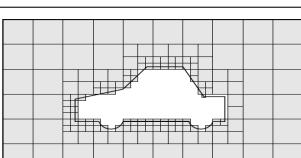
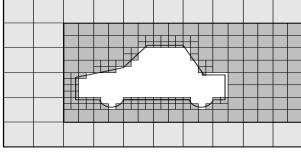
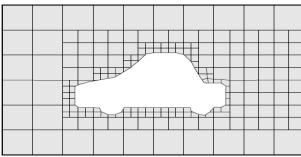
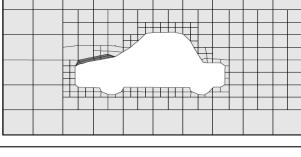
Step 1	Initial STL surface		The STL object is uploaded in OpenFOAM, the entire volume is depicted as the grey rectangular zone.
Step 2	Creation of background mesh		blockMesh enables to generate an initial mesh with an cell size ratio of around 1.
Step 3	Splitting of cells by feature edge		Feature edge is present in the castellatedMeshControls sub-dictionary, it spots the edgeMesh file and level of refinement.
Step 4	Splitting of cells by surface		In the same dictionary, the cells are selected for splitting in the place of specified surfaces, it requires a min and max level of refinement.
Step 5	Removing of cells		One or more regions within the bonding domain addresses the cells to remove.
Step 6	Splitting of cells by region		The refinement region sub-dictionary leads a further splitting in one or more specified volume region.
Step 7	Snapping surface		It moves cell vertex points onto surface geometry to delete jagged castellated surface from the mesh.
Step 8	Layer addition		It introduces additional layers of hexahedral cells aligned to the boundary surface.

Table 2.2: snappyHexMesh procedure [12]

Eventually, the *meshQualityControls* sub-dictionary controls the quality of the mesh just explained and, by the final balancing, it is possible to spot if some mesh irregularity or flaws are present in the mesh.

2.3 Rotating region

The rotating region is a truly important zone that has to be set properly within the control volume, it concerns where the turbine is, more precisely from the hole of the basin bottom to the output of the cylindrical casing, passing through the runner.

First of all, it is fundamental to choose the right path to set a rotating region in OpenFOAM: in this case, steady time dependency and stationary and rotating regions in the computational domain, the file *topoSet* in combination with *fvOptions* have been chosen, they enable to use multiple reference frame model and the file *fvOptions* is taken into consideration in the Appendix E.3. Not to weigh on the dissertation, the governing equations at the rotating region are not highlighted in the project.

2.4 Solvers: *simpleFOAM* and *interFoam*

In OpenFOAM, there are various models in order to solve RANS equations; in this project, it has been decided to use *simpleFoam*. It used the SIMPLE algorithm, which stand for Semi-Implicit Method for Pressure Linked Equations, this algorithm is iterative and this solver takes into account the mass and momentum conservation as depicted in the following equations:

$$\nabla \cdot \mathbf{u} = 0 \quad (2.1)$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \nu \nabla^2 \mathbf{u} - \nabla p + \mathbf{g} \quad (2.2)$$

simpleFoam is the most appropriate model for this case because it is a steady-state solver for turbulent and incompressible flow, it has been used for both laminar and turbulent zones.

Whereas, *interFoam* is chosen for the time step control, it provides automatic adjustment of time steps, specifying the valuable parameters such as time start, time step, writing

time and so on. The steps are not approximated to a convenient value hence, they are saved in somewhat arbitrary mode. The same solver utilises the OpenFOAM created limiter called MULES (Multidimensional Universal Limiter for Explicit Solution) for discretisation schemes to keep the phase fraction bounded independent from numerical scheme, mesh structures, etc. [12]

2.5 Boundary conditions

The boundary conditions are set in the θ folder. In OpenFOAM there are 2 kinds of boundary conditions: basic and derived. The basic ones are specified in suitable entries and it varies from simple to complex ones, whereas the derived are functions of the basic conditions. For any simulation considered throughout the project, these parameters are unchanged. Separately, there are other files in θ folder, regarding k and ω parameters as well as water height, these quantities are varying according to the simulation taken into consideration.

They have been explained in Section 2.1.3 and as it outlined thereafter; these values are defined in other directories.

2.5.1 Inlet/Outlet Velocities

Regarding the velocity field, the U file contains all the requirements for it. Practically, the values to add are in the Table 2.3. In all the other patches, this file sets 0 as values, *inletOutlet* type allows the flow to get out from the top patch, for instance in an extreme case of overflow which it never encountered during any simulations.

Patches	Property
Inlet	<i>flowRateInletVelocity</i> type Mass flow rate = 9 kg/s and perpendicular to the surface
Outlet	<i>inletOutlet</i> type and <i>internalField</i> value
Top	<i>inletOutlet</i> type and <i>internalField</i> value

Table 2.3: Velocities boundary conditions

2.5.2 Pressure field

Similarly for this condition, only the three interface model/air are involved in it. This configuration in the file p_{rgh} . In this condition the inlet patch is not important as it strongly depends on the inflow. In the table it is highlighted how to set it:

Patches	Property
Inlet	<i>fixedFluxPressure</i> type and <i>internalField</i> value
Outlet	<i>fixedValue</i> type and <i>uniform 0</i> value
Top	<i>fixedValue</i> type and <i>uniform 0</i> value

Table 2.4: Pressures boundary conditions

As it can be seen in the above table, as boundary conditions, outlet and top patches are set at null value which means that the pressure in those zones is equal to the environment one and, concerning the inlet patch, it is deriving from the internal simulation.

Velocity and pressure field are related between each others.

Chapter 3

Turbulence modelling

Almost every environmental stream in nature, either water or air, are turbulent. The phenomenon is quite irregular and highly intermittent to be analysed, even nowadays there is no common theory about both the turbulence theory and its statistical properties, so a holistic and empirical approach to this concept is required.

Various model has been developed throughout the decades and, in next sections, the most used ones are explained and, especially for the one chosen to undertake the project, is highlighted more in details.

3.1 Theory

With the limits of continuum as initial hypothesis, the Navier-Stokes equation has been the one that govern, almost universally, the physics of all fluid stream, turbulent ones too.

First of all, turbulence occurs at high Reynolds number because of the sophisticated mixture of inertia and viscous terms in the momentum equations. Reynolds was the first one facing the problem and investigating the transition from laminar to turbulent flow and he identified a single dimensionless parameter, the so-called *Reynolds number*:

$$Re = \frac{U\rho L}{\mu} \quad (3.1)$$

The initial assumption is that the fluid taken into account is Newtonian, usually true for many fluids in the field of engineering applications, which means that the viscosity is



constant regardless the shear rate. [16]

There are two characteristic value of the Reynolds number which can identify the approach of the flow from the laminar to transient and, in turn, from transient to turbulent; so if the it is below 2300, the flow is laminar and when it is over 4000, the flow is turbulent, in the range between those values, the flow is in transition between the two types.

In the Figure 3.1 it is clearly shown what happens to the flow over a flat plate:

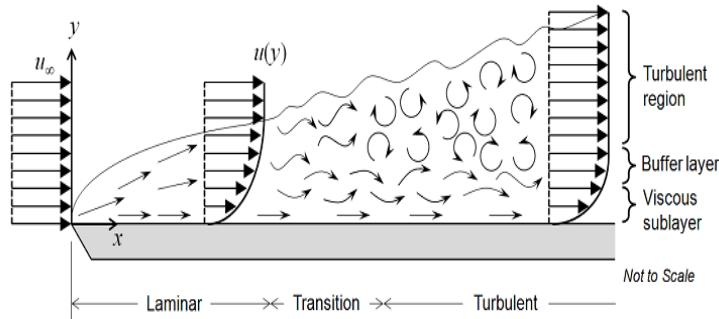


Figure 3.1: Water flow over a plate [17]

In the laminar range, at the inlet of the plate, the behaviour is quite straightforward: the flow can be solved by a steady-state Navier-Stokes equations, which predict both pressure and velocity fields; the latter does not change with time in this area, so the prediction is quite accurate. The phenomenon is way more complicate when the transition toward turbulence range starts: chaotic oscillations appear and the flow is varying with time in this field. Time domain consideration and fine mesh of small eddies flow must be taken into account and Navier-Stokes approach is not feasible anymore; on the other hand, in this range, it is possible to use Reynolds-average Navier-Stokes (RANS) approach, which is relying on the separation of the entire flow field over time in small local oscillation and time-averaged field but, by doing so, it introduces further unknowns to our system of equations. Fluctuations are involving also pressure and temperature amounts [17]. It is explained in Section 3.1.2.

3.1.1 Near-wall treatment

Near-wall treatments are used to model the velocity profile at the first node of the wall. The model is based on magnitudes such as k , ϵ , ν , τ_w , wall shear stress, friction velocity and so on. Therefore, the implemented function, provide the computation of

the law of wall. *Wall functions* are necessary to develop a proper turbulence model that perform well at the near-wall area, the range can be split in 4 different regimes as shown in the Figure 3.1. These functions are also employed in the tool used by OpenFOAM to mesh the near-wall zones on the model, in Appendix E.2 under *addLayersControls*, the wall functions are added.

The first region is called *viscous sublayer*, in this region and in correspondence of the wall, velocity is null and for the thin layer right above, the velocity is linearly varying with distances. Further up, the stream starts the transition to turbulent and this zone is called *buffer layer* (both regions are very small, approximately one hundredth of the others), which can be approximated because of its thickness. When the stream is completely turbulent developed, the average flow speed is proportional to the log of the wall distance and this third region is called *log-law region*. Further away from the wall, the free-stream region is present, which is the region not affected from the wall function. [17]

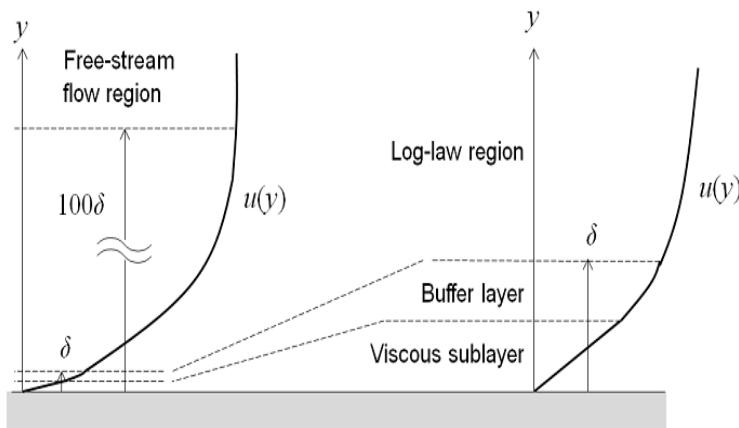


Figure 3.2: Wall function

For all these sectors, the RANS model can be computed, which calculate a non-zero velocity at the wall, ignoring the flow field in buffer region and assuming an analytic solution at the viscous sublayer. By doing so, it results in highly lower calculation needs. If a higher level of accuracy for the wall function is needed, then complementary turbulence model would be used.

3.1.2 Reynolds averaged equations (RANS)

All the forces acting on each element surface are separated in 2 different types:

- Long range force: It acts on the fluid element mass and distributed over the volume, so force per mass unit.
- Short term force: It acts only on the surface of the fluid element, whose total force exerted is proportional to the surface area

For sake of simplicity, the whole procedure of equations governing instantaneous fluid motion is not explained. As discussed before, turbulence theory is not very simple and it is easier to go through more linear parameters to evaluate, such as turbulence kinetic energy, dissipation rates, intensity and so on.

The starting point is from incompressible momentum equation, illustrated in Equation 3.2:

$$\rho \left[\frac{\partial \tilde{u}_i}{\partial t} + \tilde{u}_j \frac{\partial \tilde{u}_i}{\partial x_j} \right] = - \frac{\partial \tilde{p}}{\partial x_i} + \frac{\partial \tilde{T}_{ij}}{\partial x_j} \quad (3.2)$$

The viscous stress, which is the stress minus the average normal stress, is represented by the tensor with the initial hypothesis of incompressible fluid: [17]

$$T_{ij} = 2\mu \tilde{s}_{ij} \quad (3.3)$$

Taking into account these considerations and the above equations, it used now the Reynolds decomposition, which is the analysis of the flow in 2 parts: the mean motion, represented in capital letter and the fluctuating motions, represented in lowercases.

$$\tilde{u}_i = U_i + u_i \quad (3.4)$$

$$\tilde{p} = P + p \quad (3.5)$$

$$\tilde{T}_{ij} = T_{ij} + \tau_{ij} \quad (3.6)$$

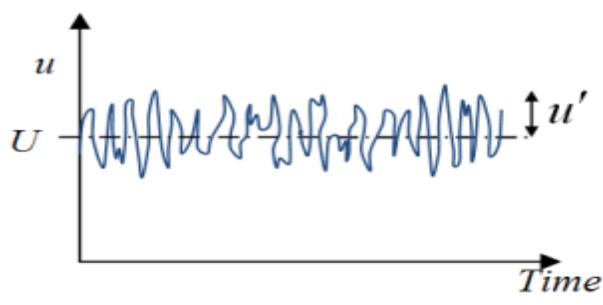


Figure 3.3: Velocity fluctuations

In the Figure 3.3 the fluctuations of velocity, for instance, is reproduced; similar consideration are for the other 2 equations written above.

So substituting the decomposition into Equation 3.2, getting the average and reformulating the correlation of fluctuations, the final result is the following equation:

$$\rho \left[\frac{\partial U_i}{\partial t} + U_j \frac{\partial U_i}{\partial x_j} \right] = - \frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j} (T_{ij} - \rho \langle u_i x_j \rangle) \quad (3.7)$$

The right-hand term of the Equation 3.7 is the contribution of the fluctuation of non-linear acceleration terms, which has not the unit of measure of a stress as in the left-hand term. From the motion point of view, it is acting as a stress, so with this consideration, it takes the name of Reynolds stress.

The main objective of the turbulence model is to solve the Reynolds stress which can be done thanks to three main selection to take:

Linear eddy viscosity models	Also known as Boussinesq hypothesis, it relates the Reynolds stress tensor proportional to the main strain rate tensor
Non-linear eddy viscosity models	An eddy viscosity coefficient is used to link the mean turbulence field to the mean velocity field
Reynolds stress model	The eddy viscosity is not taking into account and it computes directly the Reynolds stresses

Table 3.1: Models to solve Reynolds stresses [13]

As explained in Table 3.1, there are different ways to perform turbulence analysis of a flow; according to the software used to perform the CFD analysis, in this project, it has

used the *linear eddy viscosity model*, which is the most used for turbulence modelling and it is the most linear to apply among them.

3.1.3 Linear eddy viscosity model

The linear relationship discussed in the previous section is:

$$\rho \langle u_i u_j \rangle = 2\mu_t S_{ij} - \frac{2}{3}\rho k \delta_{ij} \quad (3.8)$$

The last term is the so called *Boussinesq hypothesis*, it required by tensorial algebra purpose.

Among the models included in this category, the attention is focused on those in which two extra transport equations show the turbulent property of the flow; these are named *two-equation turbulence model*. By using this model, it allows to account for history effect like diffusion of turbulence energy and convection. These models are the most common used and many softwares are available for this analysis. Very often, the first variable determines the turbulence energy and the second one the scale of turbulence.

For the project it has been used one of the so-called $k-\omega$ models. Again, among these, it has been chosen the sub category of *SST $k-\omega$ model*, which is described in the next section.

3.2 SST $k-\omega$ model

It based on physical experiments and attempts to predict solutions for common engineering issues. It can be used as low-Re turbulence model without including any other damping function. The *shear-stress transport* (SST) formulation became attractive for its features, it mixes the best of two big categories, which are $k-\omega$ and $k-\epsilon$ models.

The transport equations in order to calculate the 2 parameters k and ω are, respectively, the following:

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_i}(\rho k u_i) = \frac{\partial}{\partial x_j} \left(\Gamma_k \frac{\partial k}{\partial x_j} \right) + \tilde{G}_k - Y_k + S_k \quad (3.9)$$

$$\frac{\partial}{\partial t}(\rho \omega) + \frac{\partial}{\partial x_i}(\rho \omega u_i) = \frac{\partial}{\partial x_j} \left(\Gamma_\omega \frac{\partial \omega}{\partial x_j} \right) + \tilde{G}_\omega - Y_\omega + D_\omega + S_\omega \quad (3.10)$$

It uses the $k-\omega$ model in the inner part of the boundary layer, which allows to have a model usable until the wall through the viscous sub-layer and, at the same time, it allows the usage of the $k-\epsilon$ model in the free-stream region where the other model has some unwanted fluctuations. The big merit of this model is the good performance on the separating streams and pressure gradients.

On the other hand, the SST $k-\omega$ model generates a little bit too much turbulence levels but, by doing so, it is more accurate and less intense than the pure $k-\epsilon$ model. In the next subsection, it computes and analyses the meaningful parameters of this selected turbulence model.

3.3 Computation of the parameters

The division is in various regions according to the geometry considered and to the behaviour of the water flow within the control volume. The domain is divided up in 5 zones regarding the Flatblades model and in 6 regions regarding the Streamlines geometry.

Because of its completeness, in Table 3.2 the ones concerning the model of Streamlines are presented; in white squares, there are roughly outlined the regions investigated during the study (it detected that some zones are overlapping between each other); as it examined, the regions are peculiar at their characteristics in terms of area considered or water features, such as vortex generated in vortex basin area or water flowing down in rotating region and so on. We need to have an initial guess for the parameters k and ω as well as suitable values for all the area studied [18].

After the estimation, the boxes are considered to set the parameters properly and have a better performance from the fluid dynamic point of view and refine the CFD analysis, inserting these values with the appropriate OpenFOAM directory.

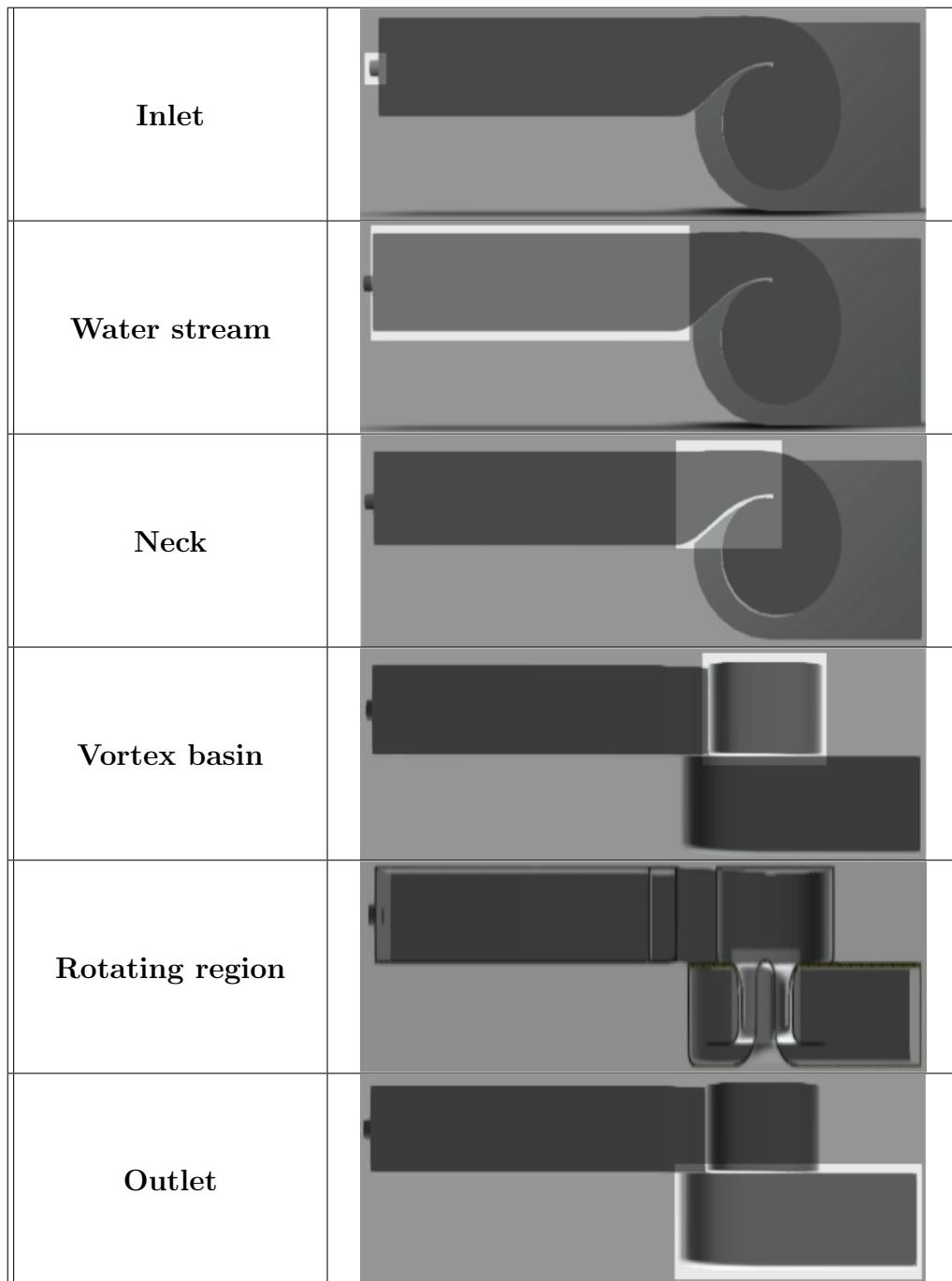


Table 3.2: Turbulence parameters' region

All the parameters necessary for the model are shown under. The Table 3.3 is resuming the entire values list to set a proper turbulence model to run the CFD analysis.

Turbulence intensity [-]	It's the root-mean-square ratio between the velocity fluctuations and the mean flow velocity; generally, the value is between 1% and 10%, in free-stream region it could be as low as 0.05%	$I \equiv \frac{u'}{u_{avg}} = 0.16 Re_{D_H}^{-1/8}$
Hydraulic diameter [m]	Relevant dimension of the duct, varying according to the configuration (A is the area section of the duct and p is the “wetted” perimeter of the duct)	$D_H = 4 \frac{A}{p}$
Turbulent length scale [m]	A physical quantity that is related to the energy in turbulent flows contained in the large eddies size, 0.07 is related to the maximum value of the mixing length in fully-developed pipe flow	$l = 0.07 D_H$
Turbulent kinetic energy [m^2/s^2]	This formulation is from the intensity value and taking the average velocity	$k = \frac{3}{2} (u_{avg} I)^2$
Turbulent dissipation rate [m^2/s^3]	This formulation is from turbulence length scale (C_μ is an empirical constant, approximately 0.09)	$\varepsilon = C_\mu^{3/4} \frac{k^{3/2}}{l}$
Specific dissipation rate [1/s]	This formulation is from turbulence length scale (C_μ is an empirical constant, approximately 0.09)	$\omega = \frac{k^{1/2}}{C_\mu^{1/4} l}$

Table 3.3: Turbulence parameters [18]

Some assumptions had to be made: most of the parameters are originally unknown so, except for the geometry entries, the velocity parameters are the ones coming from the 2nd Flatblades simulation results, which is the most close to the real condition for that geometry, but it has been extended also for Streamlines model. Regarding *Outlet* box, which in Flatblades doesn't exist, it has made a further approximation (value considering the average of *Water stream* and *Rotating region* boxes).

Hence, concerning the Streamlines configuration, the final results for all the parameters are the following:

	Inlet	Water stream	Neck	Vortex basin	Rotating region	Outlet
Turbulence intensity I	0.036	0.036	0.040	0.031	0.034	0.032
Turbulent length scale l	0.007	0.011	0.009	0.030	0.012	0.022
Turbulent kinetic energy k	0.004	0.002	5×10^{-4}	0.002	0.004	0.002
Turbulent dissipation rate ε	0.007	0.001	2×10^{-4}	6×10^{-4}	0.003	8×10^{-4}
Specific dissipation rate ω	17.29	6.79	4.30	2.99	9.43	3.88

Table 3.4: Turbulence parameters results for Streamlines

The average values for the specific dissipation rate and for turbulent kinetic energy, which are needed for the OpenFOAM setup, are the following:

k	$0.002493 \text{ m}^2/\text{s}^2$
ω	7.44873 1/s

Table 3.5: Average values for k and ω for Streamlines

Concerning the Flatblades configuration, the final results for all the parameters are

the following:

	Inlet	Water stream	Neck	Vortex basin	Rotating region
Turbulence intensity I	0.036	0.033	0.037	0.028	0.034
Turbulent length scale l	0.007	0.023	0.019	0.056	0.012
Turbulent kinetic energy k	0.004	0.001	4×10^{-4}	0.002	0.004
Turbulent dissipation rate ε	0.007	3×10^{-4}	7×10^{-5}	3×10^{-4}	0.003
Specific dissipation rate ω	17.29	2.85	1.93	1.47	9.52

Table 3.6: Turbulence parameters results for Flatblades

This model contains 5 area, contrarily the previous one which also has the Outlet area as shown in the Table 3.2. The average values for the specific dissipation rate and for turbulent kinetic energy, which are needed for the OpenFOAM setup, are the following:

k	0.002404 m^2/s^2
ω	6.612115 1/s

Table 3.7: Average values for k and ω for Flatblades

Unfortunately, in OpenFOAM, it is not possible to set all the boxes in the dictionary *setFields* because, for the turbulence values purpose, it is impracticable to insert boxes or any other geometrical volume with splines or curvature, only boxes or cylinders are available for this version of OpenFOAM. During Streamlines simulation and in the last Flatblades ones, these calculated parameters regarding average, vortex basin and rotating region values are considered within the directories.

OpenFOAM returns standard coefficients in the source code but, to specify the values just calculated, it needed to inserted sub-dictionary which specify the volume domain at which the parameters are set up. In this case, these values are k and ω they have been added in RAS file.

Chapter 4

Turbulent models

In this chapter, it focused the attention on the 2 configurations the analysis is made at. Both geometries are similar between each other, the pivotal part of the model is the runner and the most important values to consider are the torque applied by the water passing through it, pressures and velocities through the runner. Throughout the chapter, it explained the geometry of both models, how the simulations are set up and the final results. The models are drawn in Inventor and, in this first step, they are as *shell* ones, like in the real life. Secondly, it has been needed a water tightened volume, which means using lids to tight the volume where the water is flowing by, thanks to a feature in Inventor, it is possible to cap the inlet, outlet and the upper parts of the both models, basically the regions where the air interact with the water flow. The origin of the model is set as depicted, more concretely for the Flatblades configuration in Figure 4.1.

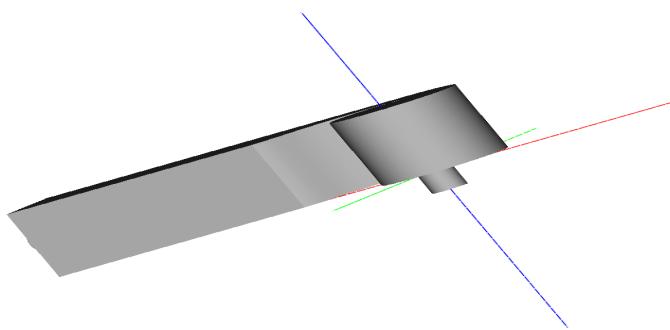


Figure 4.1: Origin and axis on the model

The next step is to split the configuration in different patches by using Meshmixer,

thanks to a swap brush, the zone we want to separate from the original model for the basin are highlighted and export them one by one in .stl format, they are the items linked in *constant*. Below, it shown particular views from the Flatblades model:

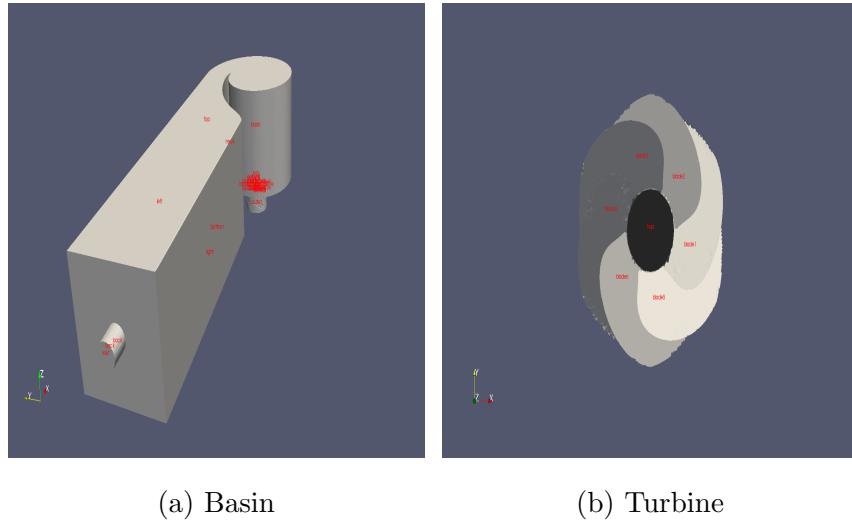


Figure 4.2: Patches of the model

In the same manner, Streamlines is divided into patches too but with the additional part of the diffuser and the area where the water is flowing out from the diffuser. The results are evaluated using the software *paraFoam* which is the main post-processing tool, in the Figure 4.3 it revealed a sample of this software windows:

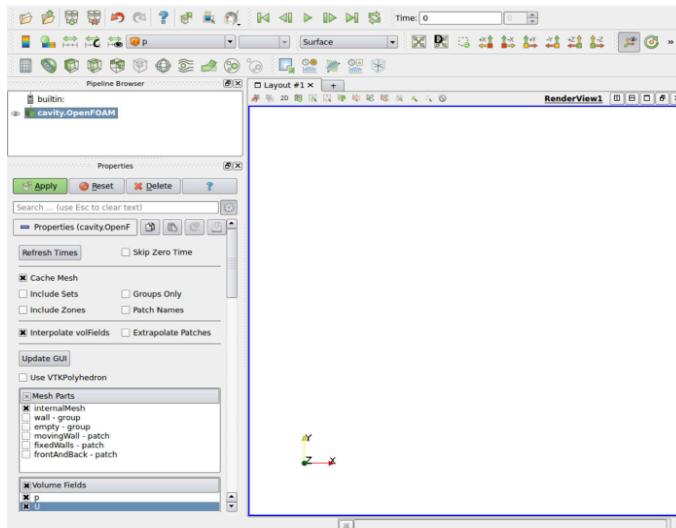


Figure 4.3: paraView home

It is a reader module that has to run in parallel with ParaView, the visualisation application by OpenFOAM. The latter uses VTK programme (Visualisation ToolKit) in order to read any data. In Pipeline Browser, the result simulation folder is loaded; in Properties Panel contains the selection of the input taken into account like time, pressure and so on and, in Display tab, colours, mesh and general representations are included.

For this project, some filters are applied in order to appreciate as much as possible the final results of a simulation, as shown in Table 4.1.

Most of the parameters are evaluated in their limit time steps: 0 sec, to check the initial conditions of water height and, in case of refined turbulence parameters, the default amounts inside the control volume, the cylinders concerning vortex basin, rotational region as well as the meshing criteria and last time step, especially for the pressure and velocities amounts. Further analyses by moving and rotate the model have been made to appreciate the velocity and pressure at the inlet and outlet of the rotating region and diffuser area, values required for the power calculations.

Filters	Specifications	Purpose
Alpha.water	Clip alpha.water field	To appreciate the water height at begin and end of the simulation.
3D Velocity magnitude	Clip alpha.water field by 0.8 absolute value filter and setting velocity as visualising datum	To evaluate the velocity throughout the model.
2D Velocity magnitude	Clip alpha.water field by 0.8 absolute value filter, setting velocity as visualising datum and introducing the x-cutting plane	To evaluate the velocity at the runner zone.
3D Pressure magnitude	Clip alpha.water field by 0.8 absolute value filter and setting pressure as visualising datum	To evaluate the pressure throughout the model.
2D Pressure magnitude	Clip alpha.water field by 0.8 absolute value filter, setting pressure as visualising datum and introducing the x-cutting plane	To evaluate the velocity at the runner zone.
Turbulent kinetic energy	Clip alpha.water field by 0.8 absolute value filter, setting turbulent kinetic energy as visualising datum and introducing the x-cutting plane	To evaluate k at the runner zone.
Specific dissipation rate	Clip alpha.water field by 0.8 absolute value filter, setting specific dissipation rate as visualising datum and introducing the x-cutting plane	To evaluate ω at the runner zone.

Table 4.1: Results analysis

4.1 Power calculations

The power is strictly related to the water stream properties, shape of the blades and geometry of the runner itself. Special focus concerns the torque value, this parameter should be as much accurate as possible because, according to the next formula, it provides the relation between the power generated and the torque from the water stream passing through the runner:

$$P_t = |T| \frac{N}{60} 2\pi \quad (4.1)$$

After the simulation, OpenFOAM issued a .dat file called *MomentZ*, where all the variables output wanted are mentioned in *controlDict* dictionary. It represents all the forces involved in the runner area, including the torque value, the syntax is the following:

Time	Forces (pressure, viscous, porous)	Moment (pressure, viscous, porous)
------	------------------------------------	------------------------------------

Table 4.2: Forces and Torques syntax results

Any single item of forces and moments are further split in x-y-z components.

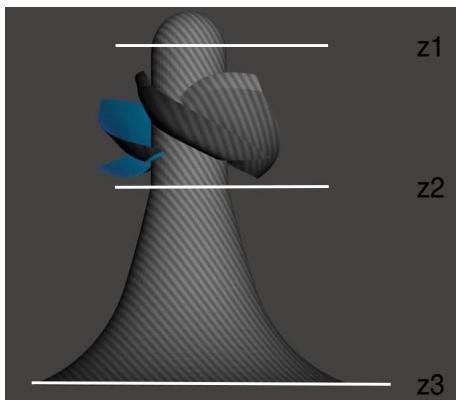


Figure 4.4: Heights in runner and diffuser zones

For this project's purpose, it focused the attention on the z-component of the pressure entry in moment vector which is the torque value needed for the above formula. Hereafter, it shows the pictures of the most meaningful parameters needed for a complete CFD simulation.

The main goal of the project is to find out how much power is generated from the water stream via runner. To do so, the most important heights of the rotating region have been split into three meaningful levels as indicated below:

1	Inlet of the turbine
2	Outlet of the turbine/Inlet of the diffuser
3	Outlet of the diffuser

Table 4.3: Particular points in the models

In Figure 4.4, there are the graphical description of the 3 heights. Hence, it applied the conservation of energy principle between the points 1 and 2 which states that the net rate of energy transfer into a control volume by heat and work transfer is equal to the time rate of change of the energy content of the control volume plus the net flow rate of energy out of the control surface by mass flow. [19]

Taking the consideration of incompressible fluid through the system and steady state, it can be addressed the following equation, regarding the runner:

$$\dot{Q}_{net\ in} + \dot{W}_{s,net\ in} = \sum_{in} \dot{m} \left(h + \frac{c^2}{2} + gz \right) - \sum_{out} \dot{m} \left(h + \frac{c^2}{2} + gz \right) \quad (4.2)$$

In this project, the first term can be deleted because there is no thermal energy involved and the mass flow rate for unique single-stream device remain constant, furthermore, dividing both members by mass flow rate, the Equation 4.2 reduces to:

$$w_{s,net\ in} = h_2 - h_1 + \frac{c_2^2 - c_1^2}{2} + g(z_2 - z_1) \quad (4.3)$$

Using the definition of enthalpy $h = u + p/\rho$ and considering the assumption of ideal flow (no irreversibility, so the internal energy remains constant), it rearranged into Formula 4.3:

$$w_{s,net\ in} + \frac{p_1}{\rho_1} + \frac{c_1^2}{2} + gz_1 = \frac{p_2}{\rho_2} + \frac{c_2^2}{2} + gz_2 \quad (4.4)$$

where, per unit mass: p/ρ is the *flow energy*, $c^2/2$ is the *kinetic energy* and gz is the *potential energy* of the fluid considered. For sake of simplicity, it introduces the mechanical energy

balance on unit-mass basis : $e_{m,i} = e_{m,o} + e_{m,l}$ where the left-hand term is the mechanical energy at the inlet and the right-hand term is the mechanical energy at the outlet plus the energy due to losses, mostly friction. Additionally, it adds the mechanical work output of the turbine as $w_{s,net\ in} = -w_t$ and so it can type the final version, expressed in the Formula 4.5:

$$\frac{p_1}{\rho_1} + \frac{c_1^2}{2} + gz_1 = \frac{p_2}{\rho_2} + \frac{c_2^2}{2} + gz_2 + w_t + e_{m,l} \quad (4.5)$$

Adding the condition that the density is not changing during the process (incompressible flow, $\rho_1 = \rho_2$), introducing the specific water weight equals to the water density times the gravity acceleration and reformulate the formula in terms of heads, which is more helpful in this case, the above equation can be also written like:

$$\frac{p_1}{\gamma} + \frac{c_1^2}{2g} + z_1 = \frac{p_2}{\gamma} + \frac{c_2^2}{2g} + z_2 + h_t + \Delta_{12} \quad (4.6)$$

The last term in the Formula 4.6, regarding the losses due to friction, is negligible because of the little span between the two points (few centimetres). From this formulation, the last step in order to calculate the power available, is shown in the Formula 4.7:

$$P_i = \dot{m} g h_t \quad (4.7)$$

So to obtain the hydraulic turbine efficiency [20], the Equation 4.8 is examined:

$$\eta_t = \frac{P_t}{P_i} \quad (4.8)$$

Regarding the diffuser zone, from point 2 to point 3, the procedure is easier because neither thermal energy nor shaft work are within the zone and, it has been figured out that it can apply the Bernoulli principle [21], as shown in the Formula 4.9:

$$\frac{v^2}{2} + gz + \frac{p}{\rho} = constant \quad (4.9)$$

Keeping this principle leads to the following constraints:

- Incompressible flow: the condition of constant density must be satisfied.
- Steady flow: the water flow itself is not in transient period, so no changes in flow conditions.
- Flow along a streamlines: no vorticity in the flow field.
- No heat transfer: not applicable for significant changes in temperatures.
- Frictionless flow: the water flows involves very little amount of frictional effects, so it can be neglected.
- No shaft work: devices, machine or impeller that can alter the streamlines must not be in the zone considered.

All the above assumptions match the diffuser taken into account.

This principle is addressed in the area in order to figure out how the diffuser affects the rotating region at the outlet, so the Equation 4.9 applied at the diffuser is in the following form:

$$\frac{p_2}{\gamma} + \frac{c_2^2}{2g} + z_2 = \frac{p_3}{\gamma} + \frac{c_3^2}{2g} + z_3 \quad (4.10)$$

However, this equation has not been used because the values p_2 and c_2 are taken directly from the simulation results. The common parameters used in this model are illustrated in Table 4.4.

γ	9810 N/m ³	g	9.81 m/s ²
ρ	1000 kg/m ³	\dot{m}	9 kg/s

Table 4.4: Constant parameters for power calculations

4.2 Flatblades

This is the first configuration analysed at testing level, it is built up with stainless steel for the basin and PLA for the runner. Physically, this model is mounted in 1 Turbulent

laboratory in Belgium but, sadly, due to lack in the basin and issues with the metering system, it has not been possible to validate the CFD results got from this project. The entire design is shown in Figure 4.5:

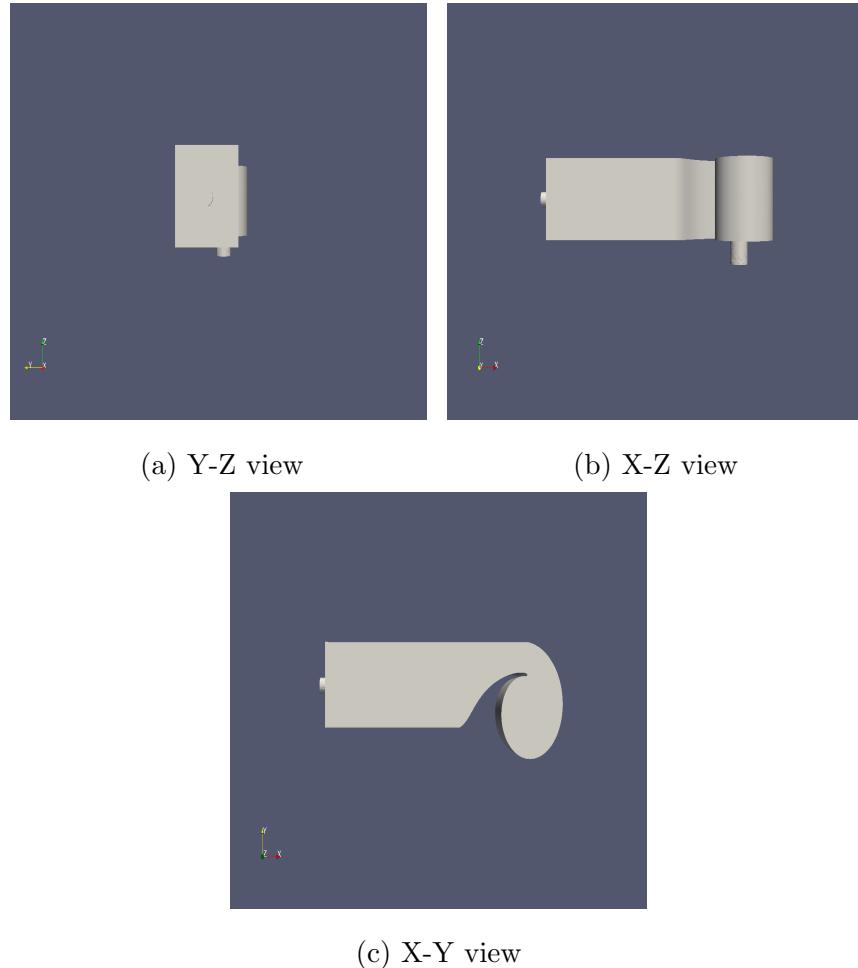


Figure 4.5: Flatblades model view

The Figures 4.5 represents the water-tightened model, this is the one ready to be simulate in OpenFOAM by the group of patches, these images are snapshots from Inventor software. Second step is the usage of the meshing tool provided, also regarding the near-wall treatment. In the next pictures, there are various particular views to highlight the treatment of the mesh according to the zone taken into account.

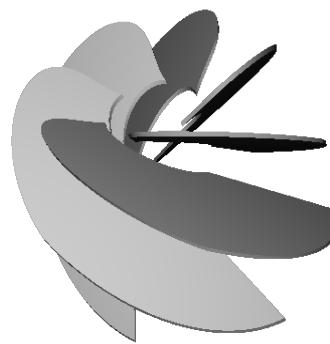


Figure 4.6: Turbines' blades for Flatblades

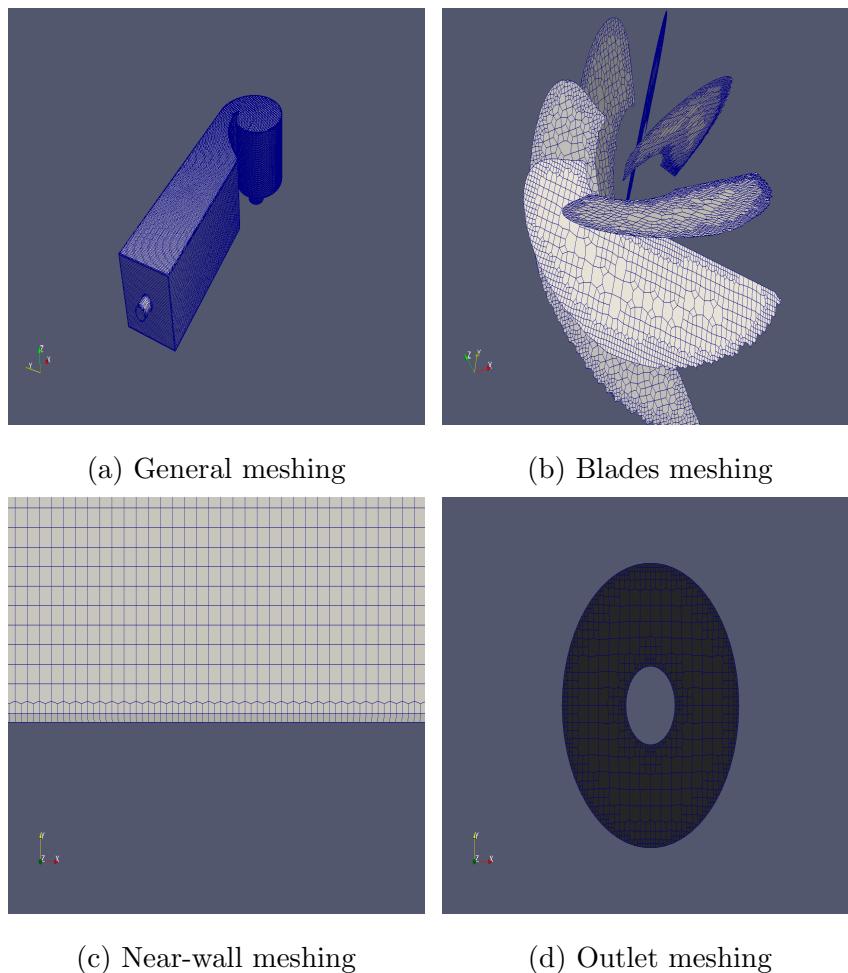


Figure 4.7: Flatblades meshing, particular views

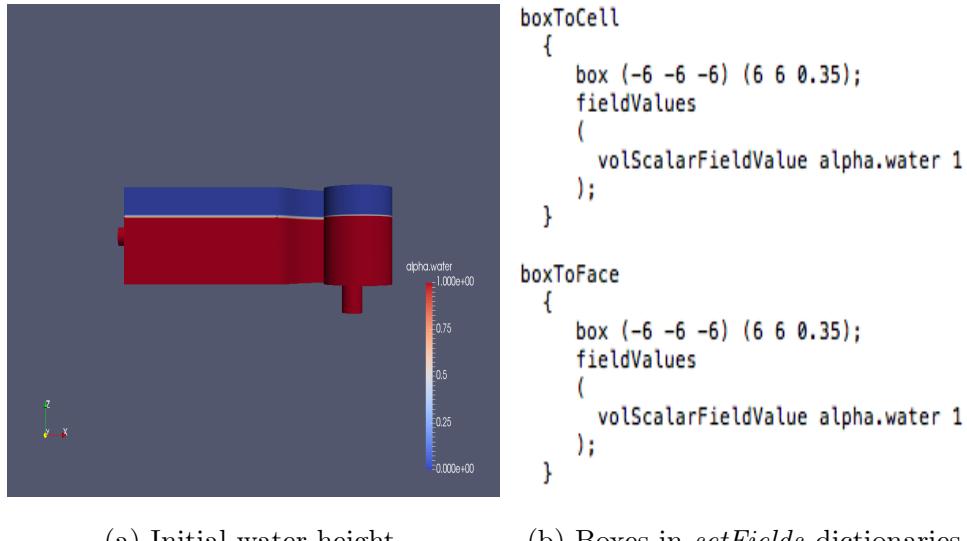
In the Figure 4.6, the blades of the turbines are illustrated, the hub is not represented because it has been treated differently, since it is not part of the fluid model, the design as shown in Figure 4.7 regards only the water-tight volume. As visible too, particular attention has been focused on the particular zones such as edge of the blades, walls and

inlet/outlet.

So now, the simulations ran on Flatblades one are presented. The simulations were done in a separate server and they took at least 3 days according to the time step given in the *controlDict* dictionary as well as on the geometry and boundary conditions. As discussed in Subsection 4.2.4, the 1st and 3rd simulations brought results that are not considered in the main project, although they are shown in Appendix C.

4.2.1 Second simulation

The 2nd simulation has been made with the constrain of nearly half water-tightened basin, as shown under:



(a) Initial water height

(b) Boxes in *setFields* dictionaries

Figure 4.8: Initial conditions for the 2nd simulation

35 cm is the initial water level inserted on the z value in the second bracket of the *boxToCell* and *boxToFace* groups, all the other entries are typed for over dimension the system and make sure that the volume considered is taken into account. The first parenthesis mentions the lowest point in terms of x, y, z amount, whereas the second one indicated the highest point and, inside *fieldValues*, the parameter it wanted to set. About the velocities:

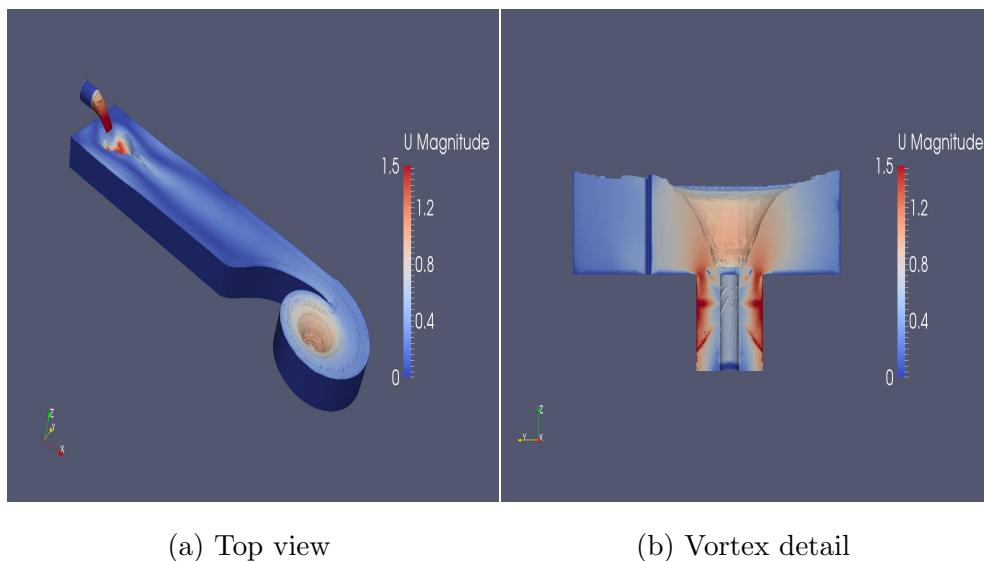


Figure 4.9: Velocities results for the 2nd simulation

Considering these results, the velocities selected for the power calculations are: 1.25 m/s at the inlet and 0.6 m/s at the outlet. About the pressures:

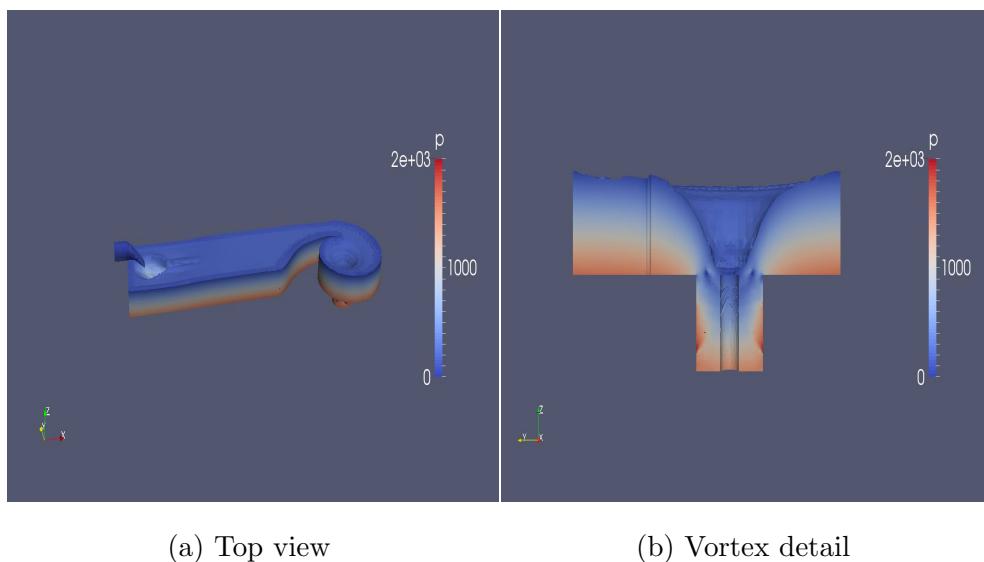
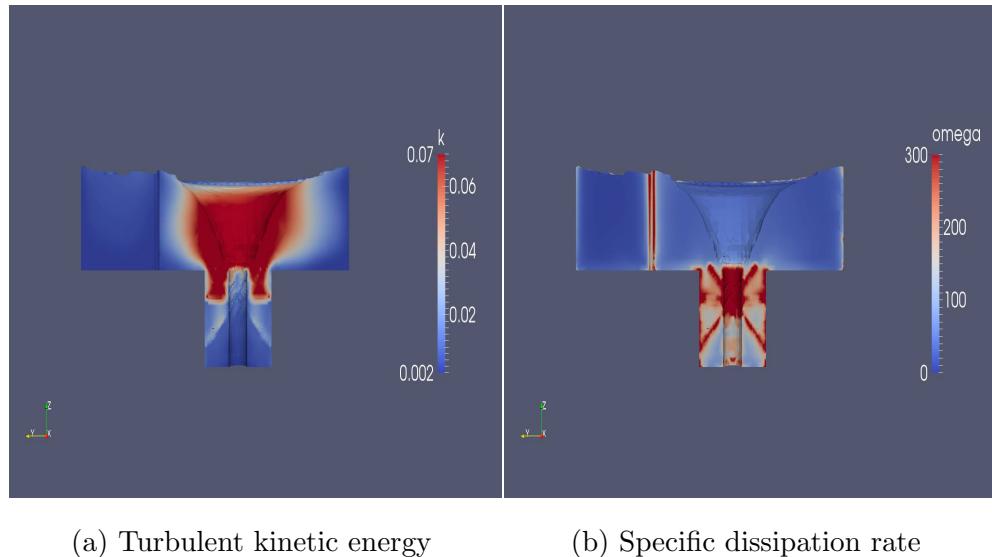


Figure 4.10: Pressures results for the 2nd simulation

Considering these results, the pressures selected for the power calculations are: 500 Pa at the inlet and 1350 Pa at the outlet. About the turbulence parameters, Figures 4.11 outlined the behaviour of the turbulence model through the runner zone:

Figure 4.11: k and ω results for the 2nd simulation

At the end, the torque generated at the last time step:

```
4.0000000e+01 ((8.3915050e-02 -3.9044443e-01 -6.3301383e+00)
(-4.9041024e-02 -1.8744974e-02 -7.9712827e-01) (0.0000000e+00 0.0000000e+00
0.0000000e+00)) ((6.8592089e-06 3.0860255e-02 -2.2628456e-01)
(-7.7683135e-04 -8.8523461e-04 5.1015488e-02) (0.0000000e+00 0.0000000e+00
0.0000000e+00))
```

Figure 4.12: Forces and Torque values for the 2nd simulation

So the Torque obtained from the simulation is -0.226 Nm approximately.

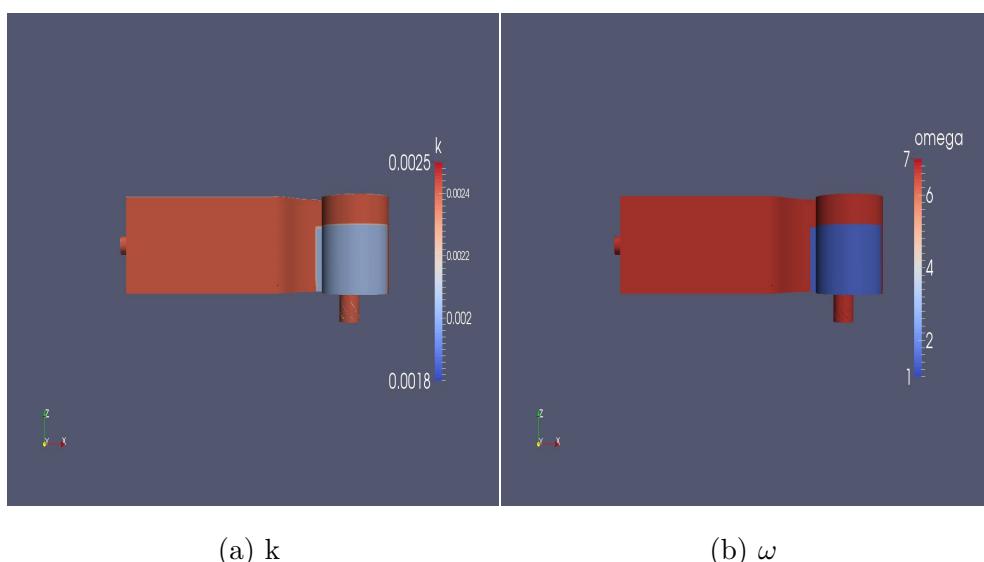
4.2.2 Fourth simulation

This simulation has been done as a adjustment of the previous case, it sets up the refined turbulence parameter as initial conditions, they are represented in Figures 4.13 as their entries in *setFields* directory and graphically in Figures 4.14:

```

defaultFieldValues
(
    volScalarFieldValue alpha.water 0
    volScalarFieldValue k 0.002404
    volScalarFieldValue omega 6.6121115
);
} cylinderToCell
{
    p1 (0 0 0);
    p2 (0 0 0.35);
    radius 0.32;
    fieldValues
    (volScalarFieldValue k 0.002034
     volScalarFieldValue omega 1.470442);
}
(a) Default values
(b) Refined parameters

```

Figure 4.13: *setFields* refined areas for the 4th simulationFigure 4.14: Turbulence parameters initial conditions for the 4th simulation

Referencing to the *cylinderToCell* syntax, it reproduced as follows: p1 is the point on the lower face of the cylinder, p2 is the point on the upper face of the cylinder, the radius and, eventually under *fieldValues*, the parameters needed to set within the volume.

About the velocities:

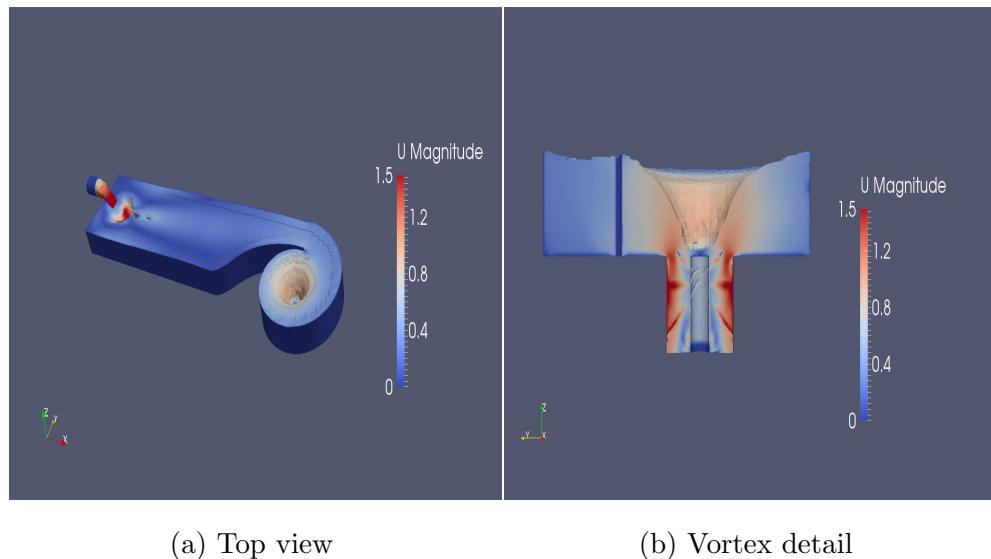


Figure 4.15: Velocities results for the 4th simulation

Considering these results, the velocities selected for the power calculations are: 1.2 m/s at the inlet and 0.4 m/s at the outlet. About the pressures:

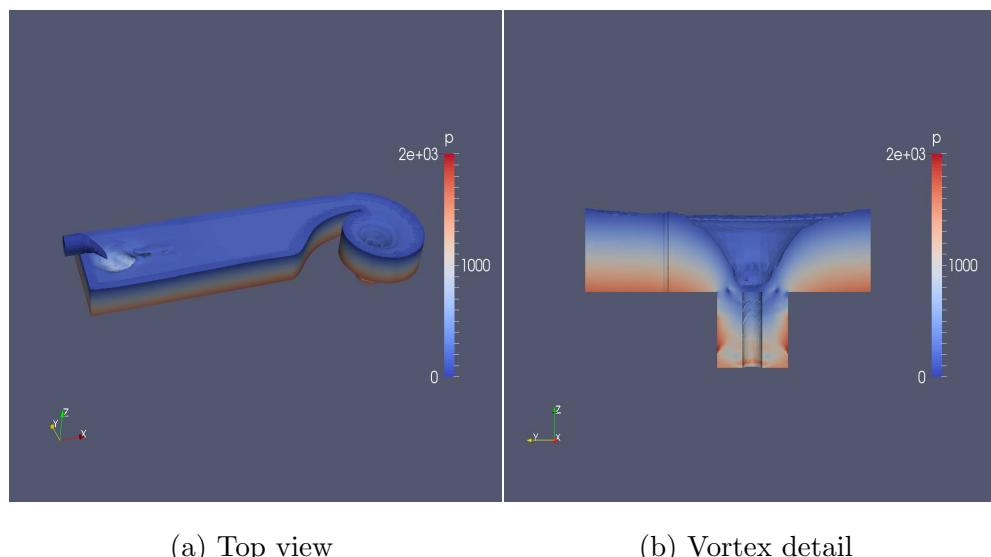
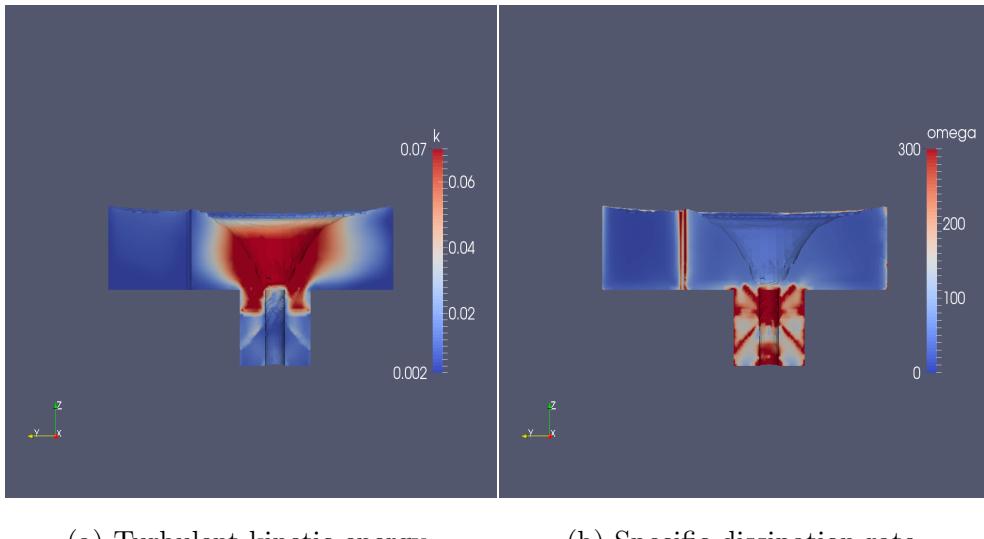


Figure 4.16: Pressures results for the 4th simulation

Considering these results, the pressures selected for the power calculations are: 550 Pa at the inlet and 1450 Pa at the outlet. About the turbulence parameters, Figures 4.17 outlined the behaviour of the turbulence model through the runner zone



(a) Turbulent kinetic energy

(b) Specific dissipation rate

Figure 4.17: k and ω results for the 4th simulation

At the end, the torque generated at the last time step:

```
4.0000000e+01 ((-3.5279046e-02 -5.4912380e-01 -6.1646321e+00)
(-5.5692272e-02 -1.6309210e-02 -7.5459321e-01) (0.0000000e+00 0.0000000e
+00 0.0000000e+00)) ((-2.5990898e-02 3.7880955e-02 -2.1566363e-01)
(-2.2134331e-03 3.0159611e-03 4.8684740e-02) (0.0000000e+00 0.0000000e+00
0.0000000e+00))
```

Figure 4.18: Forces and Torque values for the 4th simulation

So the Torque obtained from the simulation is -0.216 Nm approximately.

4.2.3 Fifth simulation

In this simulation, only one parameter varied from the 4th simulation: the average values of k and ω . In Subsection 4.2.4 has examined why.

```
defaultFieldValues
(
    volScalarFieldValue alpha.water 0
    volScalarFieldValue k 0.0038487
    volScalarFieldValue omega 9.5180608
);
```

Figure 4.19: *setFields* average values refined for the 5th simulation

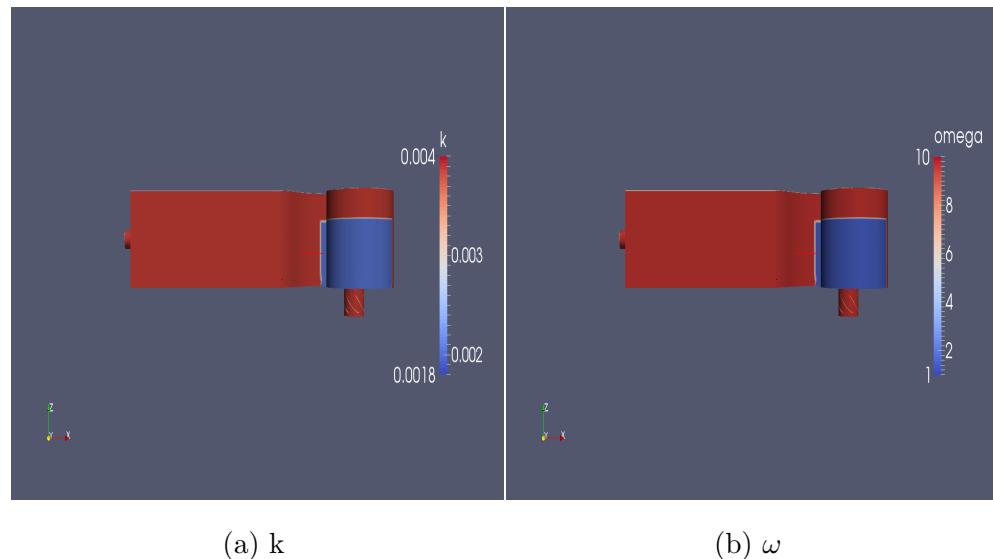


Figure 4.20: Turbulence parameters initial conditions for the 5th simulation

About the velocities:

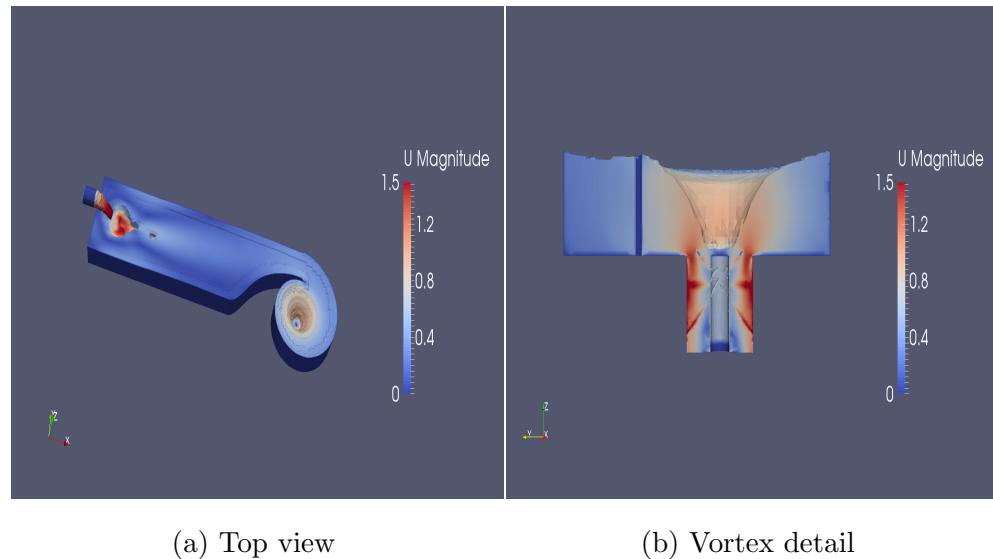


Figure 4.21: Velocities results for the 5th simulation

Considering these results, the velocities selected for the power calculations are: 1,3 m/s at the inlet and 0,55 m/s at the outlet. About the pressures:

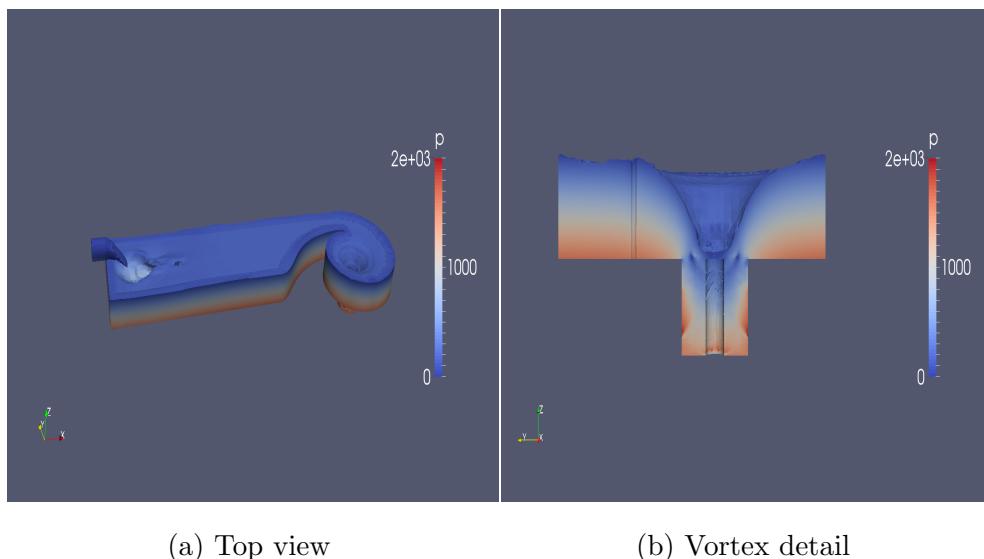


Figure 4.22: Pressures results for the 5th simulation

Considering these results, the pressures selected for the power calculations are: 530 Pa at the inlet and 1470 Pa at the outlet. About the turbulence parameters, Figures 4.23 outlined the behaviour of the turbulence model through the runner zone

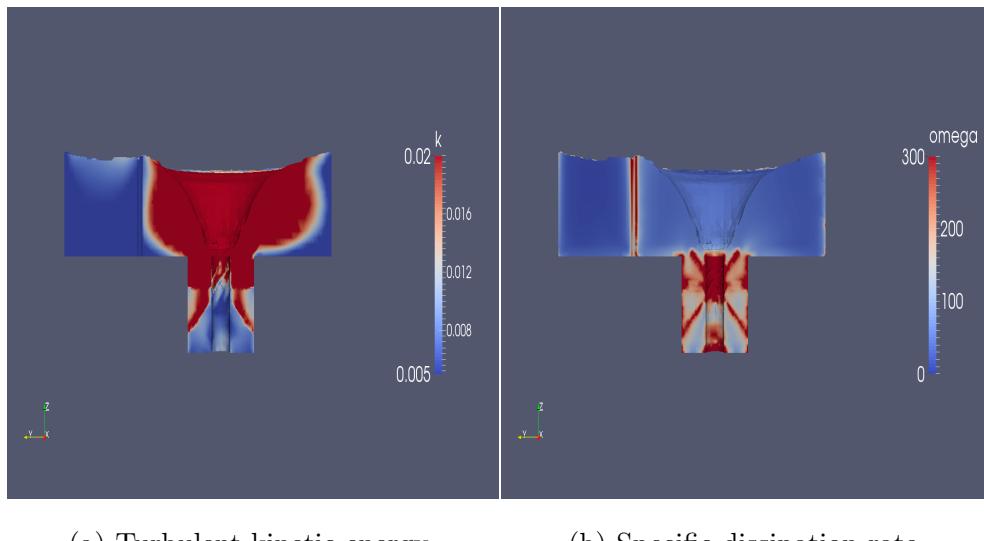


Figure 4.23: k and ω results for the 5th simulation

At the end, the torque generated at the last time step:

```

4.0000000e+01 ((1.2185518e-01 -1.7619290e-01 -6.2968696e+00)
(-6.0837802e-02 -9.9692160e-03 -7.7355330e-01) (0.0000000e+00 0.0000000e+00
0.0000000e+00)) ((-1.4386269e-02 6.7268740e-03 -2.2266400e-01)
(-2.0873116e-03 3.6518046e-03 4.9757569e-02) (0.0000000e+00 0.0000000e+00
0.0000000e+00))

```

Figure 4.24: Forces and Torque values for the 5th simulation

So the Torque obtained from the simulation is -0.223 Nm approximately.

4.2.4 Assessment of the results

Initially, the duration of the 1st simulation was very long: it took around 6 days to completely finish, the main reason is the writing procedure and the time step simulation, the results were saved every 0.1 seconds which leads to a way longer simulation and unnecessary steps to write, also many Gigabytes to save the results so, from the 2nd simulation onwards, this value has been increased to 1 sec in *controlDict* file.

The first three simulations have been made in order to appreciate the free water surface and how high the water stream level is after a certain time, 40 seconds simulation is reasonably good for visualising steady-state condition at the last and a compromise to get pretty fast results. The 4th simulation has been performed after analysing the first three simulations results as their improvements and with new enhancements to insert. Regarding the *alpha.water* initial condition: the assumption of 35 cm initial water height in the basin at the 2nd simulation is the most accurate to the real case because, as it can be seen from the animation of the first 3 simulations too, it is approaching the water flow level when reaching steady-state conditions more than the other 2 simulations which have been made under empty basin and full basin initial conditions, 35 cm is a little bit higher than half-full basin since the model is 50 cm high. So, the 4th and 5th simulations came up with this consideration and use the same initial condition as in the 2nd one. That is why the 1st and 3rd simulations are not presented in the main core of the project. Moreover, it has been added the refined turbulence parameters that are calculated in details in the Chapter 3, so only during 4th and 5th simulations, refined turbulence parameters are applied.

Concerning turbulence parameters, within the 4th simulation, it has used the average refined parameters as default values and a cylinder above the runner area with its own

values, calculated in Chapter3 but the results did not accomplish the expectations so, as a further edit, the rotating region turbulence values, as exhibited in Table 3.6, are implemented to the entire control volume for the 5th simulation in order to keep the exact turbulence amounts in the rotating region, which is the most important zone of the system both for power generation and specification accuracy and, at the same time, maintain the same cylinder set in the 4th simulation on the vortex basin.

By the separation in .stl files by Meshmixer, the model is split up in different patches, Appendix G shows how many patches are present within the configurations, it has done thorough because, whether changes in whatever parameters are made or not, it can be done patch by patch; avoiding the issue of separating the model over again for new purposes.

The turbine in the rotating region starts to run at $t = 5$ sec until $t = 40$ sec, which correspond to the end of the simulation. This value is set up in order to avoid transient problem at the begin of the simulation. From experiments, it decided to set the turbine at constant speed of 240 rpm throughout the simulation (inserted in rad/s, as shown in the Appendix E.3).

The velocity results at the inlet and outlet of the turbine are pretty similar between each others, around 0.5 at the inlet and 1.3 m/s at the outlet, as it can be viewed from the Figures 4.9, 4.15 and 4.21 as well as the previous results from Autocad CFD Simulation, as shown in the Appendix F. Looking at the Figures 4.10, 4.33 and 4.22, it is fair enough to have the same considerations for pressure results, very close values between simulations: higher on the flanks of the rotating region and lower away from the vortex (including the free water stream). So as expected, no big differences between simulations are in these results once reaching steady-state condition. It had been encountered that the pressure values at the inlet of the turbine should be higher than the ones got from the results, this is because of the quite low mass flow rate at the inlet of the basin and, additionally, the outlet pressure is nearly 3 times bigger than the inlet. The utilisation of the diffuser deals with that and it has been implemented in Streamlines model.

The most important parameter is certainly the torque: among the simulations, the

results are very analogous, around 0.22 Nm. It has been performed real-life experiments on the model and the closest value to simulation conditions are shown in the next Table:

Experimental data	$T = 0.497 \text{ Nm}$ at $\omega = 235.7 \text{ rpm}$ and $\dot{m} = 8.08 \text{ kg/s}$
Simulation results	$T = 0.226 \text{ Nm}$ at $\omega = 240 \text{ rpm}$ and $\dot{m} = 9 \text{ kg/s}$ (2^{nd} sim)
	$T = 0.216 \text{ Nm}$ at $\omega = 240 \text{ rpm}$ and $\dot{m} = 9 \text{ kg/s}$ (4^{th} sim)
	$T = 0.223 \text{ Nm}$ at $\omega = 240 \text{ rpm}$ and $\dot{m} = 9 \text{ kg/s}$ (5^{th} sim)

Table 4.5: Comparison between experimental data and simulations results

So, as it shown in Table 4.5, it figures out that the torque value should be higher and approach the experimental results. Especially for 4^{th} and 5^{th} simulations, higher torque values were expected, so the reasons why the simulation values are different from the experimental data has to be found elsewhere. In Chapter 5, some recommendations are presented.

Regarding the power calculations and taking the values of velocities and pressures at the inlet and outlet of the turbine, these are the following results:

2^{nd} simulation	$h_t = 0.1086 \text{ m}$ $P_t = 5.680 \text{ W}$	$P_{av} = 9.592 \text{ W}$ $\eta_t = 0.593$
4^{th} simulation	$h_t = 0.1075 \text{ m}$ $P_t = 5.429 \text{ W}$	$P_{av} = 9.491 \text{ W}$ $\eta_t = 0.572$
5^{th} simulation	$h_t = 0.1089 \text{ m}$ $P_t = 5.605 \text{ W}$	$P_{av} = 9.615 \text{ W}$ $\eta_t = 0.583$

Table 4.6: Power results for Flatblades

Even regarding these results, the difference between simulations is very little. The efficiency of the turbine is approximately 0.58 averagely and, actually, also the efficiency values could be higher since most of the turbine applied in this field of application have efficiency with order of magnitude approaching 70%. [22]

4.3 Streamlines

This second configuration is named Streamlines because the so-called company designed and compared models with the same concept as Flatblades, but aiming some requirements such as easy installation, shipping by pallet, capable to scale and adaptable and so on. [23]

Streamlines provided different models, but throughout this project, only one is addressed. The layout of this configuration is quite similar with the previous configuration with an additional part appended at the outlet, thanks to the implementation of the diffuser and lower walls in it. The whole model is shown in Figure 4.26. It is build out of a straightforward scroll housing, a small compact turbine based on properly designed hydrofoils. The outlet consists of a limited height below the basin.

Contrarily the previous configuration, the characteristic turbine inserted in the model has 5 blades instead of 6, as shown in Figure 4.25

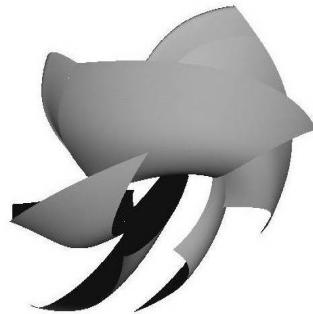


Figure 4.25: Streamlines turbine

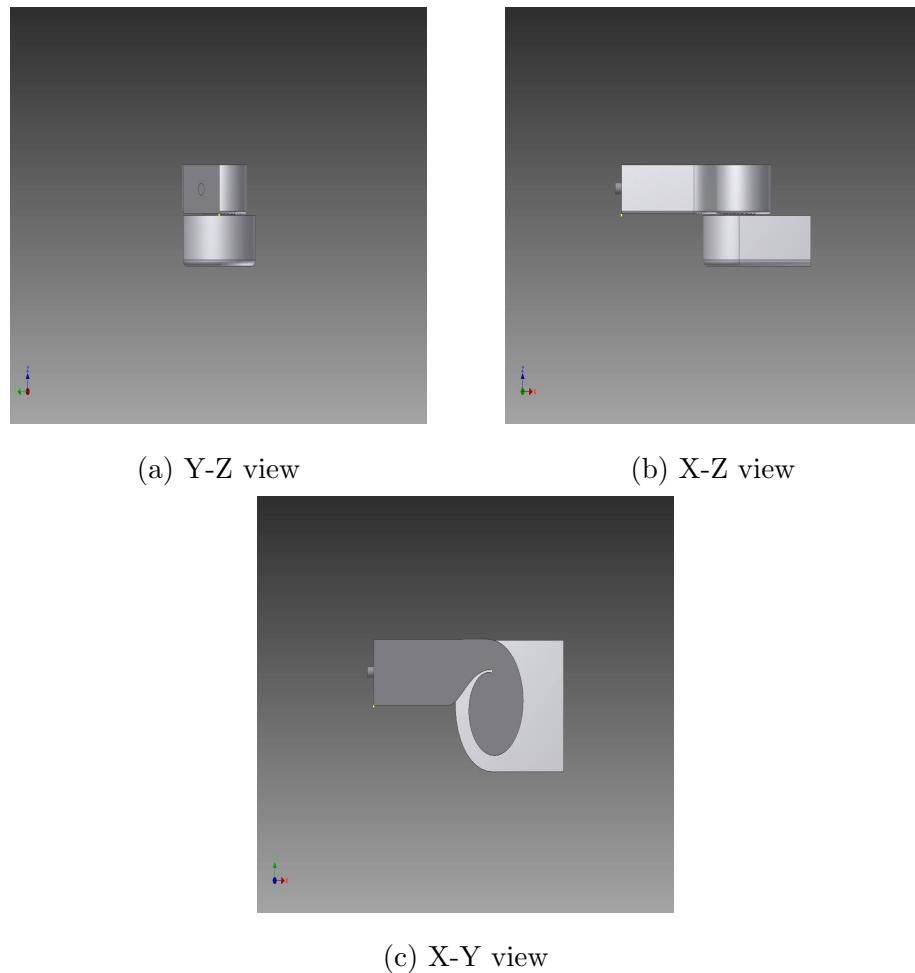


Figure 4.26: Streamlines model views

Origin and axes in this configuration are the same as shown in the Figure 4.1.

As visible from Figures 4.27, particular attention has been focused on the particular zones such as edge of the blades, walls and inlet/outlet.

In order to check the closed surface, using the command *checkSurface* in OpenFOAM, it made sure that the volume was completely closed and there are no illegal triangles within it.

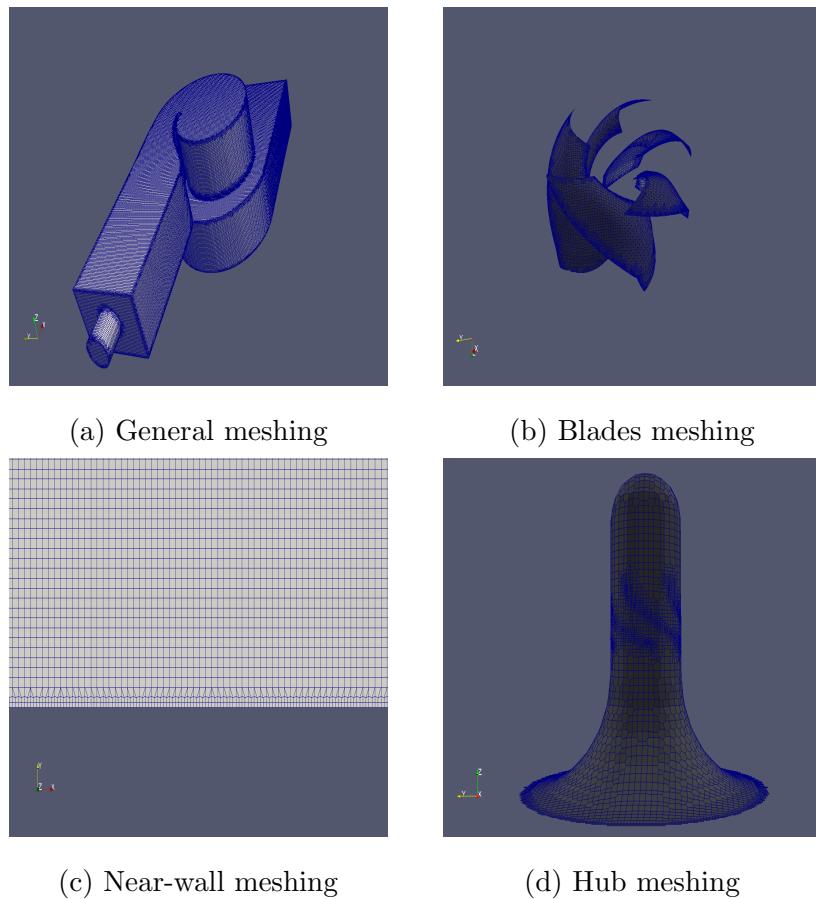


Figure 4.27: Streamlines meshing, particular views

4.3.1 Diffuser

A pivotal zone for the Streamlines model is surely the diffuser. The main role of the diffuser is to keep the air out of the runner, placing it deep and making the water flow in such a way to be as efficient as possible. In the Figure 4.28 (snapshot took from Meshmixer environment), it shown the standard layout of the diffuser assembly which, in the Streamlines simulation, has built up; further geometrical features are shown in Appendix B.

The main focus is to maximise both the difference of velocity and pressure through the diffuser. By doing so, the turbine can take advantages of the water stream as much as possible. The velocity at the outlet of the diffuser is strongly dependent on the diffuser layout.



Figure 4.28: Standard diffuser for Streamlines

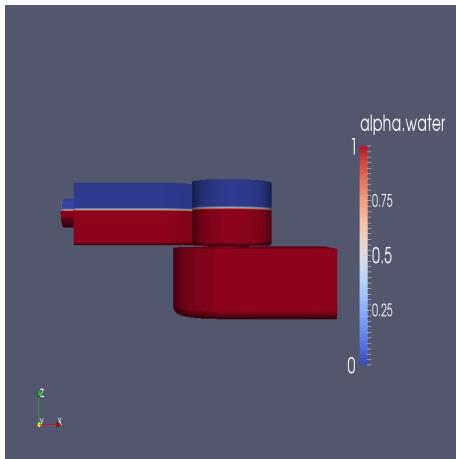
Thanks to this configuration, the problem with the stream separation on the walls is avoided because of the high curvature between inlet and outlet of the diffuser. For reasons accounting the velocity, in order to decrease it of nearly 3 times, the outlet area should be 3 times bigger than the inlet area.

Separately from this project, different diffuser geometries were analyzed (the one in Figure 4.28 is drawn as in the standard geometry for Streamlines company) and the task was to find out the best shape and geometry for it because it affects both velocity and pressure on the outflow zone of the model. [24]

4.3.2 First simulation

The mesh regarding the model is more refined than Flatblades one, for *blockMeshDict*, the *simpleGrading* is almost double of the previous model, dealing with the geometry values. It leads to a simulation way more slow, but more accurate. Because of that, and for time hustle, the model has been simulate for only 2 seconds.

So, regarding the initial water height, the Figures 4.29 revealed this condition and its relative *setFields* values and in Figure 4.30 the usual default alpha.water value is visible.



(a) Initial water height

```

boxToCell
{
    box (-6 -6 -6) (6 6 0.15);
    fieldValues
    (
        volScalarFieldValue alpha.water 1
    );
}

boxToFace
{
    box (-6 -6 -6) (6 6 0.15);
    fieldValues
    (
        volScalarFieldValue alpha.water 1
    );
}

```

(b) Boxes in *setFields* dictionaries

Figure 4.29: Initial condition for the simulation

```

cylinderToCell
{
    p1 (0 0 0);
    p2 (0 0 -0.15);
    radius 0.09;
    fieldValues
    (
        volScalarFieldValue k 0.0038408
        volScalarFieldValue omega 9.4306438);
}

defaultFieldValues
(
    volScalarFieldValue alpha.water 0
    volScalarFieldValue k 0.002492602
    volScalarFieldValue omega 7.44782969
);
cellToFace
{
    set MRFcells;
    option all;
    fieldValues
    (
        volScalarFieldValue k 0.0038408
        volScalarFieldValue omega 9.4306438);
}

```

(a) Default values

(b) Refined parameters

Figure 4.30: *setFields* refined areas for the simulation

For what the turbulence parameters are concerning, here are presented the initial boxes as mentioned in *setFields*, according to the geometry. Different from Flatblades simulations, here the *cylinderToCell* and *cellToFace* functions properly establish the refined area at the runner zone, the second function is addressed in order to include the both faces of the blades within the analysis.

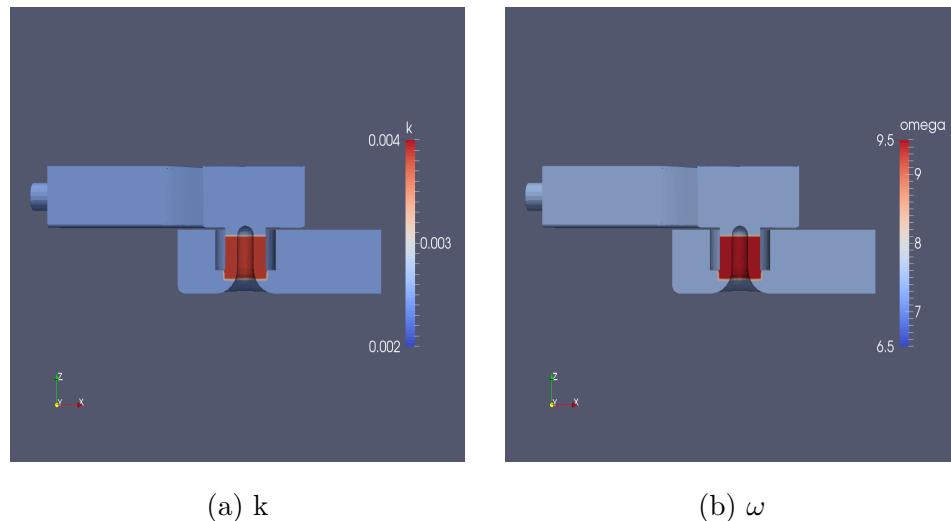


Figure 4.31: Turbulence parameters initial conditions for the simulation

About the velocities:

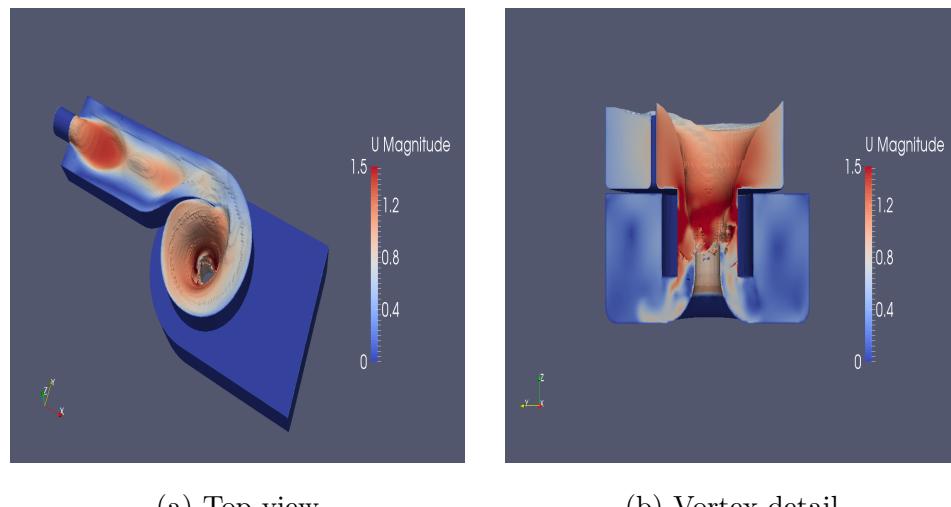


Figure 4.32: Velocities results for the simulation

Considering these results, the important velocities out of this simulation are: 1.5 m/s at the inlet and 1 m/s at the outlet and 0.7 m/s at the outlet of the diffuser.

About the pressures:

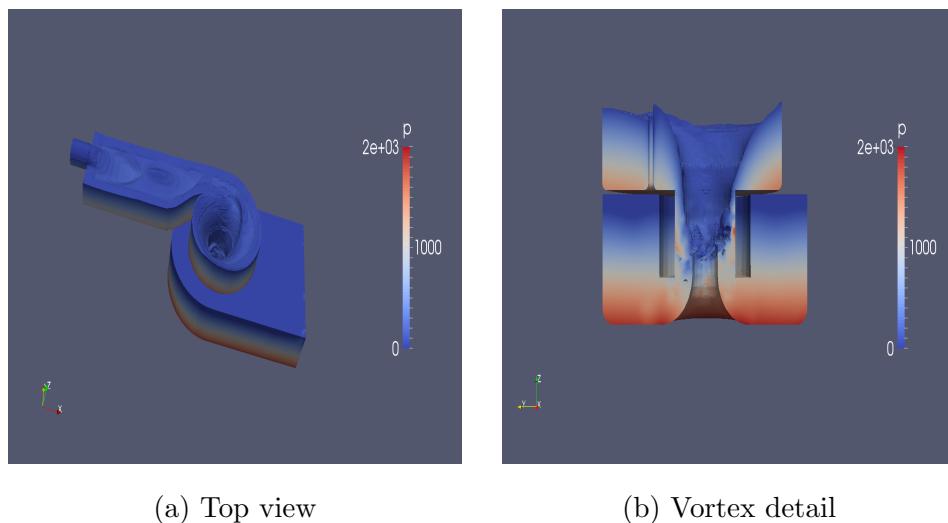


Figure 4.33: Pressures results for the simulation

Considering these results, the important pressures out of this simulation are: 500 Pa at the inlet and 1000 Pa at the outlet and 1200 Pa at the outlet of the diffuser.

About the turbulence parameters:

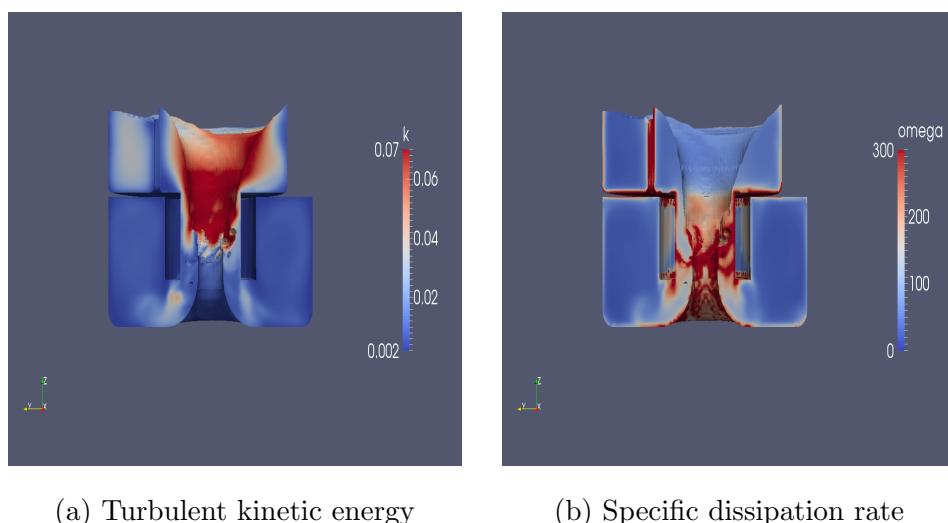


Figure 4.34: k and ω results for the simulation

At the end, the torque generated at the last time step:

```

2.0000000e+00 ((2.4094464e+00 2.7566659e+00 -5.0959734e+00)
(2.9643818e-02 -2.3008320e-02 -7.0400394e-02) (0.0000000e+00 0.0000000e+00
0.0000000e+00)) ((-1.7859141e-01 2.6277603e-02 -2.0408206e-01)
(1.5983800e-03 -8.7044382e-04 1.0700424e-02) (0.0000000e+00 0.0000000e+00
0.0000000e+00))

```

Figure 4.35: Forces and Torque values for the simulation

So the Torque obtained from the simulation is -0.204 Nm approximately. As long as the results of this simulation are concerning, longer simulation which approach the steady-state conditions are well-needed: decreasing the meshing size can help it out on having faster simulation. To simulate 2 seconds it took around 2 days!

This is why, unfortunately, the steady-state conditions are not accomplished and power results are not computed. Expecting that, specifically, the pressures of the turbine outlet would be lower and at the diffuser outlet higher than the ones got from this simulation. Either way, it can be figured out that the influence of the diffuser within the configuration is quite valuable: the decrease in pressure between the Flatblades and Streamlines simulation is around 500 Pa, which accounts for 32% of pressure drops and for the 30% of power increment (comparing these results with 5th Flatblades simulations).

Torques value is approximately the same as in Flatblades.

Chapter 5

Conclusions

Although some results are quite satisfactory, many diverse improvements can be done. First of all, a better computer in order to get the entire CFD procedure faster and more efficient is very necessary; different operative systems have to be used in addition with 2 server for simulating and visualising results, running by virtual machines. A more powerful computer is required.

CAD drawings and adjustments as well as patches separation have been made by Inventor and Meshmixer, they are powerful tools to prepare the water volume needed by OpenFOAM in *constant/triSurface* directory. Additionally, to make sure that the volume has no imperfections and flaws, MeshLab, add-in of OpenFOAM has used. So these instruments are effective to have a straightforward pre-processing CFD operations. As computed in Chapter 3, set all the boxes/cylinders for the turbulence values in *setFields* dictionary would get more accurate results but, at the same time, it slows down the simulations. Furthermore, suitable Streamlines turbulence parameters have to be inserted and run over again the simulation: the current ones are assumptions from the Flatblades result.

OpenFOAM is definitely the best software to run CFD simulation by now, even if its learning is tough, the elaborate settings allow to change whatever value in whatever patch or even cells. The only drawback encountered is regarding the interaction between fluid and material: OpenFOAM works and runs only under water-tight volume conditions, avoiding the interaction with materials which enclose the control volume. SnappyHexMesh

is a meshing tool for complex geometry and, looking at the results, it performs accurate and thoroughly. Hence, no further improvements concerning the mesh of the model and all its features are needed. To previously check the meshing before starting the simulation, many time the *meshQuality* dictionary has been ran. A further refinement for initial setup is to decrease the time step simulation in *controlDict* which, again, will make the simulation more correct but less quick.

More studies about the initial water level condition are needed, Flatblades simulations are set on 35 cm from the bottom of the model but, most likely, this height is just a little bit shorter: it has figured out from the last step simulated. I would suggest to keep it as low as 25 cm for Flatblades configuration or, in correspondence with the inlet height; whereas 15 cm for Streamlines model is logical.

Moreover, for both models, longer simulations are required: it could not be long enough for reaching the steady-state conditions completely, regarding Streamlines configuration, only a little step has been made for the simulation, normally 40 sec simulations are required to appreciate the steady-state conditions and get accurate parameters out of the simulations. Additionally, different operation points for the turbine speed are required in order to check whether the assumption of 240 rpm is consistent for a certain amount of mass flow rate (9 m/s for any simulation) with the real case or not.

Concerning the turbulence modelling: the SST $k-\omega$ turbulence model is a two-equation eddy viscosity model and the most suitable for these models in terms of compromise between wall treatment and free-stream behaviour, $k-\omega$ turbulence model is very sensitive to initial guesses inserted in the files *0/k* and *0/omega*, so the more accurate guesses are, the better result it gets. Therefore, from Figures 4.11, 4.17, 4.23 and 4.34, better guesses can be initialised.

Very important is to have more exact and meticulous real-life validation for both models, it has been very difficult compare the CFD results with the experimental data I had, because most of the operating points were with higher mass flow rates and/or higher rotational speeds. So it has not been truly possible to appreciate how much the results

are close to the experimental data.

Possible improvements are also be taken from the discretisation and time-integration schemes, because the initial estimations with simpleFoam and interFoam might not be enough accurate for these models, so more rigourous discretised solution might be helpful. Last, but not least, it is to set, as a further enhancement, new simulations with the new specific dissipation rate and turbulent kinetic energy amounts starting from the results of the 5th simulation.

List of Symbols

Δ_{ij} Regular head loss due to friction between i and j [m]

\dot{m} Mass flow rate [kg/s]

\dot{Q}_{net} Net transfer energy to a control volume by heat [J]

$\dot{W}_{s,net}$ Net shaft work transfer energy [J]

η_t Hydraulic turbine efficiency

γ Specific weight [N/m^3]

$\Gamma_{k,\omega}$ Effective diffusivity of k, ω

μ Fluid dynamic viscosity [m^2/s]

μ_t Eddy viscosity [m^2/s]

ω Specific dissipation rate [1/s]

ρ Fluid density [kg/m^3]

$\tilde{G}_{k,\omega}$ Generation of k, ω

\tilde{p} Static pressure [Pa]

\tilde{s}_{ij} i/j-component of instantaneous strain rate tensor [Pa]

\tilde{u}_{ij} i/j-component of the fluid velocity at a certain point x_{ij} and time t [m/s]

ε Turbulent dissipation rate [m^2/s^3]

c Fluid flow speed at the chosen point [m/s]

D_H Hydraulic diameter [m]

$D_{k,\omega}$	Cross-diffusion term of k,ω
$e_{m,l}$	Mechanical energy lost per mass [J/kg]
h	Enthalpy [J]
h_t	Extracted head removed from the fluid by the turbine [m]
I	Turbulent intensity
k	Turbulent kinetic energy [m^2/s^2]
L	Characteristic length [m]
l	Turbulence length scale
N	Rotational velocity of the turbine [rpm]
p	Static pressure at the chosen point [Pa]
P_t	Turbine power output [W]
P_i	Hydraulic turbine power [W]
Re	Reynolds number
S_{ij}	Mean strain rate
$S_{k,\omega}$	User-defined source term of k,ω
T	Torque generated by the water stream [Nm]
T_{ij}	Viscous (or deviatoric) stresses [Pa]
U	Average value of velocity [m/s]
u'	Velocity fluctuation [m/s]
$Y_{k,\omega}$	Dissipation of k,ω due to turbulence
z	Height at the chosen point [m]

Bibliography

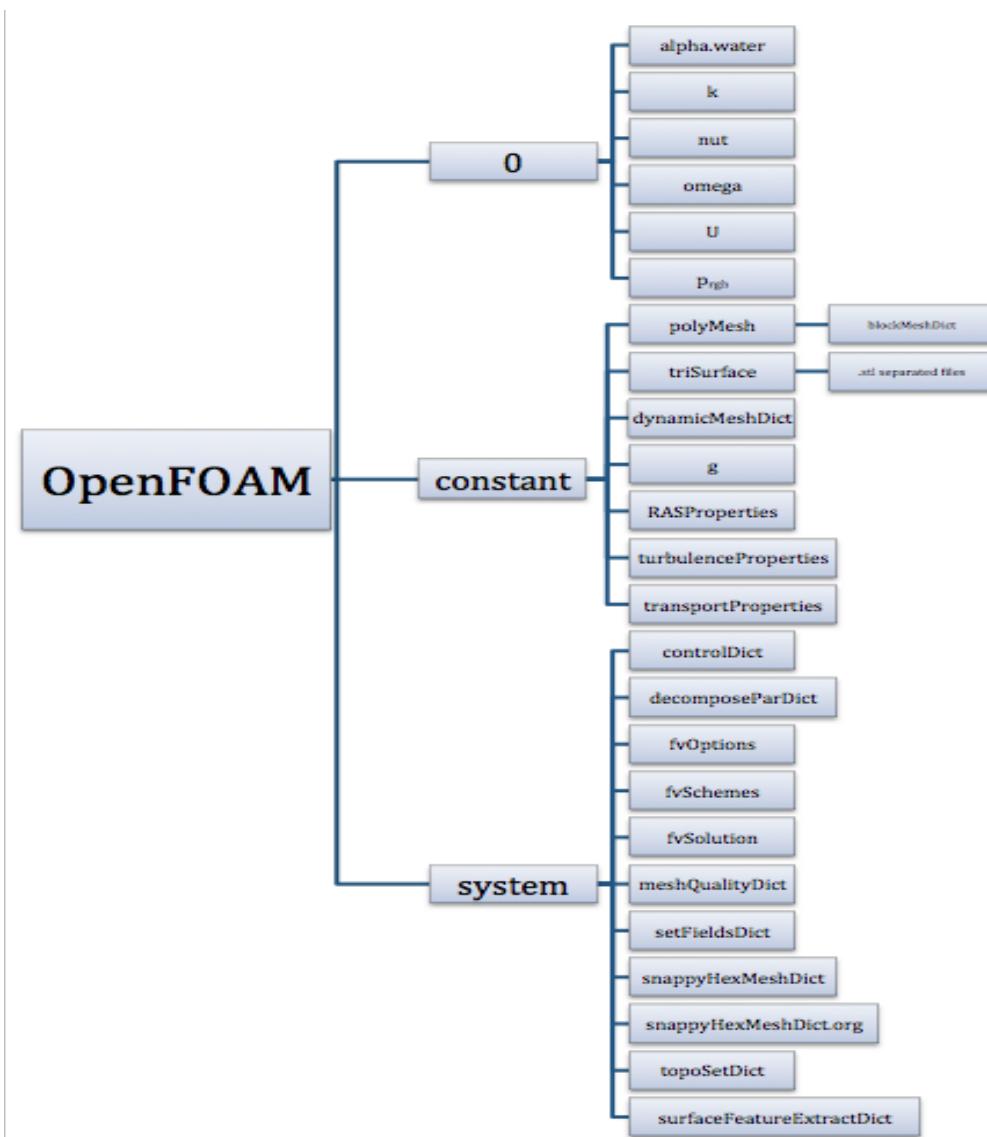
- [1] T. shift project data portal, “Breakdown of electricity generation by energy source,” Browse Energy and Climate Data, Tech. Rep., 03 2015. [Online]. Available: <http://www.tsp-data-portal.org/Breakdown-of-Electricity-Generation-by-Energy-Source#tspQvChart>
- [2] M. Conseil, “World energy resources 2013 survey,” World Energy Council, Tech. Rep., 2013. [Online]. Available: https://www.worldenergy.org/wp-content/uploads/2013/09/Complete_WER_2013_Survey.pdf
- [3] H. Perlman. Hydroelectric power and water. basic information about hydroelectricity, the usgs water science school. [Online]. Available: <http://water.usgs.gov/edu/wuhy.html>
- [4] M. A. Maehlum. Hydroelectric energy pros and cons. [Online]. Available: <http://energyinformative.org/hydroelectric-energy-pros-and-cons/>
- [5] Hydropower. [Online]. Available: <http://www.hydropower.com.cn/technologies.asp>
- [6] O. of Energy Efficiency and R. Energy. Types of hydropower plants. [Online]. Available: <http://energy.gov/eere/water/types-hydropower-plants>
- [7] A. F. Developpement. (2011, 11) Construction of a 48 mw hydropower project in jaggran. [Online]. Available: <http://www.afd.fr/lang/en/home/pays/asie/geo-asie/agence-pakistan/projets-pakistan/mandat-energie/centrale-jaggran>
- [8] “Microhydropower systems,” Energy Efficiency and Renewable Energy. [Online]. Available: <http://energy.gov/energysaver/microhydropower-systems>
- [9] Turbulent. (2016, 02) Solution-turbulenthydro. [Online]. Available: <http://turbulent.be/language/en/solution/>

- [10] Comsol. (2016) What are the navier-stokes equations? [Online]. Available: <https://www.comsol.it/multiphysics/navier-stokes-equations>
- [11] CFD-Wiki. (2016) Best practice guidelines for turbomachinery cfd. [Online]. Available: http://www.cfd-online.com/Wiki/Best_practice_guidelines_for_turbomachinery_CFD
- [12] C. J. Greenshields, "User guide, version 3.0.1," OpenFOAM Foundation Ltd., Tech. Rep., 12 2015.
- [13] OpenCFD. (2004) Openfoam- the open source computational fluid dynamics (cfd) toolbox. [Online]. Available: <http://www.openfoam.com/>
- [14] CFD-Wiki. Meshing. [Online]. Available: <http://www.cfd-online.com/Wiki/Meshing>
- [15] M. Matheson. (2016) Automatic vs manual meshing: Which one is right for you? [Online]. Available: <http://www.engineeringexchange.com/profiles/blogs/automatic-vs-manual-meshing-which-one-is-right-for-you>
- [16] S. Peters. (2016) What are newtonian and non-newtonian fluids? [Online]. Available: <https://blog.craneengineering.net/what-are-newtonian-and-non-newtonian-fluids>
- [17] Comsol. (2013) Which turbulence model should i choose for my cfd application? [Online]. Available: <https://www.comsol.com/blogs/which-turbulence-model-should-choose-cfd-application/>
- [18] Fluent. Determining turbulence parameters. [Online]. Available: <https://www.sharcnet.ca/Software/Fluent6/html/ug/node217.htm>
- [19] Y. A. Cengel and J. M. Cimbala, "Mass, bernoulli and energy equations," in *Fluid Mechanics: Fundamentals and Applications, 2nd Edition*. McGraw-Hill, 2010. [Online]. Available: <http://highered.mheducation.com/sites/dl/free/0073138355/366295/Chapter5.pdf>
- [20] K. Alexander, E. Giddens, and A. Fuller, "Axial-flow turbines for low head microhydro system," *Renewable Energy*, 2016.

- [21] What is bernoulli's principle? Khan Academy. [Online]. Available: <https://www.khanacademy.org/science/physics/fluids/fluid-dynamics/a/what-is-bernoullis-equation>
- [22] P. Action. (2005) Micro-hydro power. [Online]. Available: https://practicalaction.org/docs/technical_information_service/micro_hydro_power.pdf
- [23] Bink and V. Dijk, "Gravitational vortex generator," Streamlines, Tech. Rep., 2016.
- [24] D. S. Maldonado, "Diffuser cfd approach report," 2016.
- [25] OpenFOAMWiki. (2012) Comprehensive tour of snappyhexmesh. [Online]. Available: <https://openfoamwiki.net/images/f/f0/Final-AndrewJacksonSlidesOFW7.pdf>

Appendix A

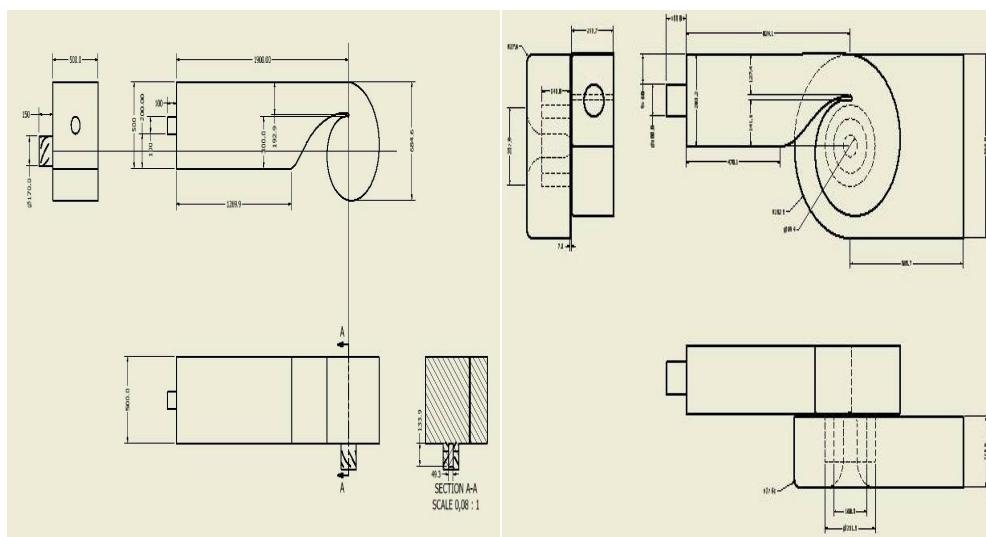
Initial setup



Appendix B

Qualitative 2D CAD drawings

These CAD models have been drawn in Autodesk Inventor in three-dimensions and, afterwards, exported in 2D .dwg files in order to visualize the geometry values. In these figures, it depicted the most valuable geometric parameters of both models (Flatblades and Streamlines). They were important not only to know the correct dimensions of the configuration, but also the origin reference because, especially to set the turbulence parameters and the rotating region, it had to be taken into account the origin and set the values accordingly, e.g. *cylinderToCell* of the turbulence model, outlines cylinders starting from the origin. As already mentioned, the origin in Flatblades and Streamlines drawing lays on the top part of the runner zone.



2D Flatblades drawing

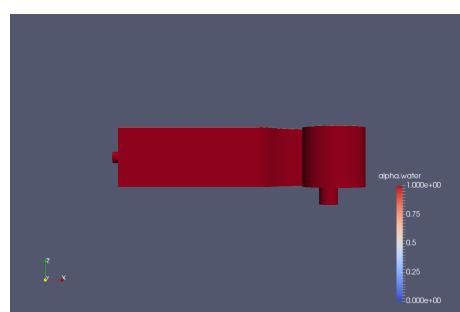
2D Streamlines drawing

Appendix C

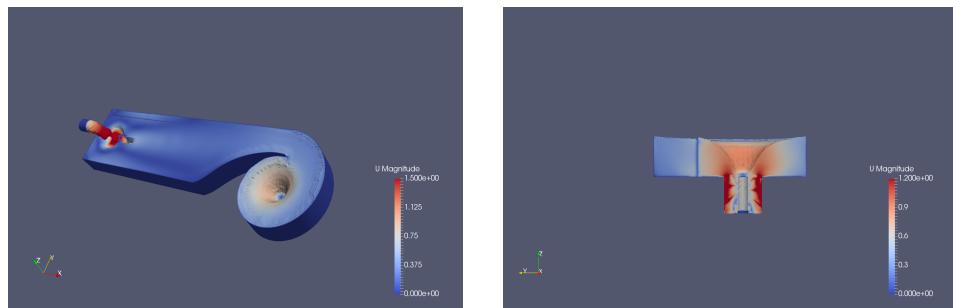
1st and 3rd Flatblades simulations

Here are presented the results for the 1st and 3rd Flatblades simulations, which are taken into consideration to compare the simulation results among different initial water height. In order, the results are displayed as water height, velocities, pressures and turbulence coefficients. Eventually, forces and torque values in the rotating region are exposed.

C.1 First simulation



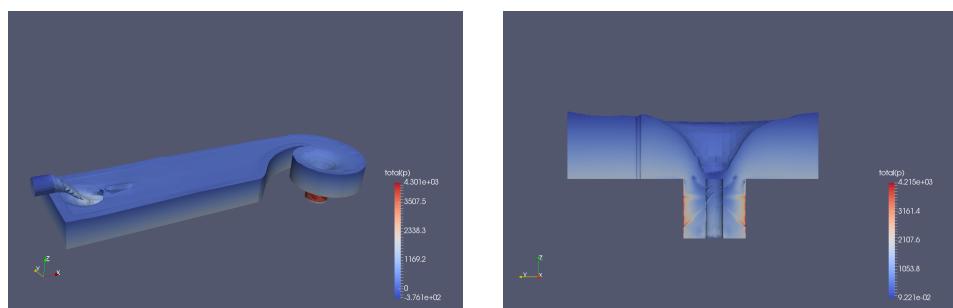
Water height initial condition



(a) Top view

(b) Vortex detail

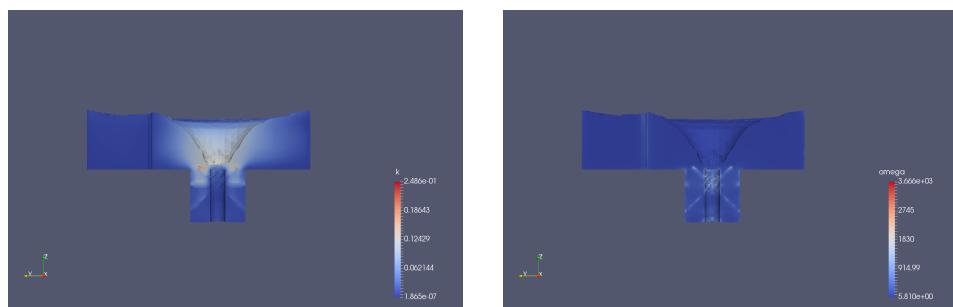
Velocity results



(a) Top view

(b) Vortex detail

Pressure results



(a) k

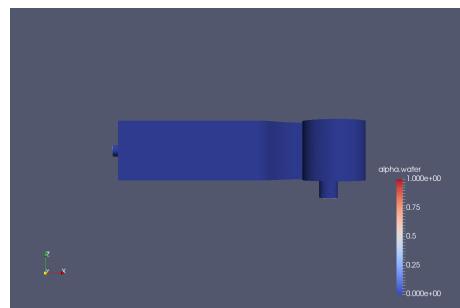
(b) ω

Turbulence parameters results

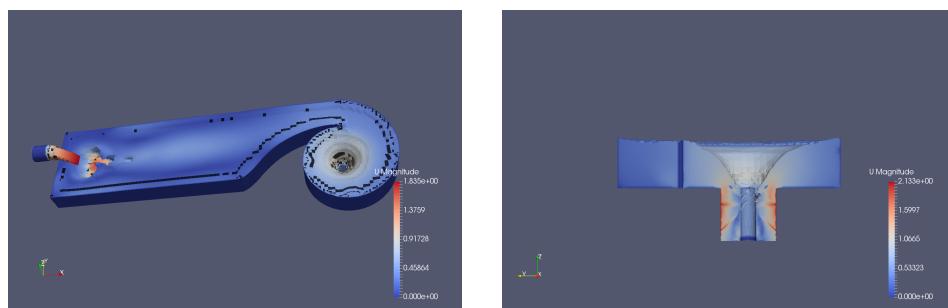
```
4.0000000e+01 ((-3.7057318e-01 -8.0708087e-02 -6.6874388e+00)
(-6.6381987e-02 -8.9217066e-03 -8.1565965e-01) (0.0000000e+00 0.0000000e+00
0.0000000e+00)) ((1.9041587e-02 1.1869997e-02 -2.3798217e-01)
(-2.1066239e-03 -1.8273948e-03 5.3423173e-02) (0.0000000e+00 0.0000000e+00
0.0000000e+00))
```

Torque and force results

C.2 Third simulation



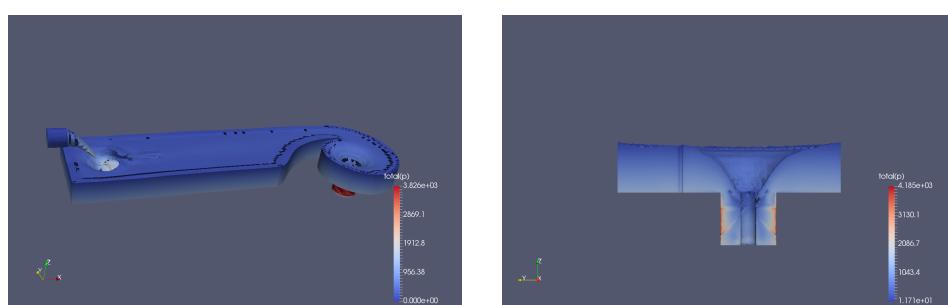
Water height initial condition



(a) Top view

(b) Vortex detail

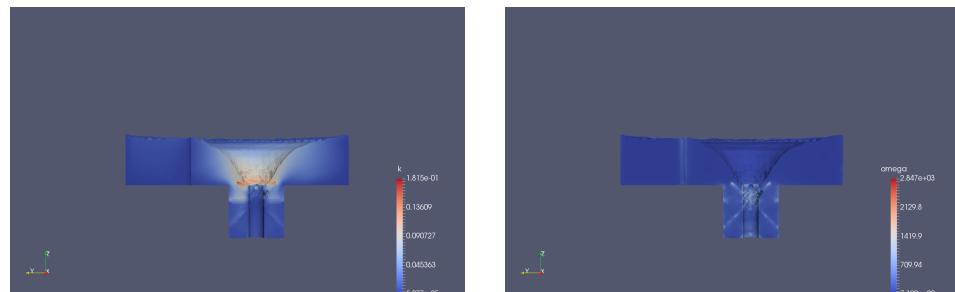
Velocity results



(a) Top view

(b) Vortex detail

Pressure results



Turbulence parameters results

```
4.0000000e+01 ((-2.8909669e-01 2.7233019e-01 -4.1481382e+00)
(-8.5059151e-02 -1.9525232e-02 -6.4268717e-01) (0.0000000e+00 0.0000000e+00
0.0000000e+00)) ((1.0808011e-02 2.2379479e-02 -1.1685382e-01)
(-1.7537186e-03 -3.0122566e-03 4.0219375e-02) (0.0000000e+00 0.0000000e+00
0.0000000e+00))
```

Torque and force results

Appendix D

Important commands on Ubuntu terminal

- **Open a virtual screen of the main server:** typing *tmux*
- **Reopen an ongoing simulation:** typing *tmux attach -t ‘name’*
- **Exit from the server:** typing *exit*
- **Detach an ongoing simulation:** pressing *ctrl+b* and then *d*
- **Copy a folder:** typing *scp -r ‘source’ ‘destination’*
- **Copy a file:** typing *scp ‘source’ ‘destination’*
- **Make a new directory:** typing *mkdir ‘name of the folder’*
- **Rename a tmux session:** pressing *ctrl+b* and, then *\$*
- **Run the simulation:** check Appendix H
- **Delete a directory:** typing *rm -r ‘name of the folder’*
- **Delete a file:** typing *rm ‘name of the file’*
- **List the current position in the server:** typing *ls*
- **Get in the previous directory:** typing *cd ..*
- **List the ongoing tmux session:** typing *tmux list-session*

- **Kill a simulation:** pressing *ctrl+c*
- **Stop a simulation:** pressing *ctrl+z*
- **Create a file:** typing *vi ‘name of the file’*
- **Move a file:** typing *mv ‘source’ ‘destination’*
- **Check the enclosed volume:** typing *surfaceCheck ‘name of the file’*
- **Check the mesh:** typing *checkMesh ‘name of the file’*
- **Merge files:** typing *cat ‘name1’ ‘name2’ ‘...’ >‘name merged’*

Appendix E

Other meaningful files for OpenFOAM

In all the following sections, the files used in OpenFOAM for Flatblades simulation are outlined.

E.1 *transportProperties*

```

FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// ****
phases (water air);

water
{ transportModel Newtonian;
    nu          nu [ 0 2 -1 0 0 0 ] 1.09e-06;
    rho         rho [ 1 -3 0 0 0 0 ] 998.8; }

air
{ transportModel Newtonian;
    nu          nu [ 0 2 -1 0 0 0 ] 1.48e-05;
    rho         rho [ 1 -3 0 0 0 0 ] 1; }

sigma        sigma [ 1 0 -2 0 0 0 ] 0;

```

In this file, it addressed the properties of water and air to consider in the model, focusing on the kinematic viscosity with *nu* in m^2/s and density with *rho* in kg/m^3 . Moreover, the *sigma* is set to zero because not taking part of the analysis.

E.2 *snappyHexMeshDict*

```

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    object       snappyHexMeshDict;
}
// ****
castellatedMesh true;
snap          true;
addLayers     true;

geometry
{inlet.stl {type triSurfaceMesh;name inlet;} back.stl {type triSurfaceMesh;name back;}
 bottom.stl {type triSurfaceMesh; name bottom;} top.stl {type triSurfaceMesh; name top;}
 left.stl {type triSurfaceMesh; name left;} right.stl {type triSurfaceMesh; name right;}
 neck.stl {type triSurfaceMesh; name neck;} basin.stl {type triSurfaceMesh; name basin;}
 outwall.stl {type triSurfaceMesh; name outwall; outlet.stl {type triSurfaceMesh; name outlet;}}
 hub.stl {type triSurfaceMesh; name hub;} blade1.stl {type triSurfaceMesh; name blade1;}
 blade2.stl {type triSurfaceMesh; name blade2;} blade3.stl {type triSurfaceMesh; name blade3;}
 blade4.stl {type triSurfaceMesh; name blade4;} blade5.stl {type triSurfaceMesh; name blade5;}
 blade6.stl {type triSurfaceMesh; name blade6;} mergvol.stl {type triSurfaceMesh; name mergvol;}
};
castellatedMeshControls
{
    maxLocalCells 500000;
    maxGlobalCells 1000000;
    minRefinementCells 10;
    maxLoadUnbalance 0.10;
    nCellsBetweenLevels 1;
features
{
    {file "inlet.eMesh";level 2;}
    {file "back.eMesh";level 2;}
    {file "neck.eMesh";level 2;}
    {file "basin.eMesh";level 2;}
    {file "bottom.eMesh";level 2;}
    {file "outwall.eMesh";level 2;}
    {file "outlet.eMesh";level 2;}
    {file "hub.eMesh";level 2;}
    {file "blade1.eMesh";level 3;}
    {file "blade2.eMesh";level 3;}
    {file "blade3.eMesh";level 3;}
    {file "blade4.eMesh";level 3;}
    {file "blade5.eMesh";level 3;}
    {file "blade6.eMesh";level 3;});
refinementSurfaces
{
    inlet {level (0 3);} back {level (0 2);} left {level (0 2);}
    right {level (0 2);} top {level (1 2);} bottom {level (0 3);}
    neck {level (1 3);} basin {level (2 4);} outwall {level (2 3);}
    outlet {level (0 3);} hub {level (0 3);}
    blade1
    {level (0 3); faceZone blade1; faceType baffle;}
    blade2
    {level (0 3); faceZone blade2; faceType baffle;}
    blade3
}

```

```

{level {0 3}; faceZone blade3; faceType baffle;}
blade4
{level {0 3}; faceZone blade4; faceType baffle;}
blade5
{level {0 3}; faceZone blade5; faceType baffle;}
blade6
{level {0 3}; faceZone blade6; faceType baffle;}}
resolveFeatureAngle 80;
refinementRegions
{mergvol {mode distance; levels ((0.04 1));}}
locationInMesh {0 0 0.4};
allowFreeStandingZoneFaces true;}
snapControls
{
  nSmoothPatch 3;
  tolerance 4.0;
  nSolveIter 30;
  nRelaxIter 5;
  nFeatureSnapIter 15;
  implicitFeatureSnap false;
  explicitFeatureSnap true;
  multiRegionFeatureSnap false;}
addLayersControls
{
  relativeSizes false;
  layers
  {
    neck {nSurfaceLayers 3;} basin {nSurfaceLayers 3;}
    outwall {nSurfaceLayers 3;} hub {nSurfaceLayers 3;}
    blade1 {nSurfaceLayers 3;} blade2 {nSurfaceLayers 3;}
    blade3 {nSurfaceLayers 3;} blade4 {nSurfaceLayers 3;}
    blade5 {nSurfaceLayers 3;} blade6 {nSurfaceLayers 3;}}
  expansionRatio 1.2;
  finalLayerThickness 0.008;
  minThickness 0.005;
  nGrow 0;
  featureAngle 80;
  nRelaxIter 3;
  nSmoothSurfaceNormals 1;
  nSmoothNormals 3;
  nSmoothThickness 10;
  maxFaceThicknessRatio 0.5;
  maxThicknessToMedialRatio 0.3;
  minMedianAxisAngle 130;
  nBufferCellsNoExtrude 0;
  nLayerIter 50;}
meshQualityControls
{
  maxNonOrtho 65;
  maxBoundarySkewness 20;
  maxInternalSkewness 4;
  maxConcave 80;
  minFlatness 0.5;
  minVol 1e-13;
  minTetQuality 1e-9;
  minArea -1;
  minTwist 0.02;
  minDeterminant 0.001;
  minFaceWeight 0.02;
  minVolRatio 0.01;
  minTriangleTwist -1;
  nSmoothScale 4;
  errorReduction 0.75;}
debug 0;

```

The features of this dictionaries are: preservation of featured edges, correct local meshing for rotating frame, fully parallel execution and implementation of additional wall layers.

[25]

First of all, the first three lines switch either on or off the steps to run, concerning to make the basic mesh, deciding to snap the back of surfaces and add viscous layer on them. All of them are activated. Secondly, in geometry brackets, the .stls files are loaded with their types and names.

In *castellatedMeshControls* the maximum amount of cells per CPU core, the maximum

amount of cells to use before mesh deletion steps, the minimum of bad cells allowed during the refinement stages, the expansion factor between low and high refinement zone are expressed; furthermore, under features the explicit edge and surface based refinement are specified. At the end, it resolves the sharp angles by spotting local curvature and refine it as well as type the refinement regions, whether to allow zone faces that share the same neighbour or owner cell zone and the cartesian point to retain the required volume mesh. Under *snapControls*, it mentions the settings for the snapping: number of pre-smoothing iterations before surface projection, scaling the maximum edge length, number of smoothing iterations applied and the control number of scaling back iterations for error reduction and, eventually, the number of snapping iterations to perform. Optional new commands are implemented under the main controls, mostly in order to detect errors in snaps. The final mesh steps is the *addLayersControls* which permits to add a specific set of boundary patches: initially, the final layer and minimum thickness and the .stls files to work on are typed, final layer thickness and expansion ratio are the ratio of height of adjacent surface and consecutive layer in the direction away from the surface and their grow.

There are also advanced features for the specification of feature angle above which layers are automatically collapsed, the smoothing of the normal surfaces, the definition of the medial axis used for mesh moving away from the surface, and, possibly, to reduce the layer thickness; at the end, layer iteration settings are addressed in case of not converging iterations. All along snappyHexMesh, the output quality is constantly monitored; under *meshQualityControls* the orthogonality, skewness, concavity of the faces, minimum face area and pyramid volume, the quality of the decomposition from cells to tetrahedra and from faces to triangular element, cell determinant, face weight metric, minimum face volume ratio, face triangle twist are checked. The vantage point of snappyHexMesh is to scale the mesh back locally so the the advanced features regards the scaling and the smoothing of displacement field. The mesh quality controls all the constrains and it might cause instability and/or divergence, any entries previously mentioned have different level of importance, e.g. non-orthogonality is way more important than face twist metric calculated and some of them can be give any error if not accomplished.

In snappyHexMesh, the rotating region is also mesh refined: under *refinementSurface*, it

is taken into consideration the blades of the turbine. For each of those, the minimum and maximum level of refinement, arrange the baffles type, which is a zero-thickness mesh objects on the blades and, the blade at which the baffle is applied.

E.3 *fvOptions*

```

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "system";
    object       fvOptions;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

MRF
{ type           MRFSource;
  active         true
  timeStart      5;
  duration       35;
  selectionMode cellZone;
  cellZone       MRFcellzone;
  MRFSourceCoeffs
  {
    active       true;
    origin       (0 0 0);
    axis         (0 0 1);
    omega        -25.13;
  }
}

```

For specifying direction and order of magnitude of the rotation and other features, it has been used the dictionary *fvOptions*. This dictionary concerns exactly the first simulation of Flatblades configuration.

The velocity of the runner is fixed at 240 rpm (equals to 25.13 rad/s) for any simulation considered. In this file, it specified the starting time (the *timeStart* was 0 for this simulation, increased to 5 sec for all the other simulations), total duration, how many axes of rotation are taken into consideration (by typing 0 or 1 to activate it) and the angular velocity, using the same selection modes as in *topoSet*.

E.4 *fvSchemes*

```

FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * *
ddtSchemes
{
    default      Euler;
}

gradSchemes
{
    default      cellLimited Gauss linear 1.0;
}

divSchemes
{
    default      none;
    div(rhoPhi,U)          Gauss linearUpwind grad(U);
    div(phi,alpha)         Gauss vanLeer;
    div(phib,phi)          Gauss linear;
    "div(phi,(k|omega)\\"   Gauss upwind;
    div((muEff*dev(T(grad(U))))) Gauss linear;
}

laplacianSchemes
{
    default      Gauss linear corrected;
}

interpolationSchemes
{
    default      linear;
}

snGradSchemes
{
    default      corrected;
}

fluxRequired
{
    default      no;
    p_rgh;
    pcorr;
    alpha.water;
}

```

fvSchemes is the file that gathered all the mathematical models to solve equations concerning operators and schemes. Similarly the other dictionaries, this file is in common among all the simulations and configurations.

E.5 *blockMeshDict*

```

FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * *
vertices
(
    (-2.2 -0.4 -0.2)
    (0.5 -0.4 -0.2)
    (0.5 0.5 -0.2)
    (-2.2 0.5 -0.2)
    (-2.2 -0.4 0.6)
    (0.5 -0.4 0.6)
    (0.5 0.5 0.6)
    (-2.2 0.5 0.6)
);
blocks
(
    hex (0 1 2 3 4 5 6 7) (135 45 40) simpleGrading (1 1 1) );
edges
();
boundary
{
    allBoundary
    {
        type patch;
        faces
        (
            (3 7 6 2)
            (0 4 7 3)
            (2 6 5 1)
            (1 5 4 0)
            (0 3 2 1)
            (4 5 6 7)
        );
    };
}

```

This is a sample of blockMesh tool in OpenFOAM applied to the Flatblades configuration. So, under *vertices* the 8 points, took as the extreme bounding volume is typed. The



negative and positive values are referred to the origin of the control volume (for any model, set at the top of the runner zone, in the middle of the runner upper face).

E.6 *controlDict*

```

FoamFile
{
    version      2.0;
    format       ascii;
    class        dictionary;
    location     "system";
    object       controlDict;
}
// * * * * *
application      interFoam;
startFrom         startTime;
startTime          0;
stopAt            endTime;
endTime            40;
deltaT             0.01;
writeControl      adjustableRunTime;
writeInterval     1;
purgeWrite        0;
writeFormat       ascii;
writePrecision    7;
writeCompression  compressed;
timeFormat        general;
timePrecision     6;
runTimeModifiable true;
adjustTimeStep    yes;
maxCo              0.5;
maxAlphaCo        0.5;
maxDeltaT          1;

```

Here, not entirely represented, it is the control panel for reading and writing the results, different values are managed in this dictionary. Beside the 1st simulation for the Flatblades model, all the other simulations are set as in the above picture, this edit has been done in order to speed up the running of the simulation which still take around 3 days implementing this.

Specifically, the most of the picture shown that the simulation start at time 0 at end at time 40 sec, *deltaT* sets the time step (0.01 sec) with adjustable time step and it writes results as many time step as it has. The application used for the simulation is interFoam as already explained in the Chapter 2.

E.7 *topoSet*

```

FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      topoSetDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

actions
(
    { name    MRFcells;
      type    cellSet;
      action   new;
      source   cylinderToCell ;
      sourceInfo
      {p1 (0 0 -0.135);p2 (0 0 0);radius 0.0855;}}
    { name    MRFcellzone;
      type    cellZoneSet;
      action   new;
      source   setToCellZone;
      sourceInfo
      {set MRFcells;}}
);

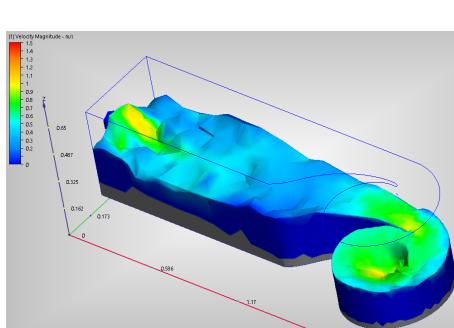
```

To specify the cells within the rotating region, it has been used the dictionary *topoSet*. As visible, the new region has been selected as for the option *cellSet* among 6 different choices; by doing so, a virtual cylinder to turn has considered and to adjust the size, 3 important points are defined: the center of the lower face, the center of the upper face and the radius.

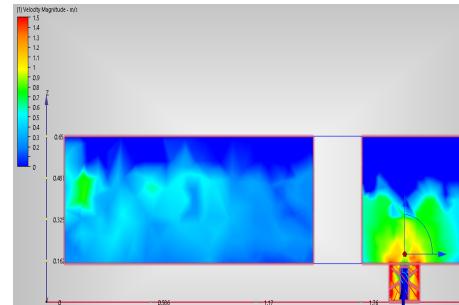
Secondly, it added the *cellZoneSet* to put the previously defined option in a zone now generated by *sourceInfo* brackets.

Appendix F

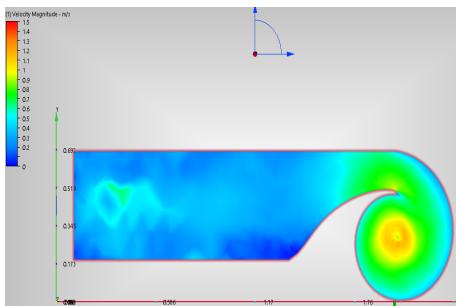
Autodesk CFD Simulations results



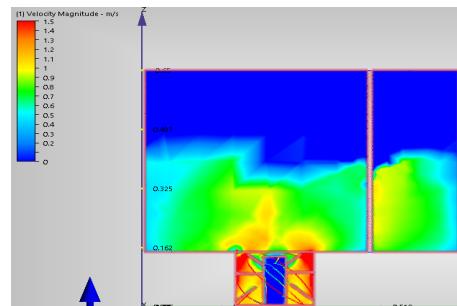
(a) Free surface view



(b) Plane x-z at the center of the runner



(c) Plane y-x few mms above the runner



(d) Plane x-z at the center of the runner

Autodesk CFD Simulation was the first software to implement CFD analysis; in the above figures, it shown the velocity values along the coordinates planes and the free surface after 60 seconds. These results are quite consistent with the OpenFOAM ones, less accurate though.

This software behaves differently compared to the one it used. The initial and boundary conditions, density, viscosity and all the other parameters are way easier to set and

the simulation is quite fast (around 1 day of time length). It is real-time visible the performance of the simulation, especially regarding the free surface stream, which allow to spot quickly if something went wrong during the configuration. A vantage point for this software is that it is possible to insert the material of the different parts of the model, thanks to a installed database and there is also the possibility to add a material which is not present in the database.

Due to problems with the torque results, no refinement on near-wall meshing and setting correctly the turbulence parameters, these results has been discarded and migrated to OpenFOAM software which is the best open-source to perform CFD on it.

Appendix G

Model .stl files

Flatblades	Streamlines
Back	Blade 1
Basin	Blade 2
Blade 1	Blade 3
Blade 2	Blade 4
Blade 3	Blade 5
Blade 4	Diffuser walls
Blade 5	Full basin
Blade 6	Hub
Bottom	Inlet
Hub	Outlet
Inlet	Rest
Left	Top
Neck	-
Outlet	-
Outwall	-
Right	-
Top	-

In the above table, it shown all the separated files by Meshmixer, the blades files are separated from the original model in a different way, because it is not concerning the

water-tight model.

Additionally, 2 files are created: *Runblades* and *MergeVol* for the purpose of checking the volume merging all the .stl files concerning the sides of the water-tight model as well as for the runner. To do so, an add-in of OpenFOAM software, in Ubuntu environment, has been used: Meshlab.

Furthermore, in some files in OpenFOAM Streamlines setup, there were additional patches called *blade 1_{slave}*, *blade 2_{slave}*, *blade 3_{slave}*, *blade 4_{slave}* and *blade 5_{slave}* because the blades of the turbine has 2 sides, the master indicated as in the table above and the slave ones additionally inserted.

Appendix H

Meshing and running simpleFoam in parallel with snappyHexMesh tool

1. **Rename 0 folder as 0.org:** to prevent snappyHexMesh interfering with it.
2. ***blockMesh*:** to generate a background mesh for snappyHexMesh.
3. ***surfaceFeatureExtract*:** to let snappyHexMesh knows where to snap to.
4. ***decomposePar*:** to divide the mesh up into 1 section per CPU core.
5. ***mpirun -np 12 snappyHexMesh -overwrite -parallel*:** to make the mesher runs in parallel.
6. ***reconstructParMesh -constant*:** to merge the mesh back together.
7. **Delete all the processor folders:** to clear up all the old mesh data.
8. ***topoSet*:** to run the sub-dictionary when and if required.
9. **Rename 0.org folder as 0:** reactivate the folder to use for the solver.
10. **Edit the 'boundary' file and remove all the references to patches created by *blockMesh* in Step2:** to leave only the patches desired for the simulation to run.
11. ***setFields*:** to run this sub-dictionary when and if required.
12. ***decomposePar*:** to put the solution setup into 1 folder per CPU core.

13. ***mpirun -n 12 renumberMesh -overwrite -parallel***: to optimise the mesh.
14. ***mpirun -np 12 interFoam -parallel***: to make the solver run in parallel, from this point the simulation run smoothly.
15. ***reconstructPar***: put the CPU results back together in one if a transient solver is running.
16. **Delete all the processor folders**: to clear up all the old mesh data.
17. ***paraFoam***: to visualise the results.