UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Design of a Low Cost Injection System for a Small Liquid Fuel Rocket Motor

Document:
Report

Author:
Adrià González Cano

Director – Co-director:
Jaume Solé Bosquet
Oriol Casamor Martinell

Degree:
Bachelor in Aerospace Vehicle Engineering

Examination session:
Spring, 2022

BACHELOR FINAL THESIS

Final Thesis for the degree of

Grau en Enginyeria en Vehicles Aeroespacials

DESIGN OF A LOW COST INJECTION SYSTEM FOR A SMALL LIQUID FUEL ROCKET MOTOR

Student: Adrià González Cano
Director: Jaume Solé Bosquet
Co-director: Oriol Casamor Martinell

ESEIAAT - Universitat Politècnica de Catalunya - BarcelonaTech

Spring 2022

This document contains: **REPORT**

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Acknowledgements

I would like to express my sincere gratitude to my parents for all their support and love, as well as for inculcating me the importance of hardworking and pursuing my aspirations, allowing me to become the person I am today. Also, special thanks to my brother who helped me in a difficult moment of the project.

To my directors: Jaume Solé and Oriol Casamor for their continued support and encouragement and for giving me the opportunity to work on this topic and use their facilities in SENER-NTE.

Looking back, we were the luckiest people in the world. There was no choice but to be pioneers; no time to be beginners.

Margaret H. Hamilton

Abstract

Rocket fuel injectors are in charge of propellant atomization and mixing processes, which determine the performance and stability of the liquid rocket engine itself. This small, yet significant components, are in charge of reducing the size and weight of the propellant droplets while mixing them in the shortest possible amount of time. The injector's design and implementation must be suitable for performing their tasks in order to achieve maximum performance of the nozzle.

Aiming to design and test a feasible rocket injector, a conceptual and physical design is presented, using cheap off-the-shelf stream jet nozzles, and its basic behaviour assessed. The numerical approach involves using computational fluid dynamics software (OpenFOAM) while the experimental approach involves lathe machining considerations. The numerical results indicate that the doublet injector design is feasible and mixture quality can be attained but still needs to be crafted and tested using a particular jet nozzle configuration.

Els injectors de motor coet estan a càrrec de l'atomització i els processos de mescla de combustibles, que determinen el rendiment i l'estabilitat del motor coet en si. Aquests petits components, encara que significatius, s'encarreguen de reduir la mida i el pes de les gotes dels reactius mentre els barreja en el temps més curt possible. El disseny i la implementació de l'injector han de ser adequats per dur a terme les seves tasques i aconseguir el màxim rendiment de la tovera.

Aspirant a dissenyar i provar un injector de coet factible, es presenta un disseny conceptual i físic, utilitzant broquetes d'esprai barats, i avaluant el seu comportament bàsic. L'enfocament numèric implica l'ús de programari de dinàmica de fluids computacional (OpenFOAM) mentre que l'enfocament experimental implica consideracions de maquinatge amb el torn. Els resultats numèrics indiquen que el disseny d'un l'injector doble és factible i la qualitat de la mescla es pot aconseguir, però encara s'ha d'elaborar i provar utilitzant una configuració de broquetes en particular.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Aim of the project

The aim of this project is to design, simulate and effectively test a low cost injection system for a relatively small liquid fuel rocket motor which is meant to use NOX as oxydiser and ethanol and/or parafine as fuel.

By definition, an injector in a liquid rocket engine is in charge of atomizing and mixing the fuel with the oxidizer to produce efficient and stable combustion that will provide the required thrust. It will be necessary to become familiar with injection system and mixture quality problematics, as well as simulation tools and proper prototype testing.

1.2 Scope of the project

The previous section defined the intention and main objective of the project. It is necessary to define a more detailed scope. The work will combine analytic tasks involving numerical simulations and application of formulae with an important part involving experimental set-up and testing of one or more prototypes.

The project will include the following aspects:

- Identification of the project's specific requirements and state-of-the-art study of rocket engines injection systems and different cheap off-the-shelf commercial sprayers, nebulizers, and 3D printer small extruding mouths.
- An extensive study of different areas of Computational Fluid Dynamics, such as fluid governing equations and turbulence models.
- An Open Source Computational Fluid Dynamics (CFD) software package called OpenFoam will be used to study the efficiency of the mixture varying parameters such as the injector geometries, flow velocity, and respective orientation. Different CFD models will be created with all the possible injection combinations.
- Construction of a test prototype to put into practice the results obtained numerically using the computational fluid analysis software OpenFoam.
- Machining considerations will be taken into account from the beginning so that fabrication costs are kept low.
- A experimental laboratory test with water injected into a transparent combustion chamber will be conducted in order to assess mixture quality.
- Evaluation and documentation of the results will be carried out during the project development.

1.3 Requirements

The following list identifies the requirements needed to achieve the aim of this project:

- RQ-1: The student shall have wide experience with CFD tools such as OpenFoam, together with willingness to carry out laboratory tasks.
- RQ-2: The simulations created with the CFD tools must reach convergence.
- RQ-3: The injection system must reach a mixture fuel/oxidizer homogeneous, verifiable with water injected jets (mixture uniformity must be achieved).
- RQ-4: The injection system must not get jammed during operation.
- RQ-5: The injection system must withstand the temperatures of the combustion chamber.
- RQ-6: The whole program shall not exceed 100€ cost in materials.

1.4 Justification

Liquid Rocket Engines (LRE) are very complex machines that require an immense amount of time to obtain the best performance possible and whose components have a high cost. Efficiency, stability and power are concerns of LRE and injectors are the main elements of combustion efficiency. Therefore, reducing the cost of injectors while maintaining their efficiency would be extremely beneficial for the aerospace industry.

Injectors are the main element in which propellants are supplied to the combustion chamber. Their proper implementation in liquid rockets determines the percentage of the theoretical performance of the nozzle that can be attained. The primary function of an injector is to provide atomization of the fuel. This atomization is a key factor in order to minimize the losses at the combustion chamber. A poor injector performance causes unborn propellant to leave the engine, resulting in poor efficiency. Facing this difficulties is key in order to improve the actual technology.

1.5 Background & State of the art

1.5.1 General Review

A liquid rocket engine employs liquid propellants which are supplied under pressure from tanks into a combustion chamber. The propellants usually consist of a liquid oxidizer and a liquid fuel. In the combustion chamber the propellants chemically react and burn to form hot gases which are then accelerated and ejected at high velocity through a nozzle, thereby imparting momentum to the engine. Momentum is the product of mass and velocity. The thrust force of a rocket motor is the reaction experienced by the motor structure due to the ejection of the high velocity matter. This is the same principle that makes a gun recoil when fired.[1]

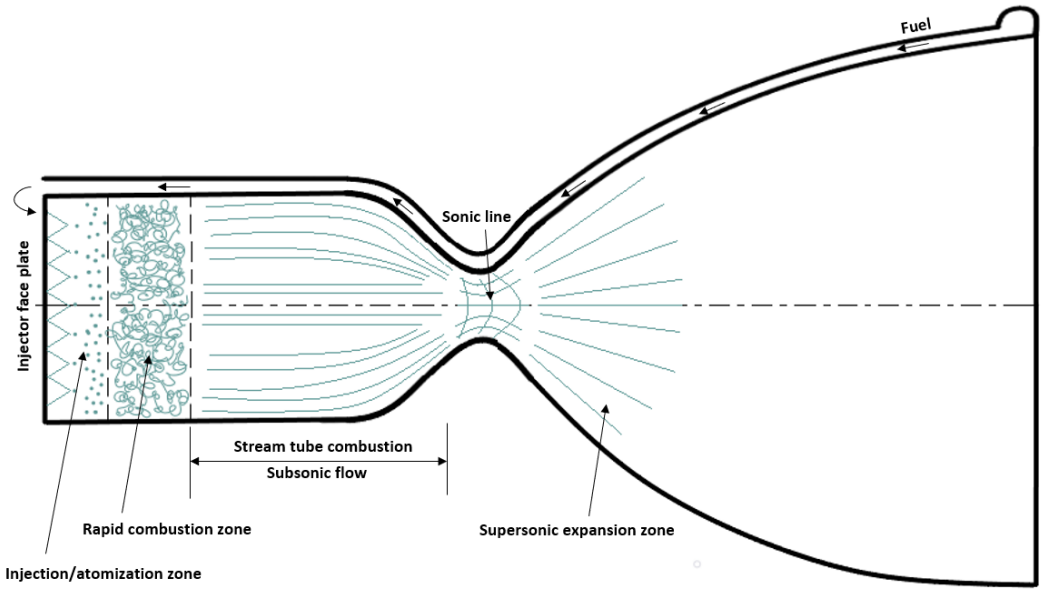


Figure 1 Scheme of a Liquid Fuel Rocket Engine, Source: Own

A scheme of a liquid rocket engine (LRE) chamber is given in Figure 1. It consists of the combustion chamber, where the burning of propellants takes place at high pressure, and the nozzle. The chamber must be strong enough to contain the high pressure and high temperature generated by the combustion process. The chamber must also be long enough to ensure complete combustion before the gas enters the nozzle.

At the beginning of the combustion chamber the injectors are carefully placed on the injector plate and they are in charge of mixing the fuel and the oxidizer in an atomization process. This element should be designed and manufactured carefully in order to provide an homogeneous atomization and to minimize losses at the combustion chamber. This project will be focused on the injection/atomization zone.

The nozzle's job is to transform the chemical-thermal energy produced in the combustion chamber into kinetic energy. The nozzle converts the combustion chamber's slow-moving, high-pressure, high-temperature gas into a high-velocity, lower-pressure, lower-temperature gas. Since thrust is the product of mass and velocity, a very high gas velocity is desirable.[1]

There exists a type of nozzles called DeLaval nozzles (named after their inventor) which perform the task of accelerating the exhaust gas amazingly well. They consist of a convergent and divergent section, as shown in Figure 1, where the minimum flow area between the two sections is called the nozzle throat and the flow area at the end of the divergent section is called the nozzle exit area. Depending on the operating height of the rocket engine, the nozzle may vary its length such that the pressure in the combustion chamber is reduced at the nozzle exit to a similar value of the pressure existing outside the nozzle.[1]

1.5.2 Advantages and Disadvantages of LREs

There exist many advantages as well as disadvantages regarding liquid-propellant rocket engines (LRE) over the use of hybrid-propellant rocket engines and solid-propellant rocket engines.

Advantages of LREs:

- LREs have throttle-ability to control them or in case of a malfunction. Shut down, restart and throttling are possible.
- LREs have higher Specific Impulse. They offer the highest energy per unit of fuel mass.
- The flow of propellants can be monitored and regulated to precisely control the magnitude of the thrust.
- Overall performance of LREs is higher than for solid propellant rockets because of the liquids' higher exhaust velocities.
- Several liquid oxidizers such as liquid oxygen, nitrogen tetroxide and hydrogen peroxide have better specific impulse than the ammonium perchlorate used in most solid rockets, when paired with comparable fuels.
- Development and operational costs are kept low.

Disadvantages of LREs:

- Need of complex storage containers. The oxidizer and the fuel must be stored separately which means that extra mass has to be carried by the launch vehicle.
- Several oxidizers are unstable, energetic and toxic.

- They require complex plumbing and precise fuel and oxidizer metering.
- Liquid-fuelled rockets require potentially problematic valves, seals, and turbopumps, which increase the cost of the launch vehicle. Turbopumps are particularly troublesome due to high performance requirements.
- Fueled rockets are difficult to store.

2 Injector Design

The function of the injector is to introduce the propellants into the combustion chamber in such a way that efficient combustion can occur. In liquid rockets, the injector implementation determines the percentage of the theoretical performance of the nozzle that can be achieved. A poor injector performance can cause unborn propellant to leave the engine, resulting in low efficiency. Thus, for a fast and efficient combustion, the propeller's injection should be homogeneous and result in many atomized droplets.

2.1 Injector Types

As previously mentioned, performance of the injection system has a considerable impact on the combustion system. Therefore, many studies on different injector configurations have been conducted over time.

The injector's design can be as simple as a series of small diameter holes arranged in carefully constructed patterns on top of an injector plate through which the fuel and oxidizer will flow. Injectors today classically consist of a number of small holes which aim jets of fuel and oxidizer in a collision trajectory at a point in space close to the injector plate. This impingement point is where the flow gets atomized into small droplets that can be burnt more easily.

The most common configurations are showerhead injectors (Figure 2) which were used in early rockets but do not generate great atomization or mixing. However, they are relatively easy to design and manufacture, meaning they have a wide use in amateur rocketry. The main problem with simple showerhead jets is that they travel in a straight line with little spread until they hit something, relying on turbulence produced on the combustion chamber to drive the mixing process.

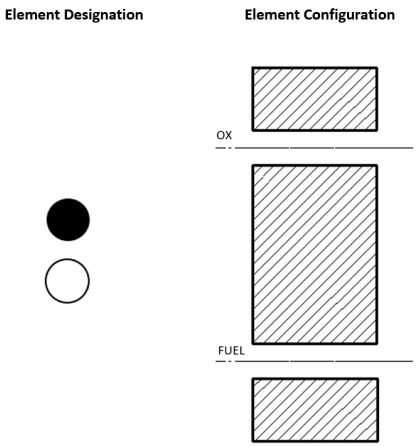
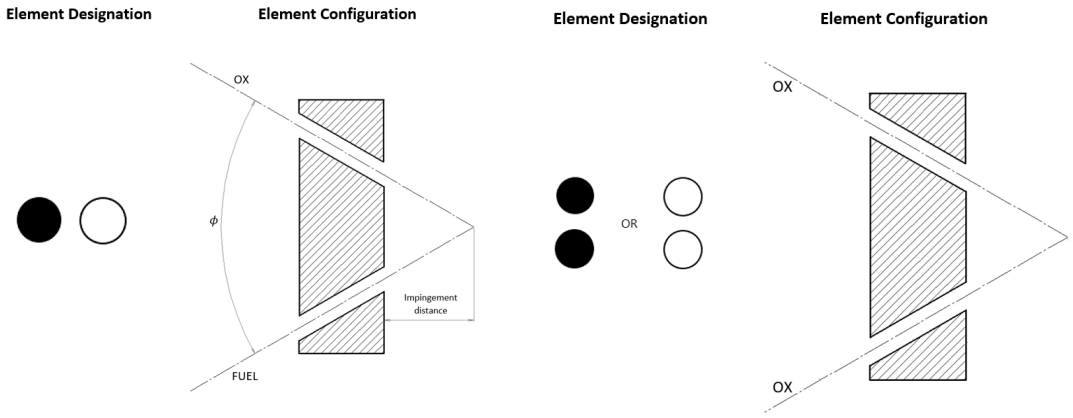


Figure 2 Schematic representation of the showerhead injector design, Source: Own.

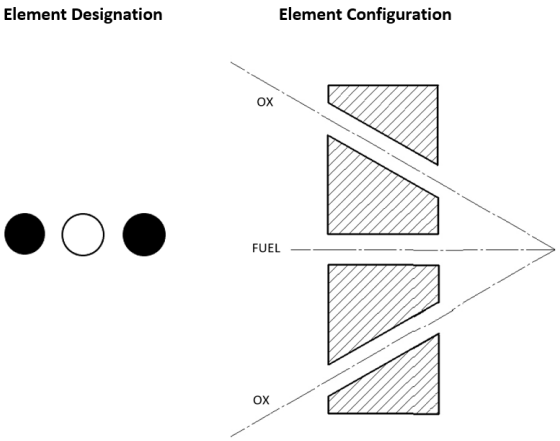
Many rocket designs improve the showerhead by using impinging injectors. In this type of injectors, the propellant jets are angled so that they collide with each other scattering the liquid over a wider angle and resulting in significantly smaller droplets. This means they provide a significant improvement on atomization and mixing. However, small holes are difficult to drill and they must be drilled with extreme accuracy, otherwise the injector performance would be affected. A disadvantage of this type of injector is that extremely small holes are required for small liquid fuel rocket motors which need small flow rates and the hydraulic characteristics and equations used to predict injector parameters do not give good results.[1]

There exist a lot of impinging jet injectors varying the number of jets involved and the way they collide and mix (Figure 3). Like-impinging jets refers to those that are the same type, for instance oxidizer-oxidizer or fuel-fuel. On the other hand, unlike-impinging jets accomplish mixing and atomization by direct impingement of fuel and oxidizer jets.

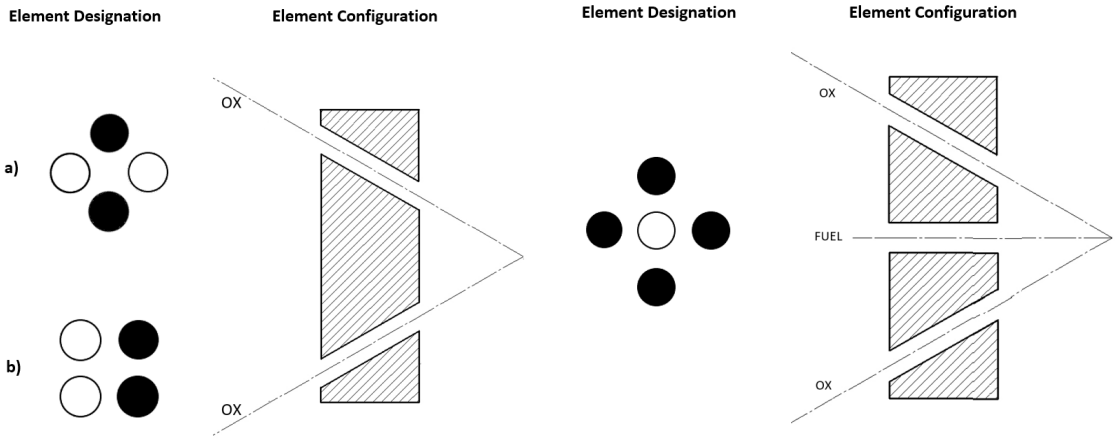


(a) 1-on-1 Unlike Doublet

(b) Like Doublet



(c) 2-on-1 Unlike Triplet



(d) Trans and Cis 2-on-2 Unlike Quadlet **(e)** 4-on-1 Unlike Pentad

Figure 3 Schematic representations of various kinds of symmetrical impinging-jet injector elements, Source: Own.

When two opposing, free liquid jets collide in the same plane but are not co-axially aligned, a liquid sheet forms in the plane perpendicular to their momentum vectors. Due to instabilities caused by aerodynamic forces such as air density and hydrodynamic forces caused by the jets' momentum exchange, this sheet will eventually break up. This break up destabilizes the sheet and causes ligaments to detach from the bottom periodically. These ligaments will continue to decrease in size and finally expel droplets; atomization has occurred. A representative image of the impinging jets and the liquid sheet can be found in figure 4[6].

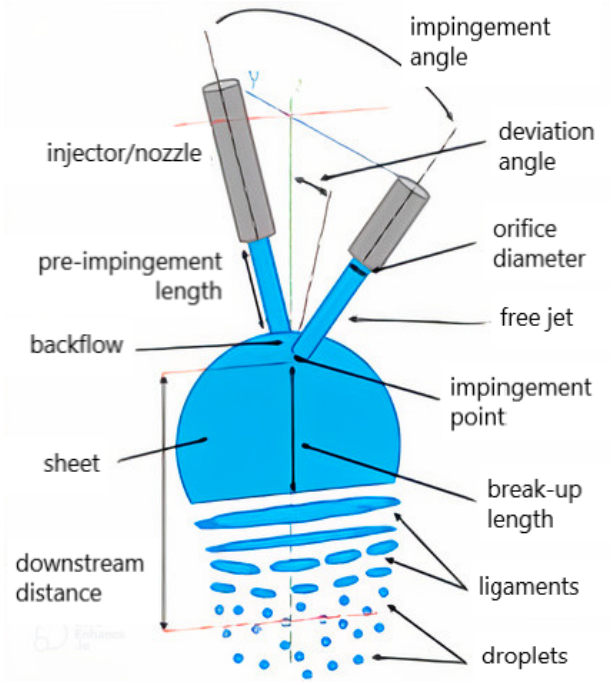
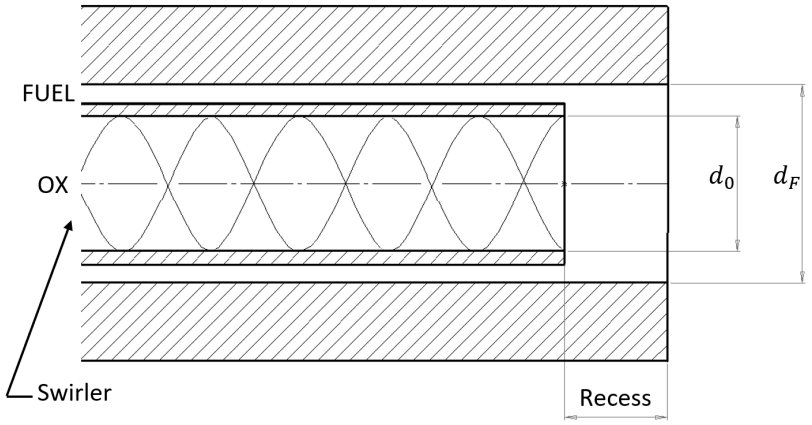


Figure 4 Doublet injector and sheet with typical terminology, Source: Vincent Notaro (2014)[6]

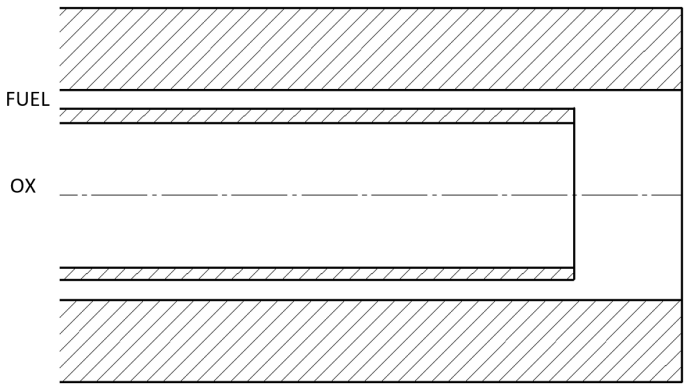
Sheet or spray type injectors are the other primary class of injectors, which are more popular today than jet kinds. These produce conical sheets that spread out and achieve atomization over a larger area. The swirl-type injector is an example of this sort of injector, which relies on the fluid's spinning velocity to spread out when it emerges from the orifice (see Figure 5). This type of injectors was used on the V-2 rocket with an interesting design.

Nowadays, they're most commonly seen in concentric tube injectors like the Copenhagen Suborbitals[32] design (see Figure 6). Their popularity stems from the fact that they are simple to manufacture and provide excellent mixing and performance. Their configuration can be seen in Figure 5.

Spray nozzles are especially appealing for the amateur builder since several companies manufacture them commercially for use in oil burners and other applications. It is only necessary to determine the size and spray characteristics required for the engine design and the correct spray nozzle can then be purchased at a low cost.



(a) Concentric Tube with Swirler



(b) Concentric Tube without Swirler

Figure 5 Schematic representation of two kind of concentric tube injectors, Source: Own.

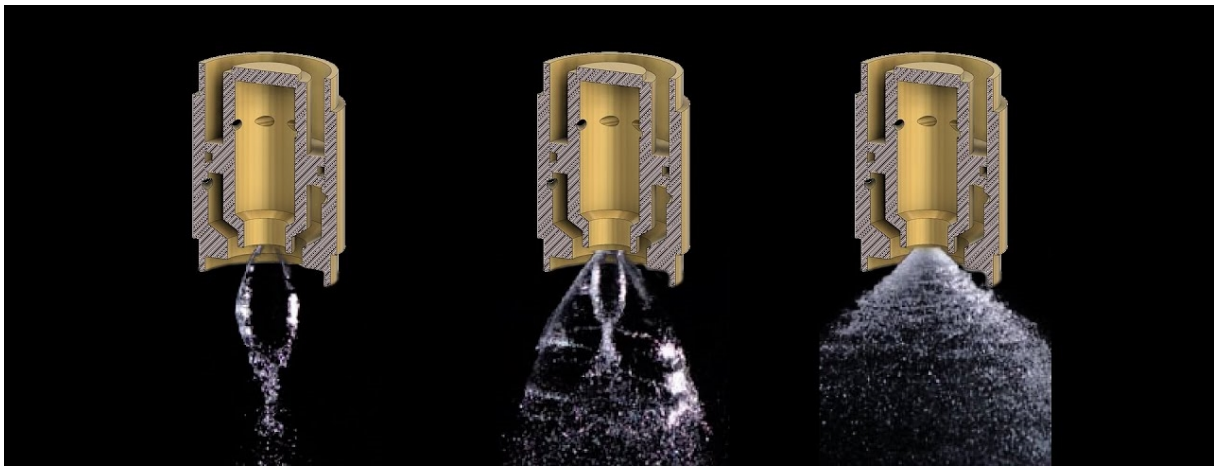


Figure 6 Coaxial swirl injectors design from Copenhagen Suborbitals, Source: [32]

On the other side, there exists a popular coaxial design known as pintle injector. Instead of swirling propellants it has an inverted cone structure at the top where the propellant flows up into it and then is pushed outwards. The main advantage of this type of injectors is that the pintle can be adjusted up and down, which means that the geometry of the injector can be modified according to the thrust needs.

This type of injector was used on the Apollo descent engine and in the Merlin engine on the Falcon 9. Both of them have a great throttle control and can use them as a valve, allowing to close up the injector and to seal the fuel flow while reducing leaks after the engines are shot down.

For the reasons stated before, impinging jets are very commonly used in rocket injector systems (Figure 3). For combustion systems, a set of doublet nozzles can be utilised as an injection interface. This configuration would simplify the design and analysis of our injection system and would provide an efficient way to introduce mixed and atomized fuel and oxidizer to the combustion chamber. For this reason, this study will focus on a pair of nozzles. Following, the nozzle geometry will be studied to meet our requirements.

2.2 Solid Stream Nozzles

The injector design for this project will be created with a simple device, solid stream nozzles. Solid stream nozzles, also known as solid jet nozzles, are the simplest of all nozzles, consisting just of a circular orifice at the end of a funnel or tube. They are precision devices that facilitate dispersion of liquid into a spray. Solid jet nozzles are used for three purposes: to distribute a liquid over an area, to increase a liquid surface area, and to create impact force on a solid surface. There exist a variety of design variations that result in different spray properties.

The great variety of nozzle configurations results in infinite possible combinations of doublets. The external nozzle variables include impingement length and impingement angle. Whereas the internal nozzle variables include, orifice diameter, orifice shape, orifice length-to-diameter ratio, orifice entrance conditions (e.g. round, sharp, or bevel edged), and orifice length roughness. The external variables define the impingement properties whereas the internal variables generate the shape and size of the free liquid jet (e.g. spray angle, spray pattern, etc.). Figure 7 is a diagram of the internal nozzle geometry variables of a typical edged nozzle. [6]

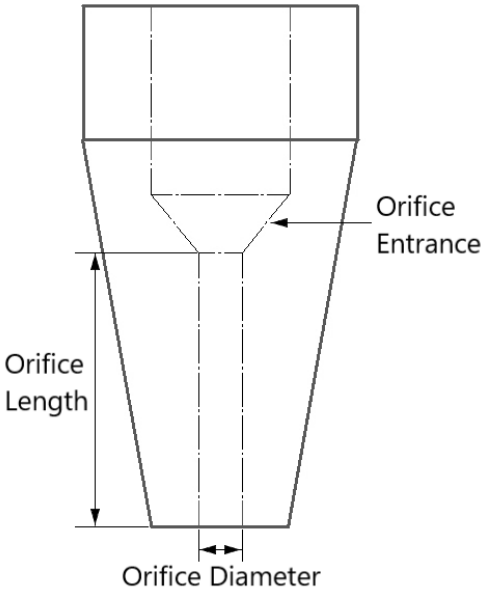


Figure 7 Edged nozzle with typical terminology, Source: Own

2.3 Mixing Mechanisms of Impinging Jets

Given that our design involves a pair of nozzles, this chapter discusses the mixing mechanisms that may occur when two jets collide. Ashgriz et al. [5] and Robert W. [43] provide an explanation of the mixing mechanism occurring in impinging jets and the criteria for optimum mixing. It is shown that mixing occurs in two regimes: pre-atomization and post-atomization.

The first one involves the fluid from the exit of the orifices up to the break-up of the sheet. There exist three atomization modes, which are: reflective, well mixed, and transmitive (see Figure 8).

Reflective atomization is the result of the liquid jets bouncing off from one another, while transmitive atomization is the result of both free stream jets being crossed and, as a result, the fluids flow to the opposite side of their injectors. On both of these modes, the post-atomization regime's mixing mechanism involves turbulent dispersion, which is often a welcome ingredient

of a process where efficient mixing is required. Well mixed atomization can be understood as a balance between the reflective and the transmitive modes. Regarding mixture quality, it is the best scenario.

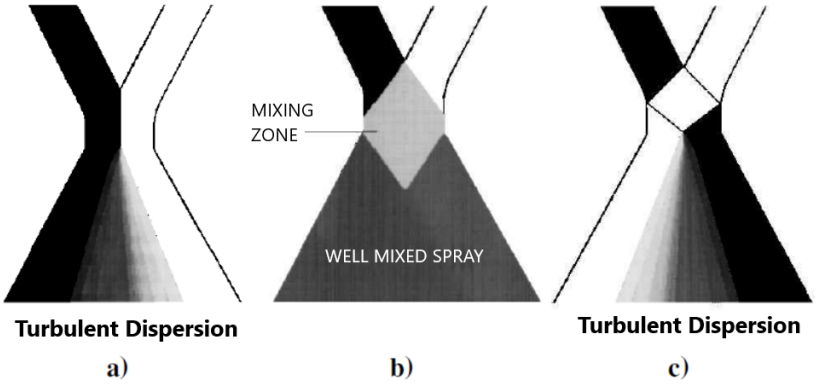


Figure 8 Schematic of atomization modes: a) reflective, b) well mixed, c) transmitive atomization.

The authors from reference [5] find that an increase in jet velocity, jet diameter, and impingement angle reduce the time needed to redirect the flow, i.e. they provide better and faster mixing. In addition, they show how most practical impinging jets operate in transmissive atomization.

2.4 Injector's Design Parameters

For many years, dropsize characteristics for unlike-impinging jets were empirically obtained through physical parameters such as orifice size and injection velocity. The unlike-impinging doublet is the most common element used for storable-propellant engines. As explained in the previous section, this element is composed of oxidizer and fuel jets that impinge at a given angle and at a given distance from the injector plate. The most commonly used impingement angle is 60^{o} [2]. A schematic of most typical unlike impinging-jet injectors is provided in figure 3. Some of the early studies aimed at quantifying drop size generated by different liquid propellant unlike-impinging elements are described in reference [38].

The purpose of this study was to experimentally investigate poorly understood aspects of the atomization mechanism and to expand and improve the already known dropsize correlations. The study related parameters including orifice shape and dynamic pressure ratio of the impinging jets for unlike doublets and it resulted in the most thorough and detailed dropsize study for liquid/liquid propellants to date.

It is proven that orifice diameter ratio, orifice size, impingement angle, and impingement

distance affect the spray distributions for unlike-impinging elements. The results of this program have helped to provide a more basic understanding of the parameters that have a substantial impact on the atomization process.

2.4.1 Orifice Diameter Ratio

Numerous studies, e.g. reference [40], have shown that the diameter of the element orifice and the ratio of the diameters of the oxidizer and fuel orifices d_0/d_f affect the injector's mixing and atomization levels produced by the injector. For unlike-impinging injector types orifice diameter ratio has a significant impact on the level of mixing that can be achieved. For all element types, small orifices consistently outperform larger orifices, because the smaller droplet sizes result in increased vaporization rates and because mixture ratio is more nearly uniform.

For biliquid propellants unlike-impinging doublet elements, the mixing correlation for circular orifice geometry presented in reference [40] indicates that a specific diameter ratio is required for optimum mixing and that this ratio depends only on the propellant density and flowrate ratios. This correlation has been verified only up to a diameter ratio of 1.5. Experiments have shown that when the diameter ratio d_o/d_f is significantly different from 1.22, the level of mixing attainable with an unlike-impinging is greatly reduced.

There exists a correlation relating the orifice area ratio for maximum mixing efficiency for all of the above unlike-impinging elements when designed for an included impingement angle of 60º. This correlation can be written (adapted from reference [43]):

$$\left(\frac{d_c}{d_{ou}}\right)^2_{MME} = M\left[\frac{\rho_{ou}}{\rho_c}\left(\frac{\dot{\omega}_c}{\dot{\omega}_{ou}}\right)^2\right]^{0.7} \tag{1}$$

Where:

d_c = diameter of the centre orifice.[1]

d_{ou} = diameter of the outside orifice.

M = mixing factor determined from experiment (typical values given in table 1).

ρ = liquid density.

$\dot{w}_c$ = total mass flowrate through all center orifices.

[1] for 1-on-1 and 2-on-2 the "centre" orifice is assigned arbitrarily

$\dot{w}_{ou}$ = total mass flowrate through all outside orifices.

Subscript MME = maximum mixing efficiency.

Element Type	Mixing Factor (M)
1-on-1, 2-on-2	1.0
2-on-1	1.6
3-on-1	3.5
4-on-1	9.4
5-on-1	27.5

Table 1 Values of mixing factor (M) for several types of elements

For a given propellant combination, the flowrate and density ratios are fixed and the only remaining independent variable in Equation 1 is M, a constant defined for a specific element type from Table 1. Therefore, after selection of propellants, element type, and flowrate, Equation 1 is used to define the diameter ratio d_c/d_{ou} that produces optimum mixing.

2.4.2 Impingement Angle

The angle of impingement between impinging jets affects propellant backsplash, mixing uniformity, and atomization characteristics. Injector backsplash on the injector plate can result in injector-face burnout. This parameter's importance depends on the element type.

For unlike-impinging elements, the greater the impingement angle, the greater the quantity of mixed propellant flowing back toward the injector plate. This phenomenon is illustrated in Figure 9.

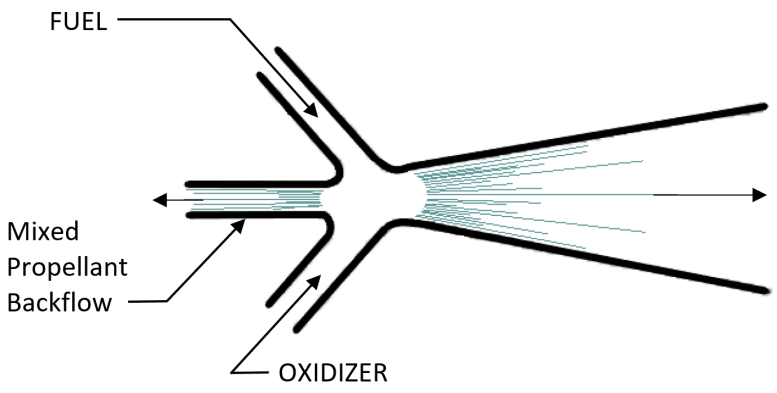


Figure 9 Two-dimensional flow field for a typical unlike-impinging element, Source: Own

It can be easily seen that the quantity of mass flowing backward is proportional to the cosine of the impingement half-angle. In addition, the angular distribution of the atomized spray leaving the impingement point is also dependent on the impingement angle; the greater the angle, the greater the mass and droplets distributed/atomized at angles greater than 90º. Experiment has shown that impingement angles greater than 90º result in high heat flux to the injector face.[2]

Mixing uniformity is also affected by the impingement angle. Qualitative experimental evidence has shown that the degree of mixing attained with biliquid injectors has a considerable effect upon rocket-motor performance. The results show that over the range of impingement angle from 80º to 40º, mixing increases as the impingement angle decreases. Most unlike-impinging injectors have been designed with impingement angles of 60º.[2]

For like-impinging doublets, backsplash is not as serious a problem as it is with unlike elements, because the propellants are not physically mixed until the spray clouds intermix. Consequently, the propellants that strike the face are not burning.

Other elements such as non-impinging elements, e.g. showerhead or concentric-tube injectors, and hybrid elements, e.g. pintle injectors, use equivalent impingement angles which are extremely important to their design.[2]

2.4.3 Impingement Distance

The impingement distance is the length of the jet measured along the jet centerline from the orifice exit to the point of impingement. Regardless of the impinging element type, long free-stream jet lengths result in the impingement of streams that are already partially fragmented or atomized, leading in jet misimpingement.

Misimpingement has a significant impact on mixing uniformity; for example, a little change in jet centerline direction can result in a significant change in mixing uniformity. In addition, stream misimpingement is amplified by the ratio of free-stream length to orifice diameter l_{fs}/d_o. Large values for this ratio ($\geq$ 10) magnify the effects of stream misimpingement that can be caused by a poor orifice geometry.[2]

3 Computational Fluid Dynamics Analysis

3.1 Concept of Computational Fluid Dynamics

Computational Fluid Dynamics (CFD)[3] is the simulation of fluids engineering systems using mathematical physical problem formulation and numerical methods including discretization methods, solvers, numerical parameters, grid generations, etc. The CFD process is described in figure 10.

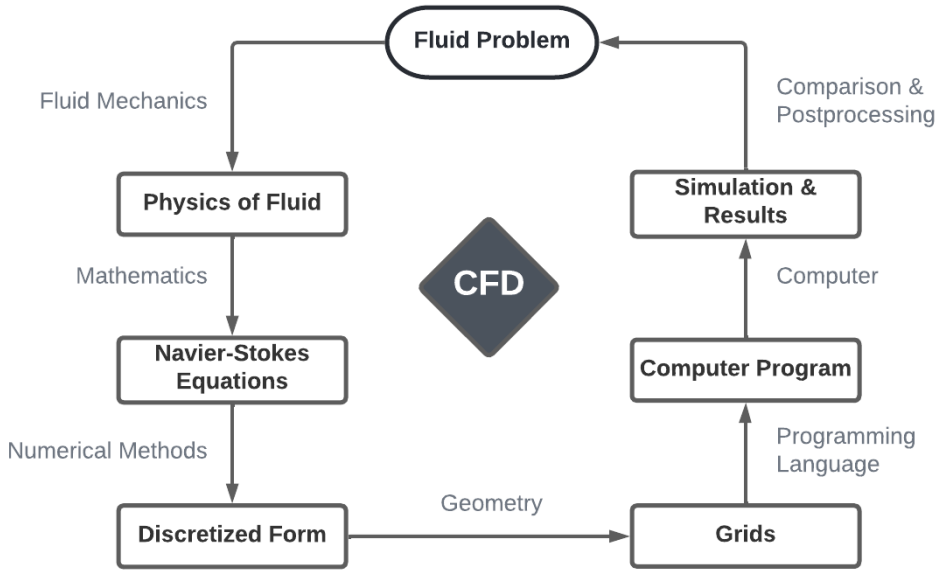


Figure 10 Process of Computational Fluid Dynamics, Source: Own

First and foremost, we have a fluid problem. In order to solve this problem, we need to know the physical properties of the fluid by using fluid mechanics. Then, these physical properties can be described by using mathematical equations in order to obtain the Navier-Stokes formulae, which are the governing equations of CFD. However, if we want to solve these equations by computer, we must first convert them to the discretized form. This procedure is typically used as a first step in preparing these equations for numerical evaluation and implementation on digital computers. As a result, we also need to divide our hole geometry domain into multiple small pieces where the discretized equations will compute the different variables. Then, multiple programs can be written to solve the equations in each grid cell. These programmes are typically executed on workstations or supercomputers. Finally, we can obtain our simulation results and, using a variety of postprocessing techniques, compare them to our original problem with experiments. Iteration is a key factor in order to get satisfactory results; if the results are insufficient to address the problem, the procedure must be repeated until a satisfactory solution is found.

3.2 Importance of Computational Fluid Dynamics

CFD is best employed in cases where the system behaviour cannot be computed using conventional calculation, not necessarily due to the complexity of the mathematical equations, but because of the complexity of the overall system geometry. When compared to experiments, CFD has numerous advantages:

(a) **Cost:** A CFD analysis involves only computational power (electricity) and engineer brain matter, making it significantly less expensive than, say, wind tunnel experiments, which require the creation of physical models of the testing system. However, CFD Open Source softwares are not free; they require a deep understanding of the subject, which can be extremely time consuming.

(b) **Flexibility:** The same way making a prototype is costly, adjusting a physical model is far more complex than just quickly changing parameters of a computer-generated model.

(c) **Information:** The final analysis can be much deeper and detailed than with experimental testing. Multiple postprocessing tools can be used to help visualize the resulting data of the simulation. Forces, streamlines, and even the pressure field around the object of study are some examples of data that can be gathered by CFD that would be difficult to obtain in real-world experiments.

(d) **Scale and precision:** The precision of a CFD simulation can be absurd as these techniques have been refined over years. Any simulation is achievable with enough computational resources, no matter how small or large the scale is. Experimental analysis is typically used to validate (i.e. to prove the CFD model is indeed correct) the results obtained with computer simulations.

3.3 Physics of Fluid

A fluid is any liquid or gas, or generally any material that cannot withstand a tangential or shearing force when at rest and changes shape continuously when subjected to such a stress. When under shear stress, flow is a characteristic property of fluids that involves the irreversible change in position of one part of the material relative to another. Elastic solids, on the other hand, can sustain shearing forces, holding and twisting without flowing and recovering their original shape. Fluids have many properties, such as velocity, pressure, temperature, density and viscosity.

The density, equation (2), of a fluid is defined as the ratio mass/volume. If the density of the fluid is constant, we can say that the fluid is incompressible, whereas a non-constant density leads to a compressible fluid. Generally, water and air are treated as incompressible fluids so that the final equations can be simplified.

$$\rho = \frac{M}{V} \quad \left[\frac{kg}{m^3}\right] \tag{2}$$

The viscosity, equation (3), is an internal property of the fluid that offers resistance to flow, i.e., the resistance of the fluid to change in shape.

$$\mu = \left[\frac{Ns}{m^2}\right] \tag{3}$$

Other macroscopic properties of the flow are the pressure (p), the temperature (T) and the velocity (v). In chapter (3.4) these properties will be combined to deduce the governing equations of fluid motion.

Based on the previous characteristics of fluids, fluid flow is divided into the following types:[29]

- **Steady Flow:** A flow is said to be steady flow if different fluid properties and fluid velocity does not change with respect to time.
- **Unsteady flow:** A flow is said to be unsteady flow if different fluid properties and fluid velocity changes with respect to time.
- **Compressible Flow:** It is the flow in which density of fluid varies with respect to change in pressure.
- **Incompressible Flow:** It is the flow in which density of fluid does not vary with respect to change in pressure. All fluids are compressible, even water, thus, their density will change as pressure changes. Under steady conditions, and provided that the changes in pressure are small, it is usually possible to simplify analysis of the flow by assuming it is incompressible and has constant density. Gasses, on the other hand, are very easily compressed; it is essential in most cases to treat them as compressible, taking changes in pressure into account.
- **Laminar Flow:** In a laminar flow, the flow of fluid is smooth and highly ordered. In this flow, the fluid layers move parallel to each other and do not intersect.
- **Turbulent Flow:** In a turbulent flow, the flow of fluid is chaotic and not in any order. In this flow, the fluid layers cross each other and do not move parallel.
- **Internal Flow:** When the fluid flows inside a closed surface like pipe or duct.
- **External Flow:** When it does not flow inside any closed surface or flows on the outer surface of a body.
- **Viscous Flow:** It is a flow in which the resistance provided by viscosity is considerable. In this type of flow, the frictional effects are significant.
- **Inviscid Flow:** It is a flow in which the resistance provided by viscosity is negligible. In this type of flow, the frictional effects are not significant.

Dimensionless or non-dimensional numbers are important in fluid mechanics when determining the flow properties of a fluid. Knowing that inertia force always exists if there is any mass in motion, these dimensionless numbers are obtained dividing the inertia force by other forces such as viscous force, gravity force, surface tension, elastic force, or pressure force. [10]

Some important dimensionless numbers used in fluid mechanics and their importance is explained below:

3.3.1 Reynolds Number

The Reynolds number is the ratio of inertia force to the viscous force. It describes the predominance of inertia forces to the viscous forces occurring in the flow systems. It is defined as:

$$R_e = \frac{\text{Inertia Forces}}{ViscousForces} = \frac{\rho \cdot U \cdot L}{\mu} \tag{4}$$

Where:

L = characteristic length scale of flow (m)

The Reynolds number plays an important part in calculations in fluid dynamics and heat transfer problems for many reasons; it is essential for calculating the friction factor (f)[2] and for heat transfer calculations. It is also used to predict the transition from laminar to turbulent flow, as well as the scaling of similar but different-sized flow situations, such as between an aircraft model in a wind tunnel and the full-scale version of it.[14]

3.3.2 Froude Number

The Froude number is the ratio of inertia force to the gravitational force. Froude number is significant in case of free surface flows where the gravitational force is predominant compared to other forces. It is defined as:

$$F_r = \frac{\text{Inertia Forces}}{GravitationalForces} = \frac{U}{\sqrt{g \cdot L}} \tag{5}$$

[2]The Darcy Friction Factor is a dimensionless number that allows to calculate the frictional energy loss in a pipe using the fluid velocity and friction resistance. It is used almost exclusively to calculate head loss due to friction in turbulent flow. It is usually selected from a chart known as the Moody diagram, a family of curves that relate the friction factor, f, to the Reynolds number, Re, and the relative roughness of a pipe, ε/D. [19]

3.3.3 Weber Number

The Weber number is the ratio of inertia force to the surface tension. The formation of droplets or water bubbles in a fluid is normally due to surface tension. If Weber number is small, surface tension is larger and vice versa. It is defined as:

$$W_e = \frac{\text{Drag Forces}}{\text{Cohesion Forces}} = \frac{\rho \cdot d \cdot U^2}{\sigma} \tag{6}$$

Where:

d = characteristic length, typically the droplet diameter (m).

σ = surface tension of fluid (N/m).

3.4 Navier-Stokes Equations

In physics, the Navier-Stokes equations are a group of partial differential equations which describe the motion of viscous fluid substances and serve as the governing equations of Computational Fluid Dynamics. They are named after French engineer and physicist Claude-Louis Navier and Anglo-Irish physicist and mathematician George Gabriel Stokes and were developed over several decades of progressively building the theories, from 1822 (Navier) to 1842–1850 (Stokes).

The Navier-Stokes equations are based on the conservation law of physical properties of fluid, mathematically expressing conservation of momentum and conservation of mass for Newtonian fluids. They are sometimes followed by an equation of state relating pressure, temperature and density.[37]

Certain mechanical qualities of a system cannot change if it does not interact with its environment in any way. They are sometimes referred to as "motion constants." These quantities are said to be "conserved," and the conservation laws that follow can be considered the most fundamental principles of mechanics. Energy, mass, momentum, and angular momentum are examples of conserved quantities in mechanics. [16]

Applying the mass, momentum and energy conservation, we can derive the continuity, the momentum and the energy equations.

3.4.1 Continuity Equation

The first of the Navier-Stokes Equations is also known as Continuity Equation.

Knowing that mass cannot be created, destroyed, nor transformed and that mass diffusive transport does not occur, the Continuity Equation can be written as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho U) = 0 \tag{7}$$

Equation 7 is the general form of the Continuity Equation. However, when flow can be considered compressible, density becomes constant and so can be removed from the equation. This leads to the incompressible form of the first of the Navier-Stokes Equations:

$$\nabla \cdot U = 0 \tag{8}$$

3.4.2 Momentum Equation

The second of the Navier-Sotkes Equation is another way to formulate Newton's Second Law ($F = ma$) and stands as:

$$\frac{\partial U}{\partial t} + (U \cdot \nabla)U = \nabla \cdot (\lambda \nabla \cdot U) + \nabla \cdot Q_S + Q_V \tag{9}$$

External forces are those acting in proportion to a given fluid body volume and so they will take the place of volume sources/sinks (Q_V) in the conservation law. Examples of this kind of forces are gravity or electromagnetic forces.

The term that appears in the majority of cases is gravity. We will use the symbol f_b for body forces per unit mass. Therefore:

$$Q_V = f_b$$

Internal forces act in proportion to the surface area of a fluid body, hence they will replace surface source/sinks (Q_S) in the conservation law. The only example of this type of force are internal stresses and their treatment is more complex compared to external forces.

When talking about stresses it is necessary to state the hypothesis of Newtonian fluid; these fluids display a linear relation between viscosity and shear stress and their viscosity remains constant, no matter the amount of shear applied for a constant temperature.

The internal stress tensor (σ) can be decomposed into:

$$\sigma = -p\mathbf{I} + \tau$$

where p is pressure, which acts perpendicularly to the fluid body sufaces (isotropic), $\mathbf{I}$ is the identity tensor and τ is the viscous shear stress tensor, constituted by all the viscous stresses acting in all possible directions tangent to the fluid body surface.

For an incompressible and isotropic Newtonian fluid, the viscous stress is proportional to the strain rate, i.e. the velocity gradient in the direction perpendicular to the wall, it can be written as:

$$\tau = \mu \frac{dU}{dy} \tag{10}$$

When transposed to a three-dimensional Cartesian system, if the fluid is incompressible and viscosity is constant across the fluid, this equation can be written as:

$$\tau = \mu \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \tag{11}$$

The viscous terms in τ appear as second derivatives, so they can be simplified using the Laplacian operator (∇^2). This leads to the incompressible form of the second of the Navier-Stokes Equations:

$$\rho \frac{\partial U}{\partial t} + \nabla \cdot (\rho U U) = -\nabla p + \mu \nabla^2 U + \rho f_b \tag{12}$$

The same procedure can be applied to obtain the third of the Navier-Stokes equations, the Energy Equation.

The standard form of the transport equation for a scalar property ϕ is:[36]

$$\underbrace{\frac{\partial \rho \phi}{\partial t}}_{\text{Temporal Derivative}} + \underbrace{\nabla \cdot (\rho U \phi)}_{\text{Convection Term}} - \underbrace{\nabla \cdot (\rho \Gamma_\phi \nabla \phi)}_{\text{Diffusion Term}} = \underbrace{S_\phi(\phi)}_{\text{Source Term}} \tag{13}$$

When $\phi = 1, U, T$, the continuity, momentum and energy equations can be respectively obtained.

3.4.3 Closure Problem

At this point, it should be noted that the Navier-Stokes equations have an issue. If we consider equations (8) and (12), we have five unknown variables:

- Pressure (p)
- Density (ρ)
- Velocity in x (U_x)
- Velocity in y (U_y)
- Velocity in x (U_z)

However we only have a system of four equations (14); the mass conservation equation and the three momentum conservation equations (one for each cartesian direction: x, y and z):

$$\begin{cases} \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} = 0 \\ \rho\left(\frac{\partial v_x}{\partial t} + v_x\frac{\partial v_x}{\partial t} + v_y\frac{\partial v_x}{\partial t} + v_z\frac{\partial v_x}{\partial t}\right) = -\frac{\partial P}{\partial x} + \mu\left(\frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2} + \frac{\partial^2 v_x}{\partial z^2}\right) + \rho f_{b_x} \\ \rho\left(\frac{\partial v_y}{\partial t} + v_x\frac{\partial v_y}{\partial t} + v_y\frac{\partial v_y}{\partial t} + v_z\frac{\partial v_y}{\partial t}\right) = -\frac{\partial P}{\partial y} + \mu\left(\frac{\partial^2 v_y}{\partial x^2} + \frac{\partial^2 v_y}{\partial y^2} + \frac{\partial^2 v_y}{\partial z^2}\right) + \rho f_{b_y} \\ \rho\left(\frac{\partial v_z}{\partial t} + v_x\frac{\partial v_z}{\partial t} + v_y\frac{\partial v_z}{\partial t} + v_z\frac{\partial v_z}{\partial t}\right) = -\frac{\partial P}{\partial z} + \mu\left(\frac{\partial^2 v_z}{\partial x^2} + \frac{\partial^2 v_z}{\partial y^2} + \frac{\partial^2 v_z}{\partial z^2}\right) + \rho f_{b_z} \end{cases} \tag{14}$$

This results in a mathematical closure problem that, unfortunately, cannot be addressed by coupling the energy conservation equation (the third of the Navier-Stokes Equations), as this would add more unknown variables (e.g. temperature)[24]. This is the reason why the solution to the Navier-Stokes Equations is still unknown. In other words, we do not even know if a solution actually exists. In fact, the Clay Mathematics Institute made this one of its Millennium Problems[25], whose solution will be awarded with 1,000,000$.

The key to this closure problem is turbulence. In a turbulent flow, each of these quantities may be decomposed into a mean and a fluctuating part, Averaging the equations results in the Reynolds-averaged Navier-Stokes (RANS) equations, which will be further explained in chapter 3.7.

3.5 Finite Volume Method

3.5.1 The approach of Finite Volume Method

The Navier-Stokes equations are analytical equations. If we want to solve them by computer, we have to convert them into discretized form. This process is called discretization. Finite difference (FDM), finite element (FEM), and finite volume methods (FVM) are the most common discretization approaches. Following, the Finite Volume Method will be introduced as it is the most common method for CFD solvers.[20]

This Method arises from the necessity to transform the set of partial differential equations, the Navier-Stokes equations (14), into the following matrix form:

$$MU = B \tag{15}$$

Where M is a matrix of coefficients, U is the unknown velocity vector, i.e. the velocity of every cell in the mesh, and B is a right-hand side term. Once the equations are written in this linearized matrix form, a variety of different linear equation solvers can be used to solve the velocity field. The main advantage of using the Finite Volume Method (FVM) is that there are no restrictions on the user over what type of mesh can be created with polyhedral cells.[37]

Recalling equation (12) and its simplified form in equation (13), this is a second-order equation, as the diffusion term includes the second derivative in space. For good accuracy, it is necessary for the order of the discretisation to be equal, or higher, than the order of the equation that is being discretised. The discretisation practice adopted in Jasak's study (1996) [36] is second-order accurate in space and time, thus it will be presented in the rest of this Chapter. The individual terms of the transport equation will be treated separately.

The FVM discretization method assumes that the flow variables (p,T,U, etc.) are stored at each cell centroid and vary linearly across the cell, i.e. it is a second order method. This cell-centered method is used by codes like OpenFoam and Ansys Fluent.

For the treatment of the method it is necessary to consider an owner cell and several neighbour cells surrounding it, in which the flow variables are stored in their centroids. In general, each cell will have M neighbours, which corresponds to the number of faces of each cell.

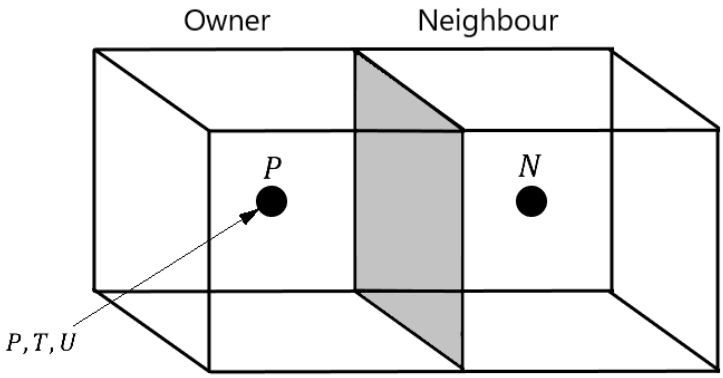


Figure 11 Representation of an Owner and a Neighbour polyhedral cells, Source: Own

The first step is to integrate the entire Navier-Stokes equation across the volume of a cell. Assuming steady conditions we have:

$$\int_V \left[\nabla \cdot (UU) + \frac{1}{\rho}\nabla p - \nabla \cdot (\mu \nabla U) - f_b\right] dV = 0 \tag{16}$$

Each term can be integrated separately as follows:

$$\underbrace{\int_V [\nabla \cdot (UU)] dV}_{\text{Convection Term}} = \underbrace{\int_V \left[-\frac{1}{\rho}\nabla p\right] dV}_{\text{Pressure Gradient}} + \underbrace{\int_V [\nabla \cdot (\nu \nabla U)] dV}_{\text{Diffusion Term}} + \underbrace{\int_V f_b dV}_{\text{Source}} \tag{17}$$

On equation (17) we have the convection term on the left-hand side and the pressure gradient, the diffusion term and the source term respectively on the right-hand side.

3.5.2 Source Terms Integration

Constant source terms are the most straight forward to integrate as they represent a constant value across the cell:

$$\int_V f_b dV = f_b V_P \tag{18}$$

Where V_P is the volume of the owner cell P. Hence, in the matrix form: $MU = B$, the gravity force is added to the right-hand side (to the B vector) as it does not depend on velocity.

On the other side, linear source terms require more treatment. They are unusual for the Navier-Stokes equations but they are more common in other scalar transport equations such as turbulent kinetic energy and temperature. Let's assume that the source term SU was added to the Navier-Stokes equations (where S is some scalar):

$$\frac{\partial U}{\partial t} + \nabla \cdot (UU) = -\frac{1}{\rho}\nabla p + \mu \nabla^2 U + f_b + SU \tag{19}$$

integrating the linear source over the cell volume yields:

$$\int_V [SU] dV = S_P \int_V U dV \tag{20}$$

The velocity variation across the cell is given by:

$$U = U_P + (x - x_P) \cdot (\nabla U_P) \tag{21}$$

Introducing equation (21) into equation (20), we have:

$$S_P \int_V (U_P + (x - x_P) \cdot (\nabla U_P)) dV \tag{22}$$

$$= S_P U_P \int_V dV + \underbrace{\left[\int_V (x - x_P) dV\right]}_{=0} \cdot (\nabla U_P) \tag{23}$$

$$= S_P U_P V_P \tag{24}$$

Where the underlined term equals to zero as it corresponds to the definition of the cell centroid, i.e. the average difference from every point on the cell to the centroid is zero. The result in equation (24) is proportional to the velocity. Hence, it can be added as $S_P V_P$ to the M matrix (implicit treatment, equation 25) or it can be added as $S_P U_P V_P$ to the B matrix (explicit treatment, equation 26).[20]

$$M = \begin{pmatrix} -S_1V_1 & 0 & 0 & \dots & 0 \\ 0 & -S_2V_2 & 0 & \dots & 0 \\ 0 & 0 & -S_3V_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -S_MV_M \end{pmatrix} \tag{25}$$

$$B = \begin{pmatrix} S_1U_1V_1 \\ S_2U_2V_2 \\ S_3U_3V_3 \\ \vdots \\ S_MU_MV_M \end{pmatrix} \tag{26}$$

The implicit treatment gets a diagonal contribution as the variables are stored in the cell centroid. The treatment of source terms for CFD codes will depend on the diagonal dominance; negative S values improve the diagonal dominance, then it is recommended an implicit treatment, whereas positive terms would decrease the diagonal dominance, in this case it is preferable an explicit treatment.[20]

3.5.3 Convection and Diffusion Terms Integration

The convection and diffusion terms (see equation 17) contain the divergence operator and for this reason require a special treatment to be integrated. This treatment is known as the Divergence Theorem (equation 27), it is a useful tool to convert volume integrals into surface integrals. It can be expressed as:[37]

$$\int_V (\nabla \cdot F) dV = \int_S (F \cdot \vec{n}) dS \tag{27}$$

Where F is a general vector and $\vec{n}$ is the unit normal vector.

Using the divergence theorem on the convection term leads to:

$$\int_V [\nabla \cdot UU] dV = \int_S [U(U \cdot \vec{n})] dS \tag{28}$$

The dot product of the velocity, the unit normal vector and the surface element corresponds to the volume flow rate out of the surface.

$$(U \cdot \vec{n})dS$$

The other velocity outside the bracket (U) is the unknown we are solving for.

The surface integral can now be split into a sum over a finite number of faces, i.e. compute the integral over each face of the cell and add them together:

$$\int_S [U(U \cdot \vec{n})]dS = \sum_{i=1}^{M} \int_S U_i(U_i \cdot \vec{n}_i)dS_i \tag{29}$$

As previously stated, the variation across the faces is linear (second order method), hence, an approximation can be done; the integral over each of these faces can be taken as the value at the face center U_f, equation (30), see (Jasak, 1996) for proof [36]. Once this approximation is made, the integration is gone. All that is left is a summation of terms, e.g. a summation of six terms for an hexahedra.[20]

$$\sum_{i=1}^{M} \int_S U_i(U_i \cdot \vec{n}_i)dS_i \approx \sum_{i=1}^{M} \int_S U_{fi}(U_{fi} \cdot \vec{n}_{fi})dS_i = \sum_{i=1}^{M} U_{fi}F_{fi} \tag{30}$$

Where F_{fi} represents the volume flux through the face.

However, we don't really have the value of these face centres (f), we only have the centroid values (P and N), as shown in Figure 12. Then, an interpolation is needed in order to compute the summation of the convection terms.

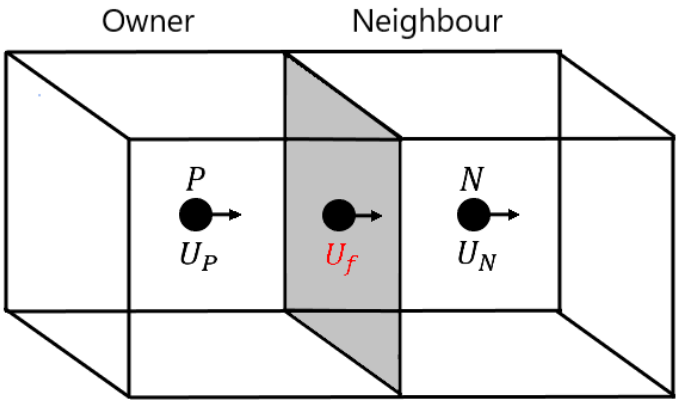


Figure 12 Representation of the face centre between the Owner and the Neighbour cells, Source: Own

There exist a variety of discretization schemes in CFD codes that allow the user to select the method of interpolating the flow variables on the cell faces, assuming that the values at the cell's centroids are known. The following list shows some of the interpolation schemes for calculating flow variables in (f), further explained in chapter (3.5.4):

(a) Upwind

(b) Linear (Central differencing or second order schemes)

(c) Linear Upwind

(d) QUICK (Advanced)

(e) Limited linear (Advanced)

Recalling equation (30), the contribution of the convection terms must be added to the M matrix from equation (15):

$$M = \begin{pmatrix} M_{11} & M_{12} & M_{13} & \dots & 0 \\ M_{21} & M_{22} & 0 & \dots & 0 \\ 0 & M_{23} & M_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & M_{mm} \end{pmatrix} \tag{31}$$

The convection term leads to diagonal and off-diagonal terms. The diagonal terms are given by the contribution of the owner cell centroid (P), and the off-diagonal terms are given by the contribution of the neighbour cells' centroids (N). The number of terms of this matrix is a reflection of the cell connectivity, e.g. a hexahedral cell will have a value on the diagonal and six off-diagonal terms. As it can be seen we can have multiple contributions to the M matrix and the B vector, hence, they can be added together to obtain an overall $MU = B$ matrix and it can be solved for the velocity field using an appropriate iterative solver.[20]

The treatment of the diffusion term is very similar to the convection term, as the divergence theorem is used as well.

3.5.4 Interpolation Schemes

For this chapter we are going to assume some quantity ϕ that can represent any field of the simulation T, p, U, etc., on the owner centroid (ϕ_P) and the neighbour cell centroid (ϕ_N). The objective is to calculate ϕ_f using the different interpolation schemes.[20]

- **Linear/Central Differencing.**

This first scheme assumes that the flow variables variation between the cell's centroids is linear, as well as in the Finite Volume Method. However, this scheme is often called unbounded, i.e. it is prone to oscillations.

It can be expressed as follows:

$$\phi_f = \lambda\phi_N + (1-\lambda)\phi_N \qquad \lambda = \frac{|x_f - x_p|}{|x_N - x_P|} \tag{32}$$

Where λ can get values between 0 and 1:

$$0 \leq \lambda \leq 1$$

Central Differencing is used for the diffusion term because it is really accurate. However, it should never be used for the convection term in a RANS (Reynolds Averaged Navier-Stokes) solver because it is unbounded and, as a result, it often leads to non-physical oscillations in the solution.[20]

- **Upwind Differencing**

The most popular type of schemes used for the convection term are based on Upwind Differencing. This method depends on the mass flux that is flowing through the cell, i.e. the mass flow rate through a face. As we consider the velocity through a face, which does not necessarily have to be aligned with the unit normal vector, we use the dot product of the velocity with the unit normal vector for that face, see equation (33):

$$F_f = \rho_f A_f (U_f \cdot \vec{n}) \tag{33}$$

Once the flow flux through the face is known we can decide the method to be used to interpolate onto the cell face. The value ϕ on the cell face (ϕ_f) is given by:

$$\phi_f = \begin{cases} \phi_P & F_f > 0 \quad \text{(Mass Flow Out)} \\ \phi_N & F_f < 0 \quad \text{(Mass Flow In)} \end{cases} \tag{34}$$

Contrary to the Central Differencing method, ϕ no longer varies linearly across the cell faces and it is considered constant between the cell centroid and the cell face. Hence, Upwind Differencing is only first-order accurate, and in consequence it loses a lot of resolution. However, it is the most stable scheme for convection dominated flows. For this reason, it can be used to generate an initial, stable solution.

- **Linear Upwind Differencing**

Linear Upwind Differencing is a very popular scheme used in almost every CFD code for the convection term. It is more accurate than the Upwind Differencing method. This scheme also depends on the value of the mass flux F. Thus, it uses information about the gradient at the cell centroid ($\nabla\phi$) to improve the accuracy of the projection on the cell face.

$$\phi_f = \begin{cases} \phi_P + (\nabla\phi)_P \cdot r & F_f > 0 \\ \phi_N + (\nabla\phi)_N \cdot r & F_f < 0 \end{cases} \tag{35}$$

The variation between the cell centroid and the face is linear. Hence, the scheme is nominally second order accurate. However, it may be slightly unstable. What some CFD codes begin with is using Upwind Differencing for the convection term, in order to get an initial solution, and then they introduce the Linear Upwind Differencing to increase the value on the cell ϕ_f and improve the accuracy of the solution.

However, we often have to limit the gradient $(\nabla\phi)_P$ to prevent local maxima/minima, i.e. to avoid solutions on the face greater than the values of the two surrounding centroids. In the final solution this can lead to oscillations on the solution domain due to the peaks on the cell faces. This is the reason why it is preferable to use the Upwind scheme first.

What can be done to improve this instability is to apply a gradient limiter, expressed in equation (36):

$$\phi_f = \phi_P + \varphi(\nabla\phi)_P \cdot r \qquad 0 \leq \varphi \leq 1 \tag{36}$$

- **Advanced Discretization Schemes**

The Advanced Discretization Schemes use a combination of Linear/Central Differencing for accuracy (ϕ_{CD}) and Upwind Differencing for stability (ϕ_{UD}), equation (37):

$$\phi_f = \psi\phi_{UD} + (1 - \psi)\phi_{CD} \tag{37}$$

Where ψ is a blending function to switch between schemes.

More information about discretization schemes and gradient limiters can be found in the ANSYS FLUENT User Manual (ch. 26) [35].

3.6 Grids

The initial phase of any numerical simulation begins with the generation of a suitable mesh. When generating meshes for the simulations, it is important to understand how discretization in a numerical method influences the accuracy of the results. Grid generation in CFD refers to a set of techniques for defining a numerical mesh throughout the system to be simulated. The grid used in CFD simulations will define the accuracy and resolution of the simulation results, as well as the calculation time and the amount of detail in the results.[15]

The process of mesh generation can be classified into two categories based on the topology of the elements that fill the domain; structured and unstructured meshes. The different types of grid generation methods have their advantages and disadvantages in terms of both solution accuracy and the complexity of the mesh generation process.[15]

A structured mesh is defined as a set of hexahedral elements with an implicit connectivity of the points in the mesh. Generating a structured mesh for a complex geometry can be a time-consuming task due to the possible need of breaking the domain manually into several blocks depending on the nature of the geometry. In a structured grid the node arrangement in the grid follows the same shape as the boundary surface, effectively scaling its interpolated nodes into the interior of the system. This type of grid generation provides high accuracy.[44]

The Cartesian Grid is the most basic method of grid generation. In this method, a rectangular grid is used to represent the entire system in order to be used in a CFD simulation, with grid faces aligned with the Cartesian axis system. Grids composed out of regular brick elements have the simplest structure since it is only necessary to define three one-dimensional arrays for the x, y and z values of the surfaces defining the element surfaces. Systems with curved and more complex surfaces and boundaries may require higher meshing density to ensure accuracy and stability, which consequently increases the computational time in the simulation.[15]

In order to deal with curved surfaces, the grid elements can be deformed to conform with specified geometric shapes. The resulting elements will then have general hexahedral shapes, and the grid is often referred to as a body-fitted grid and it generates distortions on the grid. The distortion of elements away from a simple rectangular shape has several consequences. It may affect numerical accuracy as well as make numerical approximations more complex. Because of these complications, an alternative grid approach is normally used in most of these systems (see Figure 13).

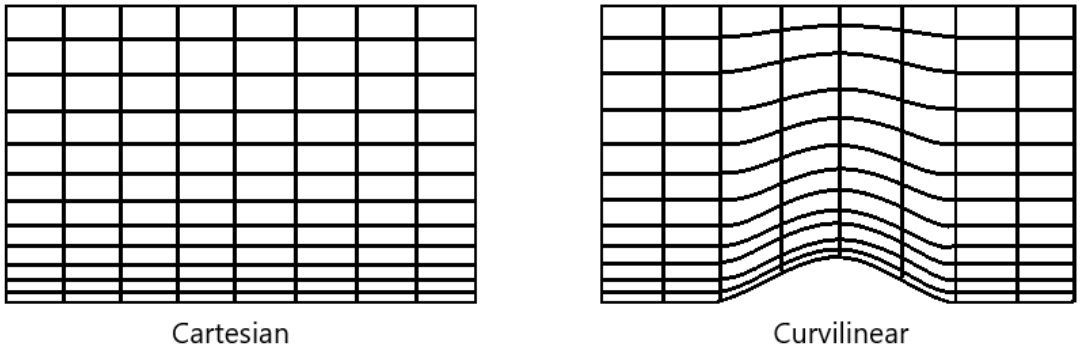


Figure 13 Structured Cartesian Grid examples, Source: Own

An unstructured mesh is defined as a set of elements, most typically triangles (for 2D problems) or tetrahedrons (for 3D problems). The unstructured mesh is a favorable choice due to its flexibility, automation and has no restrictions on the number of faces or edges on a cell, yet due to the existence of skewed elements in sensitive regions such as boundary layers, solution accuracy may be inferior to that of structured meshes. Therefore, the use of an unstructured grid does not always guarantee more efficient computation. The disadvantage of this type of grids is that because the data structure is irregular, it is more difficult to describe and store them.[44]

Another option is hybrid mesh generation, which aims to combine the benefits of both structured and unstructured meshes. In a hybrid mesh, the viscous region is filled with prismatic or hexahedral cells, while tetrahedral cells cover the rest of the domain. It employs an unique approach for dealing with complex shapes. For example, consider a square-shaped house with a round hole on the middle; hybrid grid generation would use a structured grid around the curved portion and a Cartesian grid along the orthogonal boundaries. The two grid styles would converge on each other and make a smooth transition somewhere between the two zones. The challenge in this grid generation method is defining the transition between different regions.[44]

Other grid generation techniques can be seen in reference [15].

3.7 Turbulent Methods

As discussed in previous chapters, the exact analytical solution of the Navier-Stokes equations (14) is still unknown as it has too much accuracy that even current super computers fail to solve it. However, actual concern relies on the averaged solution of the Navier-Stokes equations as it is enough for engineering problems. This chapter will introduce the Reynolds Averaged Navier-Stokes equations (RANS).

The idea behind the RANS equations is Reynolds decomposition, initially proposed by Osborne Reynolds, which consists in dividing an instantaneous variable, say u, by its time-averaged value, $\overline{u}$, and fluctuating quantities, u':[4]

$$u = \overline{u} + u' \tag{38}$$

Where an averaging operation is defined like:

$$\overline{u} = \frac{1}{\Delta t}\int_{t}^{t+\Delta t} u dt \tag{39}$$

The Navier-Stokes equations (14) can be split into an average and a fluctuating part. When averaging the fluid equations, a stress on the right hand side appears of the form $\rho\overline{u_i' u_j'}$. This is the Reynolds stress, conventionally written as R_{ij}:

$$R_{ij} = \rho\overline{u_i' u_j'} \tag{40}$$

The divergence of this stress is the force density on the fluid due to the turbulent fluctuations. The RANS equations can be written as:

$$\frac{\partial(\rho U)}{\partial t} + \nabla\cdot(\rho UU) = -\nabla p + \mu\nabla^2 U + \rho f_b - \underbrace{\nabla\cdot(\overline{\rho U'U'})}_{\textbf{Reynolds-stress}} \tag{41}$$

The Reynolds stress is the product of the two fluctuating velocity components averaged and then taken the divergence of that term. It is an unknown in our RANS equations. In order to solve the closure problem and solve for the mean velocity U, a Reynolds stress model is needed, i.e. express the Reynolds stress in terms of mean quantities so that equation (41) can be solved.

There exist various models available in order to represent Reynolds-stress. The most popular way is with the Boussinesq hypothesis which relates the Reynolds stress to the mean velocity gradients in the flow:

$$-\underbrace{\nabla\cdot(\overline{\rho U'U'})}_{\textbf{Reynolds-stress}} = \mu_t\left(\nabla U + (\nabla U)^T\right) - \frac{2}{3}\rho k\mathbf{I} - \frac{2}{3}(\nabla\cdot U)\mathbf{I} \tag{42}$$

The Reynolds stress is related to the mean velocity gradients by the dynamic turbulent

viscosity or Eddy viscosity (μ_t). Once it is known, the Reynolds stress term can be calculated with equation (42) and finally close the system of equations. Turbulent methods are used to compute the eddy viscosity, some of them will be introduced in the following chapters.

3.7.1 The k-epsilon Model

The k-ε turbulent model[37] was first proposed in 1973 and it is the most common model used in computational fluid dynamics (CFD) to simulate mean flow characteristics for turbulent flow conditions. Its name refers to two turbulent flow variables:

- The first transported variable is the turbulent kinetic energy (k).
- The second transported variable is the rate of dissipation of turbulent kinetic energy (ε).

Epsilon (ε) is the turbulence dissipation rate $[m^2/s^3]$, the rate at which the turbulent kinetic energy (k) is converted into thermal energy by the action of viscosity.

Before the k-epsilon model existed, a mixing length (l_m) approach was used to calculate the eddy viscosity (μ_t).

$$\mu_t = \rho k^{1/2} l_m \qquad \text{or} \qquad \mu_t = \rho l_m^2 \left| \frac{\partial U}{\partial y} \right| \tag{43}$$

The mixing length physically represents an indicative size of the turbulent eddies that exist in the flow. Any turbulent flow has a full spectrum of sizes and energies of these eddies. In areas where the flow is very energetic the local mixing length is going to be really large, and according to equation (43), because the mixing length is large the eddy viscosity will be so too.

One of the earliest approaches is the Prandtl mixing length hypothesis, which can be seen in equation (44):

$$l_m = ky \qquad k = 0.41 \tag{44}$$

This hypothesis states that when the flow is some distance y from a wall, the maximum size of the eddies is limited by the size of the wall, i.e. the presence of the wall physically blocks the size of the eddies. This effect is also produced by the viscosity in the viscous sub-layer very close to the wall.

An improvement on the Prandtl model is the Van Driest mixing model, which takes into account the turbulence damping besides specifying that as the flow approaches the wall the

action of viscosity is also going to block the size of the eddies, i.e. the viscous sub-layer reduces the effective size of the eddies in the mixing length hypothesis.[8] It takes the form given in equation (45):

$$l_m = ky\left[1 - exp\left(-\frac{y^+}{A^+}\right)\right] \qquad A^+ = 26.0 \tag{45}$$

Where y^+ is the distance to the wall or wall normal coordinate.

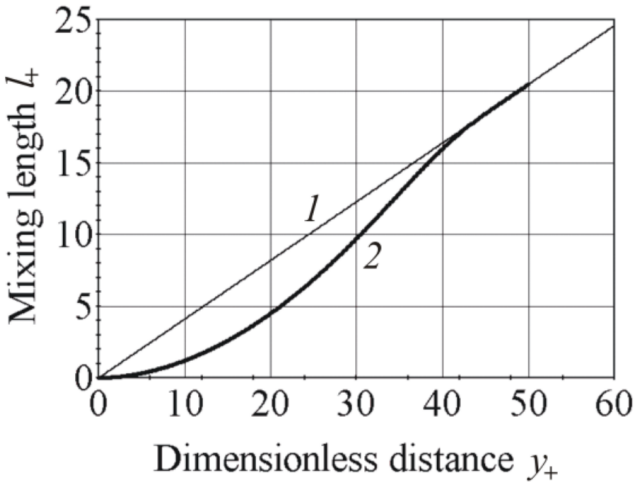


Figure 14 Functional dependence of the dimensionless mixing length on wall-normal coordinate, Source: V. A. Kuznetsov, et.al. (2019)[8]

As it can be seen in figure 14, curve number 2, which corresponds to the exponential function (45), shows that when the viscosity in the viscous sublayer is taken into account, as the flow gets closer to the wall, the mixing length gets smaller than the Prandtl model described in curve number 1.

In this early mixing models, the mixing length is specified algebraically, i.e. the mixing length only depends on the distance to the nearest wall, as shown in the previous equations. Once the geometry is specified, the distance to the nearest wall can be determined throughout the entire domain, therefore the mixing length is fixed and so is the eddy viscosity.

Some improvement can be made on this model, because turbulence is convective and defuses through the flow, rather than being static and fixed to some distance from the wall. For this reason, a transport equation is introduced, which needs to be solved for the turbulent quantities. Both the turbulent kinetic energy (k) and the turbulence dissipation rate (ε) can be determined by solving a transport equation. When this is done, equation (46) can be used to compute the eddy viscosity and solve the system of RANS equations.[20]

$$\mu_t = C_\mu \frac{\rho k^2}{\varepsilon} \tag{46}$$

When equations (43) and (46) are combined, we can calculate the mixing length from the turbulence dissipation rate:

$$l_m = \frac{C_\mu k^{3/2}}{\varepsilon} \tag{47}$$

When the transport equations are being solved in the k-epsilon model, the transport equation for k represents the turbulent kinetic energy in the flow while the equation for epsilon (ε) is in reality another way of solving a transport equation for the mixing length, or the physical size of the eddies. For this reason, the transport equation for epsilon may sometimes be referred to as the dissipation rate equation or as the scale determining equation in the k-epsilon model.[20]

The transport equation for the turbulent kinetic energy k is given in equation (48):

$$\underbrace{\frac{\partial(\rho k)}{\partial t}}_{\text{Time}} + \underbrace{\nabla \cdot (\rho U k)}_{\text{Convection}} = \underbrace{\nabla \cdot \left[\left(\mu + \frac{\mu_t}{\sigma_k}\right) \nabla k\right]}_{\text{Diffusion}} + \underbrace{P_k + P_b - \rho\varepsilon + S_k}_{\text{Sources + Sinks}} \tag{48}$$

Following the typical transport equation scheme, on the left-hand side it has a time derivative followed by a convection of turbulent kinetic energy while on the right-hand side the diffusion of turbulent kinetic energy plus the sources and sinks of turbulent kinetic energy can be found. The negative term ($\rho\varepsilon$) accounts for the dissipation rate, which shows that the equation for epsilon is acting to dissipate turbulent kinetic energy in the flow.

The transport equation for the rate dissipation of turbulent kinetic energy (ε) is very similar, it is given in equation (49):

$$\underbrace{\frac{\partial(\rho\varepsilon)}{\partial t}}_{\text{Time}} + \underbrace{\nabla \cdot (\rho U \varepsilon)}_{\text{Convection}} = \underbrace{\nabla \cdot \left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right) \nabla \varepsilon\right]}_{\text{Diffusion}} + \underbrace{C_1 \frac{\varepsilon}{k}(P_k + C_3 P_b) - C_2 \rho \frac{\varepsilon^2}{k} + S_\varepsilon}_{\text{Sources + Sinks}} \tag{49}$$

Again, there is a typical time derivative and a convection term on the left-hand side and a diffusion term on the right-hand side followed by some sources and sink terms which include some empirical model coefficients (C_1, C_2 and C_3). These may vary depending on the variant of the chosen k-epsilon model. Once the previous transport equations are solved, equation (46) can be used to calculate the eddy viscosity.

The model coefficients for the standard k-ε model have evolved through time. They have

been calculated by numerous iterations of data fitting for a wide range of turbulent flows. These are as follows: [45]

Model	σ_k	σ_ε	C_1	C_2	C_μ
Launder & Sharma (1974)	1.0	1.3	1.44	1.92	0.09

Table 2 Launder & Sharma coefficients for the standard k-ε model [45]

The Launder & Sharma (1974)[45] coefficients are the most up-to-date. They are used in solvers such as Fluent, OpenFoam, CFX and Star.

Recalling the Prantdl mixing length model from equation (45), the mixing length was damped close to the wall in order to account for viscosity and the viscous sub-layer. However, as previously stated, the k-ε model is not solving for the mixing length but for the dissipation rate ε. For this reason, it is necessary to use a strategy to damp the dissipation rate close to the wall. This is accomplished by incorporating specific damping functions into the model coefficients, which will help to solve the equations even when the closest to the wall cell is in the viscous sub-layer ($y^+ < 5$). These damping functions are termed f_1, f_2 and f_μ. This formulation of the k-ε model is called the low-Reynolds number formulation.[20]

The original damping functions for the standard k-epsilon model are introduced down below:[45]

$$f_1 = 1 \tag{50}$$

$$f_2 = 1 - 0.3exp(-Re_T^2) \tag{51}$$

$$f_\mu = exp\left(\frac{-3.4}{(1 + (Re_T/50))^2}\right) \tag{52}$$

The key parameter in these damping functions is the turbulent Reynolds number (Re_T). The Reynolds number is usually described as a ratio of some inertial forces to some kind of viscous forces, taking the usual form of equation (4). However, there is a Reynolds number that characterizes the strength of the velocity fluctuations and the turbulence near the wall known as the turbulent Reynolds number:

$$Re_T = \frac{\rho k^{1/2} l_m}{\mu} \tag{53}$$

This definition can be adapted to the k-ε model, knowing that it is not solving for the mixing length but for the dissipation rate (ε). For this reason, the mixing length can be introduced in equation (53) which will lead to equation (54):

$$Re_T = \frac{\rho k^2}{\mu \varepsilon} \tag{54}$$

When Re_T is small, viscous effects dominate, and the flow is located in the viscous sublayer (low-Re formulation).

The f_μ damping function is directly applied to the eddy viscosity as:

$$\mu_t = f_\mu C_\mu \frac{\rho k^2}{\varepsilon} \tag{55}$$

Looking at equation (52), far away from the wall, where the turbulent Reynolds number tends to larger numbers, f_μ tends to 1. This damping function is applied to every cell in the mesh, locations where $f_\mu = 1$ will be using a high-Reynolds formulation of the k-ε model, whereas locations where $f_\mu < 1$ will be using low-Reynolds formulation of the damping function.[20]

Recalling equation (49), f_1, equation (50), is applied to the term C_1 which appears in the production term for the turbulence dissipation rate. Whereas f_2, equation (51), is the damping function applied to C_2 which appears in the dissipation term for the dissipation of turbulent kinetic energy, i.e. f_2 acts to reduce the dissipation of epsilon near the wall. This means that we are going to get more dissipation of turbulent kinetic energy near the wall.[20]

It's important to remember that while the k-ε model can resolve these flows in its low-Reynolds formulation, the k-ω SST model is widely regarded as providing better performance for these low-Reynolds applications. As a result, the k-ε model is typically preferred for high-Reynolds situations where $y^+ > 30$ and none of these damping functions are involved.

3.7.2 The k-omega Model

The k-ε model[37] is not accurate at predicting boundary layers with adverse pressure gradients. The prediction gets even worse when shock waves are present (supersonic flow). For this reason, a better model is required for aerodynamics and turbomachinery, as these scenarios present many applications where it's very important to compute the flow field in the presence of an adverse pressure gradient, e.g. airfoils at high angles of attack.[20]

Once the performance limitations of the k-epsilon model were first realized, a number of different k-ω models were proposed in order to address these limitations. The version by

Wilcow is the one where most of the modern versions are built. It is important to be aware that there exist different versions of the same model with slightly different coefficients. The NASA turbulence modelling website[17] gives details of the different versions.

Once again, the k-ω model objective is to compute the eddy viscosity term (μ_t), and to solve Reynolds Averaged Navier-Stokes equations (41).

Before explaining how this model works, it is necessary to introduce what physical property does omega (ω) represent. Omega is the specific turbulence dissipation rate $[1/s]$. It can be expressed as a function of ε:

$$\omega = \frac{\varepsilon}{C_\mu k} \qquad\qquad C_\mu = 0.09 \tag{56}$$

Omega (ω) and epsilon (ε) both represent the dissipation of turbulence in the flow field. In addition, a transport equation can be solved for either omega or epsilon as they are related by equation (56).

In consequence, this model will be solving a transport equation for omega instead of epsilon, equation (57). The difference between the two models relies on the fact that the k-ω model has different values for the empirical constants. The value of these constants ($\alpha, \beta, \beta^*, \sigma_k, \sigma_\omega$) can be checked in either a user guide or the NASA website.[17]

$$\underbrace{\frac{\partial(\rho\omega)}{\partial t}}_{\text{Time}} + \underbrace{\nabla\cdot(\rho U\omega)}_{\text{Convection}} = \underbrace{\nabla\cdot\left[\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\nabla\omega\right]}_{\text{Diffusion}} + \underbrace{\frac{\gamma}{\mu_t}P_k - \beta\rho\omega^2}_{\text{Sources + Sinks}} \tag{57}$$

As explained above, the k-ε model uses empirical damping functions in the viscous sublayer ($y^+ < 5$). However, these damping become functions inaccurate in the presence of pressure gradients. The k-ω model does not need these damping functions and it gives better accuracy for aerodynamics and turbomachinery.

The main weakness of the k-ω model is that it is dependent on the freestream turbulence conditions, i.e. small changes in k_∞ lead to large changes in the turbulent viscosity μ_t, Johan C. Kok (2000)[39].

As the k-ε is not as susceptible to the freestream values of k, it can be used away from the wall, while the k-ω model can be used close to the wall. Between the two regions, a hybrid of the two models can be employed. This is the basis of the k-ω SST model proposed in 1992 which will not be described in this report.[20]

3.8 Introduction to OpenFOAM

OpenFOAM (Open-source Field Operation And Manipulation) is a C++ toolbox for creating custom numerical solvers and pre/post-processing utilities for solving continuum mechanics problems, particularly computational fluid dynamics (CFD). Since 2004, it has been a free, open source CFD software developed primarily by OpenCFD Ltd. It has a broad user base from both commercial and academic organizations around the world in almost every field of engineering and science.

The following chapters will be dedicated to understand how this software works and how to setup the desired case for this project.

OpenFOAM is supplied with pre- and post-processing environments. The interface to the pre- and post-processing are themselves OpenFOAM utilities, which enables the user to have a consistent handling of data across all the environments. The overall structure of OpenFOAM is shown in Figure 15.[12]

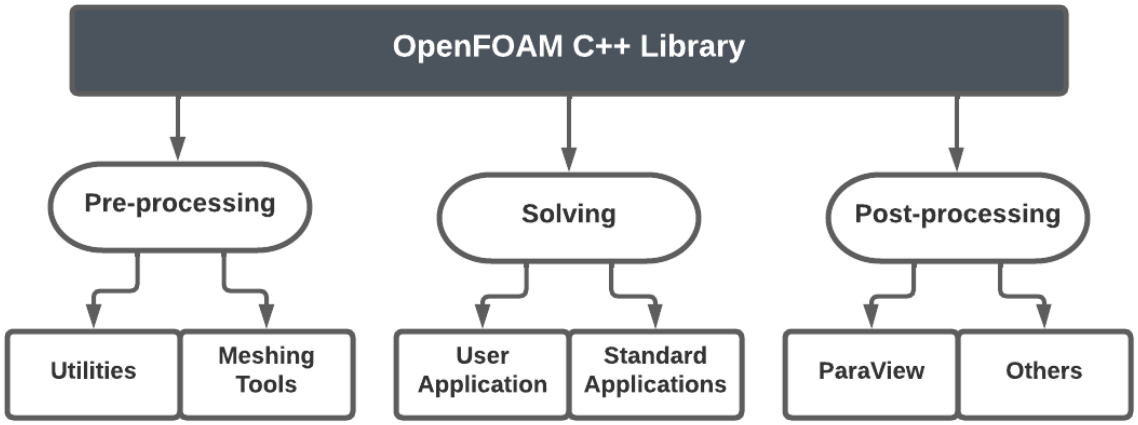


Figure 15 Overview of OpenFOAM structure, Source: Own

Regarding the structure and organization of OpenFOAM software, initially the user would assign a name to the case. This name now becomes the name of a directory in which all the case files and subdirectories are stored. Cases are usually setup in OpenFOAM by editing case files. Editing files is possible in OpenFOAM because the I/O uses a dictionary format with keywords that can be easily understood by the users. A case being simulated involves data for mesh, fields, properties, control parameters, etc. As mentioned above, in OpenFOAM this data is stored in a set of files within a case directory (Figure 16) rather than in a single case file, as in many other CFD packages.

Different tutorials may be a good starting point for users with low experience. These try to show the use of the different solvers. They are located in the *$FOAM_TUTORIALS*

directory which makes it easier to reach them in case of necessity.

The basic directory structure for a OpenFOAM case, that contains the minimum set of files required to run an application, is shown in Figure 16:[12]

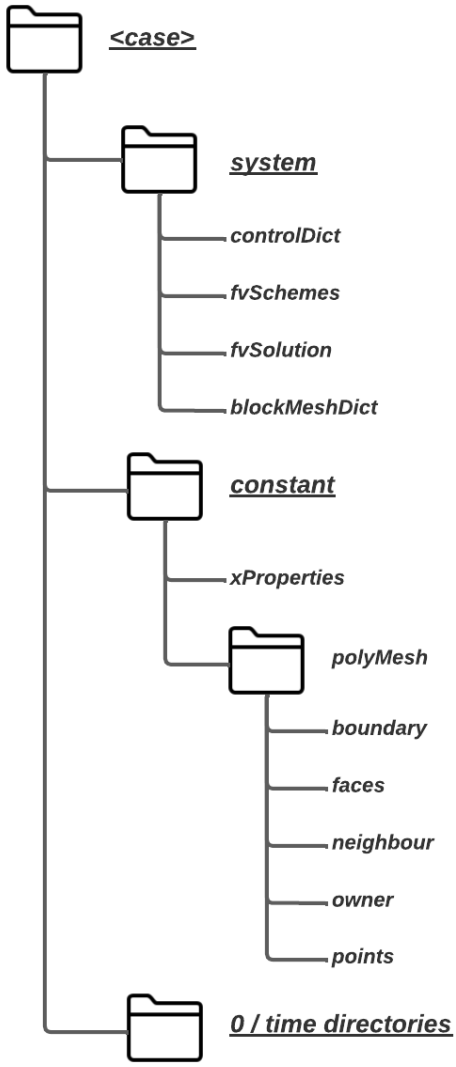


Figure 16 Case directory structure, Source: Own

A case file in OpenFOAM is divided in:

- The **system** directory: This directory is used for configuring parameters related to the solution procedure. It must include at least three of the following files: *controlDict*, which specifies the start/end time, time step, and data output parameters for the run; *fvSchemes*, which specifies the discretisation schemes used in the solution when the solver is running;

and *fvSolution*, which specifies the equation solvers, tolerances, and other algorithm controls for the run.

- The **constant** directory: This directory contains a detailed description of the case mesh in the polyMesh subdirectory and files providing physical attributes for the application in question, such as *transportProperties*.
- The **time** directories: This directories include data files for specific fields. The data can be either the user's beginning values and boundary conditions for defining and initialising the problem variables, or OpenFOAM's results saved to a file. Each time directory's name is determined by the simulated time at which the data is written. The initial conditions are usually saved in a directory named '0'.

The most typical way of specifying data in OpenFOAM is through dictionaries. Dictionaries provide the means for organising entries into logical categories. All OpenFOAM data files begin with a dictionary entitled FoamFile, which has a standard set of keyword entries, shown in Table 3:

Keyword	Description	Entry
`version`	I/O format version	`2.0`
`format`	Data format	`ascii/binary`
`location`	Path to the file, in "..."	(optional)
`class`	OpenFOAM class constructed from the data file	`dictionary` or a field
`object`	Filename	e.g. `controlDict`

Table 3 Header keywords entries for data files.[12]

OpenFOAM is able to attach dimensions to field data and physical properties in order to make meaningful algebraic operations. The I/O format for a *dimensionSet* is an array of 7 scalars delimited by square brackets, e.g. `[0 2 -1 0 0 0 0]`, where each of the values corresponds to the power of each of the base units of measurement listed in Table 4:

Nº.	Property	SI unit
1	Mass	kilogram (kg)
2	Length	metre (m)
3	Time	second (s)
4	Temperature	Kelvin (K)
5	Quantity	kilogram-mole (kg/mol)
6	Current	Ampere (A)
7	Luminous Intensity	candela (cd)

Table 4 *dimensionSet* format with SI units.[12]

Physical properties are typically specified with their associated dimensions. These entries have the format of the following example of a *dimensionedScalar*:

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

The first nu is the keyword; the next entry is the *dimensionSet* and the final entry is the scalar value.

Much of the I/O data in OpenFOAM are tensor fields, e.g. velocity (v) or pressure (p) data, that are read from and written into the time directories. The way OpenFOAM writes field data using keyword entries is described in Table 5:

Keyword	Description	Example
`dimensions`	Dimensions of field	`[1 1 -2 0 0 0 0]`
`internalField`	Value of internal field	`uniform (1 0 0)`
`boundaryField`	Boundary field	see List below

Table 5 Main keywords used in field dictionaries.[12]

The *boundaryField* keyword defines boundary conditions. A boundary is generally broken up into a set of patches for the purpose of applying boundary conditions. One patch may include one or more enclosed areas of the boundary surface which do not necessarily need to be physically connected. Every patch has a type assigned to it as part of the mesh description. This is done by including a `type` entry that specifies the type of boundary condition and a `value` to specify the value of the variable at the boundary condition. In time directories, boundary conditions are provided in field files, e.g. p, U. The main basic boundary condition types available in OpenFOAM are summarised below. This is not a complete list; for all types see reference [31].

- `fixedValue`: The value of the field is specified by a value.
- `fixedGradient`: the normal gradient of the field ($\frac{\partial}{\partial n}$) is specified by `gradient`.
- `zeroGradient`: The normal gradient of the field is zero.
- `calculated`: The patch field value is calculated from other patch fields.
- `mixed`: The patch field value has a mixed condition `fixedValue/fixedGradient`.

OpenFOAM always operates in a three-dimensional Cartesian coordinate system and all geometries are generated in 3D. Hence, OpenFOAM solves the case in 3 dimensions by default, however, it can be instructed to solve in 2 dimensions by specifying an `empty` boundary condition on boundaries normal to the (3rd) dimension for which no solution is required.

The mesh generator supplied with OpenFOAM, *blockMesh*, generates meshes from a de-

scription specified in an input dictionary, *blockMeshDict*, located in the `system` directory for a given case.

The scheme of the *blockMeshDict* dictionary is as follows; the dictionary first specifies coordinates of the block vertices; it then defines the blocks from the vertex labels and the number of cells within it (multiple blocks can be generated); and finally, it defines the boundary patches names.

The mesh is generated by running *blockMesh* on this *blockMeshDict* file. This is done simply by typing in the terminal:

```
blockMesh
```

4 Doublet Injector Simulation

This section will be dedicated to simulate a doublet injector with the OpenFOAM software recently introduced. The objective of this simulation will be to observe the two-dimensional (2D) phenomenon of atomization of two impinging jets of water (Figure 17) with different impinging angles and derive a correlation between mixture quality and impinging angle.

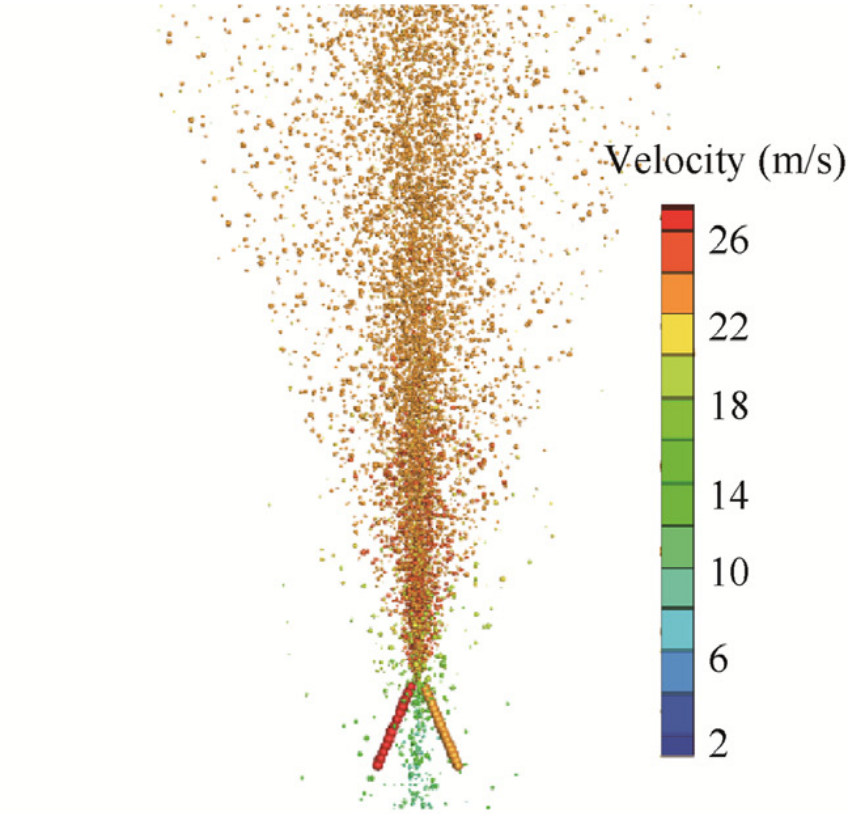


Figure 17 Doublet impinging spray simulation, Source: Qiang Wei, et.al. (2017)[42]

The disintegration of a continuous phase that gets dispersed in liquid droplets is called atomization, and the resulting droplets of the system are named spray.

In order to recreate this scenario it is necessary to select the type of simulation that is going to be used and define a convenient mesh depending on the accuracy that wants to be achieved. In this work it is intended to adopt a procedure in which the continuous phase (air) is treated Eulerian (chapter 4.1) and the discrete phase (water droplets) is treated Lagrangian (chapter 4.2), a Lagrangian-Eulerian multiphase approach.

Hence, it will be a steady, incompressible, turbulent, Lagrangian-Eulerian multiphase simulation.

4.1 Eulerian Multiphase Modeling

An Eulerian multiphase modelling approach can account for both dispersed-continuous phase interactions and continuous-continuous phase interactions. A dispersed phase can either be solid particles, liquid droplets or gas bubbles and they are dissolved in the continuous fluid. Their size can vary from μm up to mm in diameter. Following, the full Eulerian multiphase model will be introduced which accounts for dispersed-continuous and for continuous-continuous phase interactions. There exist two simplified versions of the full Eulerian model[20]:

- **Mixture Models:** simplified version of the full Eulerian model for *dispersed-continuous* phase interactions.
- **Volume-of-Fluids:** simplified version of the full Eulerian model for *continuous-continuous* phase interactions.

The Eulerian model solves a continuity equation for each phase in the system[20]. For instance, in an air-water system it will be solving a continuity equation for both the air and the water phases. The conservation of mass equation for a generic phase is given by:

$$\underbrace{\frac{\partial(\alpha_q \rho_q)}{\partial t}}_{\text{Time}} + \underbrace{\nabla \cdot (\alpha_q \rho_q U_q)}_{\text{Convection}} = \underbrace{\sum_{p=1}^{N} (\dot{m}_{pq} - \dot{m}_{qp})}_{\text{Sources+Sinks}} \tag{58}$$

The form of equation (58) is similar to the one presented for single phase flows in chapter (3.4.1) but with some small differences. The first difference is that all the terms on the left-hand side are multiplied by the volume fraction of the phase q (α_q). The reason for this is because physically the phase only exists where the volume fraction is greater than 0 ($\alpha_q > 0$). The additional term on the right-hand side of the equation represents the mass transfer between phases in the system. This mass transfer term is typically used for phase change processes such as evaporation and condensation. The subscript pq indicates mass transfer from p to q. This way, $\dot{m}_{pq}$ acts like a source term, whereas $\dot{m}_{qp}$ acts like a sink term.[20]

The main difference between the two types of phase interaction is that in a dispersed-continuous phase interaction the volume fraction should be able to take any value between 0 and 1, meaning that the concentration of the dispersed phase can vary in the continuous phase cells. On the other side, continuous-continuous phase interactions are restricted to a volume fraction of either 0 or 1, except in the interface region, where all the gradients are concentrated.

A volume fraction equation is going to be solved by the CFD solver for each additional phase in the multiphase model, e.g. a two phase system (air and water) will be solving one volume fraction equation. This is because the additional phase can be easily deduced by the volume fraction of the primary phase, as the sum of volume-fractions must equal 1:

$$\sum_{q=1}^{N} \alpha_q = 1 \tag{59}$$

For dispersed continuous phase (mixture model) interactions, e.g. water droplets dispersed in air, the upwind differencing discretization scheme presented in chapter (3.5.4) can be used to calculate the convection term in equation (58).

In addition, the Eulerian multiphase model solves a momentum equation for each phase, e.g. one momentum equation for the air velocity and another one for the water velocity. The conservation of momentum of a generic phase is given by:

$$\underbrace{\frac{\partial(\alpha_q \rho_q U_q)}{\partial t}}_{\text{Time}} + \underbrace{\nabla \cdot (\alpha_q \rho_q U_q U_q)}_{\text{Convection}} = \underbrace{-\alpha_q \nabla p}_{\text{Pressure Gradient}} + \underbrace{\nabla \cdot \alpha_q}_{\text{Difussion}} + \underbrace{\sum_{p=1}^{N} (D_{pq} + \dot{m}_{pq} U_{pq} - \dot{m}_{qp} U_{qp})}_{\text{Sources+Sinks}} \tag{60}$$

The pressure p, on the right-hand side does not present any subscript. This is because the pressure field is shared between all the faces in our system and it allows to couple the phases together in the solution field. The other way of coupling the equations together is by adding the sources and sinks term on the right-hand side of equation (60). This term represents the momentum transfer between phases (e.g. between water and air). Additional forces such as lift and drag may provide additional momentum transfer between phases (D_{pq}).

In terms of the solution field, because each phase has its own momentum equation, each phase will have its own velocity field stored at the cell centroid (U_p and U_q). For instance, in a water-air system, the water and air velocities at the cell centroid of each cell will be different. These two velocities are different due to drag and other inter-phase momentum transfer forces. The predominant momentum transfer mechanism is inter-phase drag (D_{pq}).

The general form of the drag force is given by the following expression:

$$\frac{1}{2} \rho U^2 A C_D \tag{61}$$

Knowing that the primary phase moves at a velocity U_q and that the secondary phase moves at a velocity U_p, the drag force acting between both phases can be written in the form given by equation (62):

$$D_{pq} = \frac{1}{2} \rho_q C_D A_p (U_p - U_q) |U_p - U_q| \tag{62}$$

As it is a CFD code, the force has to be per unit volume, which is given in equation (63):

$$D_{pq} = \frac{1}{2}\rho_q C_D \left(\frac{A_p}{V}\right)(U_p - U_q)|U_p - U_q| \tag{63}$$

The drag coefficient C_D is unknown as it is so the interfacial area ($\frac{A_p}{V}$), i.e. the contact area between the two phases per unit volume. These two terms can be modelled in the CFD model by the user and are going to govern the drag obtained between the two phases.

First, the interfacial area per unit cell volume for dispersed spheres ($\frac{A_p}{V}$) can be written as:

$$\frac{A_p}{V} = \frac{\text{Volume of Spheres}}{\text{Volume of cell}} = \frac{\text{Surface Area of Spheres}}{\text{Volume of Spheres}} \tag{64}$$

The first fraction represents the volume fraction of the spheres inside a cell whereas the second fraction is the interfacial area of the spheres. This equation can be developed to obtain:

$$\frac{A_p}{V} = \alpha_p \left(\frac{\pi d_p^2}{\pi d_p^3/6}\right) = \frac{6\alpha_p}{d_p} \tag{65}$$

In order to calculate the C_D coefficient[9], it is assumed that the dispersed phase is formed of small spheres. The drag on these spheres moving through a viscous fluid can be calculated using several models that are able to capture this drag coefficient variation empirically. The Reynolds number for the particles (e.g. water droplets) is going to be calculated with the difference in velocity between the different phases as seen in equation (66):

$$R_e = \frac{\rho_q |U_p - U_q| d_p}{\mu_q} \tag{66}$$

The simplest and most popular drag model is the *Schiller-Naumann* drag model:[9]

$$C_D = \begin{cases} \frac{24}{R_e}(1 + 0.15R_e^{0.687}) & R_e < 1000 \\ 0.44 & R_e > 1000 \end{cases} \tag{67}$$

Another model is the *Wen Yu* drag model, suitable for densely distributed solid particles and useful for our case simulation. It can be seen in reference Milad Mottaghi [9].

4.2 Lagrangian Particle Tracking

Because some fluid systems transport solids, a Lagrangian approach[20] is frequently required. Sand on a beach, combustion exhaust from automobiles transporting soot and other solids, or, as in our case, nozzle sprays that carry fluid atomized droplets are examples of this type of fluid system. As an introduction, this chapter will explain the transport of particles in a fluid flow.

The Lagrangian Particle Tracking method stems from the fact that conventional solvers cannot account for individual particles moving through a fluid, as their streamlines may differ from the fluid because of gravity and/or drag forces.

If we were to use a streamline to track a particle trajectory, the following differential equation would need to be solved to update the particle position x_p:

$$\frac{dx_p}{dt} = U_p \tag{68}$$

Where U_p is the particle velocity, which does not necessarily need to be the same as the local fluid velocity. When the particle has no mass, or very little mass, it will move with the fluid. In this case, the particle velocity U_p is equal to the local fluid velocity U in the cell of the CFD mesh.

In order to solve equation (68) for the streamlines, an Euler explicit time stepping approach can be used:

$$\frac{x_p^{i+1} - x_p^i}{\Delta t} = U_p^i \tag{69}$$

Where the i sub-index corresponds to the current particle position. In order to calculate the next particle position, the previous equation can be rearranged:

$$x_p^{i+1} = x_p^i + U_p^i \cdot \Delta t \tag{70}$$

Because a streamline approach is being used at the moment, the particle velocity U_p is equal to the local fluid velocity. Hence, the entire motion of a single particle can be calculated using equation (70) repeatedly for each time step using the local fluid velocity at each particle position.

However, in our case of study, the particles' mass is not negligible, hence, their velocity

is not equal to the fluid velocity. In this scenario it is necessary to calculate a force balance to calculate U_p.

According with Newton's second law, the force applied to a solid particle with mass can be calculated as:

$$F = m_p a \qquad F = m_p \frac{dU_p}{dt} \tag{71}$$

In a dispersed fluid there are multiple different contributions to the total force on a particle. These are:[20]

$$m_p \frac{dU_p}{dt} = F_{\text{Drag}} + F_{\text{Buoyancy}} + F_{\text{Others}} \tag{72}$$

The drag and buoyancy forces are main forces acting on the particle, and so, the ones described below.

Figure 18 represents the force balance in a single particle moving at a certain generic angle.

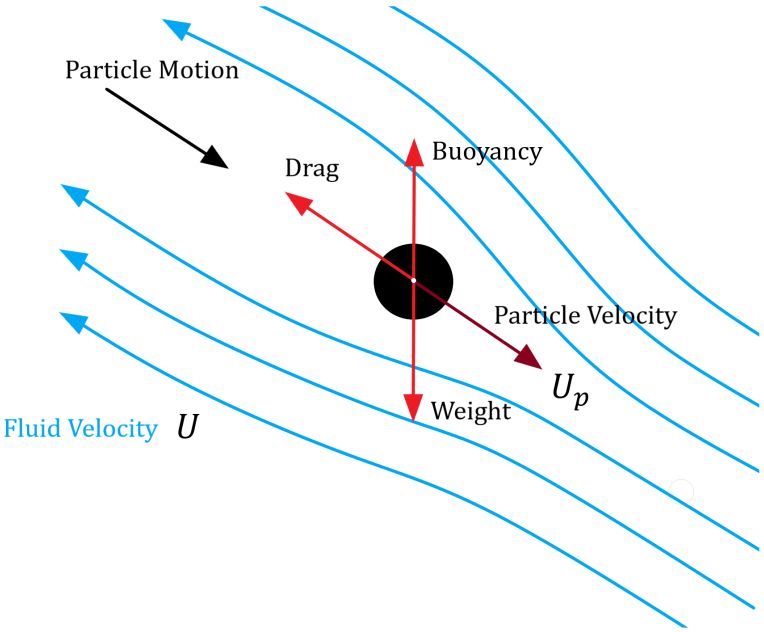


Figure 18 Forces balance of a sphere particle moving inside a fluid, Source: Own

As it is seen, the drag force acts to the opposite direction of the particle's motion (U_p), whereas buoyancy and weight always act vertical in the direction of gravity. However, if the sphere is moving against a non-stationary fluid, the drag force will act in the direction of $U - U_p$.

The weight of the particle acts in the direction of g, it can be written as:

$$F_{\text{Weight}} = \rho_p g V_p \tag{73}$$

The buoyancy force can be seen as the weight of the displaced fluid and it acts in the direction of $-g$. It can be written as:

$$F_{\text{Buoyancy}} = -\rho_f g V_p \tag{74}$$

Thus, the net force in the direction of g can be expressed as:

$$F_{\text{Vertical}} = (\rho_p - \rho_f) g V_p \tag{75}$$

The difference between both densities will determine the motion of the particle in the direction of the g axis.

- The particle will go downwards if $\rho_p - \rho_f > 0$
- The particle will go upwards if $\rho_p - \rho_f < 0$

On the other hand, the drag force acts in the direction of $U - U_P$. The general form of drag can be seen in equation (61). The specific form of the drag force for Lagrangian particle tracks is shown in equation (76):

$$F_{\text{Drag}} = \frac{1}{2} \rho_f C_D A_p (U - U_p) |U - U_p| \tag{76}$$

Where A_p is the projected area in the direction of the flow. In this case, both the projected area A_p and the drag coefficient C_D need to be calculated.

The projected area of a sphere can be accounted by the following expression, which corresponds to the area of a circle:

$$A_p = \frac{\pi d_p^2}{4} \tag{77}$$

The drag coefficient can be calculated using the procedure explained in chapter 4.1. However, the particle velocity is unknown and it cannot be used to calculate the particle Reynolds

number with the following expression:

$$R_e = \frac{\rho d_p |U_p - U|}{\mu} \tag{78}$$

In CFD codes this can be solved by evaluating U_p from the previous time step of the Lagrangian track. Hence, an iterative scheme is needed to calculate the particle settling velocity.

Following, an overview of the sequence of calculations used to determine a Lagrangian particle track in CFD codes is presented:

(a) Calculate R_e using U_p from the previous time step using equation (78).

(b) Calculate the drag coefficient C_D from the corresponding drag model (*Schiller Naumann, Wen Yu*, etc.).

(c) Solve the force balance to calculate the new particle velocity U_p using equation (79).

(d) Update the particle position using equation (70).

$$m_p \frac{dU_p}{dt} = \frac{1}{2}\rho_f C_D A_p (U - U_p)|U - U_p| + (\rho_p - \rho_f) g V_p \tag{79}$$

This procedure needs to be repeated for each time step and for each particle as it moves along its track.

A simplification of the force balance can be conducted in order to introduce a term known as Particle Relaxation Time (τ_p). This simplification is conducted by multiple CFD softwares such as ANSYS or OpenFOAM and it can be seen in its user manual.[7]

The force balance, equation (79), can be divided by the particle mass m_p, giving the following expression:

$$\frac{dU_p}{dt} = \underbrace{\frac{1}{m_p}\frac{1}{2}\rho_f C_D A_p (U - U_p)|U - U_p|}_{} + \left(\frac{\rho_p - \rho_f}{\rho p}\right) g \tag{80}$$

Then, some rearranging on the first term can be performed. Multiplying top and bottom by d_p and μ, so that we get Re:

$$\frac{dU_p}{dt} = \frac{1}{\rho_p V_p}\left(\frac{C_D \mu}{2 d_p}\right) A_p (U - U_p) \underbrace{\left(\frac{\rho_f d_p |U - U_p|}{\mu}\right)}_{R_e} + \left(\frac{\rho_p - \rho_f}{\rho p}\right) g \tag{81}$$

Now substituting $V_p = \pi d_p^3/6$ and $A_p = \pi d_p^2/4$, we finally obtain:

$$\frac{dU_p}{dt} = \left(\frac{18\mu}{\rho_p d_p^2}\right)\left(\frac{C_D R_e}{24}\right)(U - U_p) + \left(\frac{\rho_p - \rho_f}{\rho_p}\right) g \tag{82}$$

The reason for this simplification is to introduce the particle relaxation time τ_p:

$$\tau_p = \frac{\rho_p d_p^2}{18\mu} \tag{83}$$

The particle relaxation time is used to characterize the capability of particles to follow sudden velocity changes in the flow. Small τ_p means that the particles follow the streamlines very closely.[20]

4.3 Problem Specification

OpenFOAM provides several Lagrangian solvers. For our simulation we will use `MPPICFoam` (Multiphase Particle-In-Cell method), a Lagrangian solver that has LPT (Lagrangian Particle Tracking) capabilities and can be used for modelling particles in continuum.[23]

In this solver the particles are assumed to be rigid, spherical and are thus described by diameter, density, coefficient of restitution (rebound properties) and coefficient of friction (tangential drag during collision). In this model, the particle dispersion in the turbulent flow field is accounted by two methods; stochastic tracking and particle cloud approach, which are further explained in reference [23].

Figure 19 depicts the exact geometry employed in this study. It's a 50x26 mm rectangle with a 6x6 mm square which is aimed to study the backflow behaviour of the fluid. Inside the geometry, dry air mixed with dispersed water droplets will flow, forming a spray pattern as the particles collide. This geometry has been designed in order to properly study the flow close to the water jets' impingement point. A larger geometry would result in unnecessary data that would be irrelevant to our study.

Figure 19 Geometry of the doublet injector simulation, Source: Own

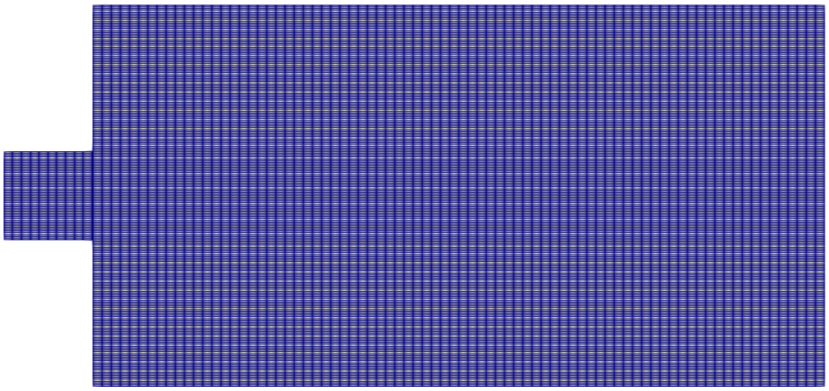


Figure 20 Generated Mesh geometry of the doublet injector simulation, Source: Own

In order to generate the mesh used for the simulation, the procedure from chapter 3.8 has been used. In a `blockMeshDict` dictionary, the vertices, blocks and number of cells per block of the geometry have been described. The result can be seen in figure 20 However, because the inlets are so small (0.5 mm), the mesh comprises a few cells that are much smaller than the others. As will be explained later, this problem can be solved by decreasing the simulation's time step. Figure 21. shows a close-up view of the simulation critical zone.

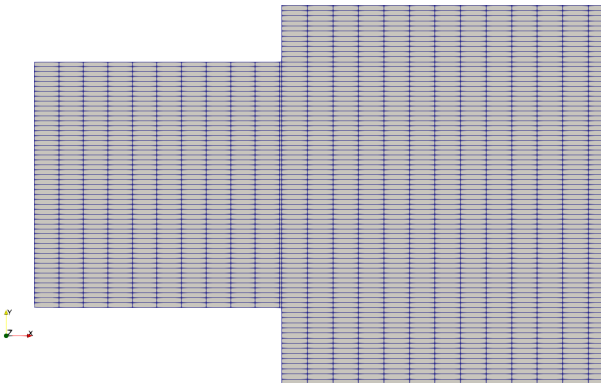


Figure 21 Close-up look of the injection zone mesh, Source: Own

Figure 22 shows the main case directory structure used in OpenFOAM in order to simulate the case. The `polyMesh` folder is the result of creating the mesh using the command `blockMesh`. The `.air` termination is used to scpecify that the continuous phase domain is air, thus the air properties must be used. The information provided by all of these files (see GitHub link in Appendix A) to the CFD solver (`MPPICFoam`) is described in the following chapters.

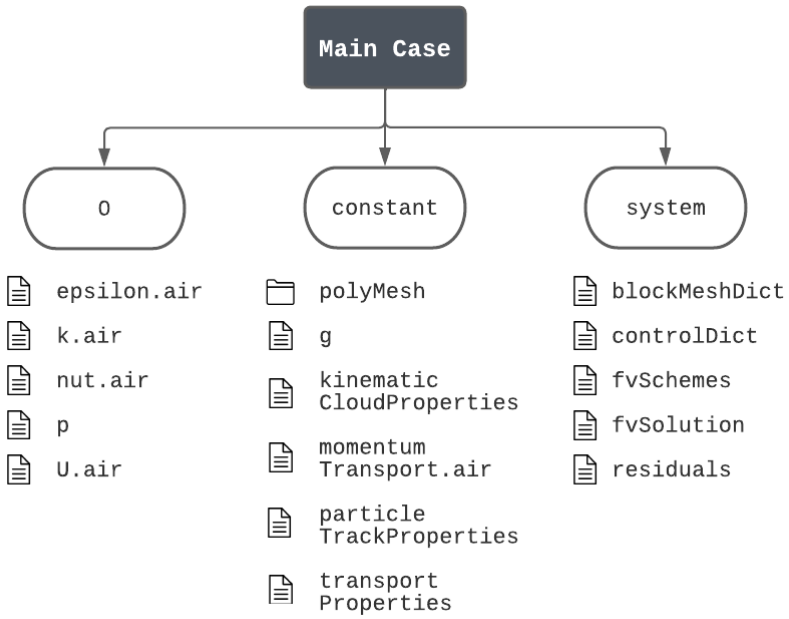


Figure 22 Main case directory structure for OpenFOAM, Source: Own

4.4 Physical Setup and Fluid Properties

It is important to define the properties of the fluids to be used. Table 6 shows the main properties (density, kinematic viscosity and surface tension) of air and water. The `constant`

directory provides files with physical attributes for the simulation. The air properties are defined in the `transportProperties` file, whereas the water droplets properties are defined in the `kinematicCloudProperties` file, as it they be treated as a Lagrangian dispersed phase. Further information about the Particle Tracking System included in the `kinematicCloudProperties` file is explained in chapter 4.7.

Properties of water ($20°C$)		**Spray properties**	
Density (kg/m^3)	1000	Injection pressure difference (bar)	10
Viscosity (m^2/s)	10^{-6}	Injection velocity (m/s)	40
Surface Tension (mN/m)	72	Nozzle diameter (mm)	0.5
Properties of air ($20°C$)		Mass flow rate (kg/s)	0.00785
Density (kg/m^3)	1.2	Injection angles	$30°, 40°, 50°, 60°,$
Viscosity (m^2/s)	$1.5 \cdot 10^{-5}$		$70°, 80°, 90°$

Table 6 Physical properties of water and air used in CFD computations, Source: Engineers Edge [34]

The `g` file contains the definition of gravity acceleration, defined as: `(9.81 0 0)`

The spray properties are empirical values defined in order to be accurate to the real life experiment that will be performed. The injection pressure difference and nozzle diameter values correspond to the ones that will be used in the physical experiment, whereas the injection velocity has been obtained using a procedure described in chapter 4.6. As specified in the table above, 7 different angles, from 30°to 90°will be tested subjected to the same initial conditions.

4.5 Turbulence Modeling

In order to apply one of the turbulent models described on this project, we must first determine the conditions under which our flow will operate. It will be an external free surface flow with very little pressure gradients. Under these conditions, the most suitable turbulent model for our simulation is the k-ε model.

The k-ε model has been proved to be reliable for free surface flow regions with small pressure gradients. As already stated in chapter 3.7.1, it is a two-equation model. This means that in addition to the conservation equations, it solves two transport equations which account for the convection and the diffusion of turbulent kinetic energy. The two transported variables are turbulent kinetic energy (k), which determines the energy in turbulence, and turbulent dissipation rate (ε), which determines the rate of dissipation of turbulent kinetic energy. [33]

The `system/momentumTransport.air` file allows to decide wether to use a RAS turbu-

lent model or a laminar model. In our case, as the `kEpsilon` model will be used, three new folders have to be added to the `0` directory with their corresponding intial values and boundary conditions. These files are:

- `epsilon.air`: Turbulent kinetic energy dissipation rate
- `k.air`: Turbulent kinetic energy
- `nut.air`: Turbulent viscosity

In most CFD simulations, values for turbulence variables at the inlets must be specified, and this necessitates making a more or less educated guess about the incoming turbulence.

Estimating the turbulence model variables, like turbulent kinetic energy or dissipation rate, directly is often difficult. Instead it is easier to consider variables such as the incoming turbulence intensity (I) and turbulent length scale (l) or eddy viscosity ratio. A procedure from reference [33] will be followed:

The first step is to evaluate the injection velocity of the water droplets; in order to calculate the injection velocity, the following procedure has been used. This accounts for the flow of liquid through a simple orifice (e.g. a round drilled hole):

Recalling Euler's equation of motion;

$$\frac{\partial p}{\rho} + UdU + gdZ = 0 \tag{84}$$

This can be integrated between two points to obtain Bernoulli's equation:

$$\int \frac{dp}{\rho} + \int UdU + \int gdZ = 0 \tag{85}$$

$$\frac{p}{\rho} + \frac{U^2}{2} + gZ = ct \tag{86}$$

This equation allows to find the exhaust velocity of a fluid through a hole;

$$U = \sqrt{\frac{2\Delta P}{\rho}} \tag{87}$$

In our case, the following values have been used to calculate the injection velocity:

$$\Delta P = 10 \text{ bar}; \rho = 1000\ kg/m^3$$

$$U = 44,72 \text{ m/s} \approx 40 \text{ m/s} \tag{88}$$

Now, it is possible to make an estimation of the Reynolds number of the flow. As it is a multiphase simulation, both phase Reynolds will be estimated:

$$R_{e_w} \approx \frac{\rho_w U L}{\mu_w} = \frac{1000 \cdot 40 \cdot 5 \cdot 10^{-4}}{10^{-3}} = 2000 \tag{89}$$

$$R_{e_{air}} \approx \frac{\rho_{air} U L}{\mu_{air}} = \frac{1.225 \cdot 40 \cdot 5 \cdot 10^{-4}}{1.5 \cdot 10^{-5}} = 1633.3 \tag{90}$$

Where U is the injection velocity and L is the inlet orifice diameter. Both values show that the case is a low Reynolds scenario, thus, little turbulence will appear.

The turbulence intensity, often referred to as turbulence level, is defined as:

$$I = \frac{u'}{U} \tag{91}$$

Where u' is the root-mean-square of the turbulent velocity fluctuations and U is the mean velocity. When setting boundary conditions for a CFD simulation it is often necessary to estimate the turbulence intensity on the inlets. According to reference [33], as we are considering a medium-turbulence case (Flow in not-so-complex devices like large pipes, ventilation flows etc. or low speed flows with low Reynolds number), the turbulence intensity (I) is between 1% and 5%. Thus, the following turbulence intensity will be assumed:

$$I \approx 5\% \tag{92}$$

On the other hand, the turbulence length scale, (l) , is a physical quantity describing the size of the large energy-containing eddies in a turbulent flow. It is easy to guess a reasonable value for the turbulent length scale since it is a quantity that can be linked to the physical size of the problem. The turbulent length scale should typically not exceed the problem's dimension, in our case it will be set to the inlet size:

$$l \approx 1 \text{ mm} \tag{93}$$

Now, the turbulent energy at the inlet (k) can be computed as:

$$k = \frac{3}{2}(UI)^2 = \frac{3}{2}(40 \cdot 0.05)^2 = 6 \tag{94}$$

Following, the turbulent dissipation rate (ε) can be computed using the following formula:

$$\varepsilon = C_\mu \frac{k^{\frac{3}{2}}}{l} = 0.09 \cdot \frac{6^{\frac{3}{2}}}{10^{-3}} = 1322 \tag{95}$$

Where C_μ is a turbulence model constant which usually has a value of 0.09. The value of the turbulent dissipation rate obtained is very high and for this reason turbulence will be negligible in the simulation.

4.6 Initial and Boundary Conditions

Now that the physical properties and the turbulence model have been defined, it is necessary to give our simulation a starting point in order to find a way to the final solution and reach convergence. The following figure shows the different boundary patches used in our geometry:

Figure 23 Boundary conditions of the main case geometry, Source: Own

These patches are defined in `system/blockMeshDict` dictionary and all our variables must be initialized in them. Note that the `frontAndBack` patch is not represented in figure 23 but it can be understood as the symmetry plane of our simulation.

Below is presented the boundary conditions used for our simulation. Multiple boundary conditions may be used for some of them, which means that multiple setups can be created.

However, this configuration has provided good results.

	InternalField	**upperInlet**	**lowerInlet**
U.air	`uniform (0 0 0)`	`fixedValue (0 0 0)`	`fixed Value (0 0 0)`
p	`uniform 0`	`fixedFluxPressure value uniform 0`	`fixedFluxPressure value uniform 0`
k.air	`uniform 6`	`fixedValue value uniform 6`	`fixedValue value uniform 6`
epsilon.air	`uniform 1320`	`fixedValue value uniform 1320`	`fixedValue value uniform 1320`
nut.air	`uniform 0`	`calculated value uniform 0`	`calculated value uniform 0`
	outlet	**walls**	**frontAndBack**
U.air	`inletOutlet`	`noSlip`	`symmetry`
p	`fixedValue value uniform 0`	`fixedFluxPressure value uniform 0`	`symmetry`
k.air	`inletOutlet`	`kqrWallFunction value uniform 6`	`symmetry`
epsilon.air	`inletOutlet`	`epsilonWallFunction value uniform 1320`	`symmetry`
nut.air	`calculated value uniform 0`	`nutkWallFunction value uniform 0`	`symmetry`

Table 7 Boundary conditions specified in folder 0 for the problem case fields.

As it can be seen, the injection velocity of the continuous air phase is zero. This is because the phase being injected into the domain is water, whereas the air is still until the water droplets affect it and cause it to move. The injection velocity of the water droplets is defined in `constant/kinematicCloudProperties` dictionary as:

Upper inlet velocity: $(cos(\alpha_{\text{imp}}),\ -sin(\alpha_{\text{imp}}),\ 0))$

Lower inlet velocity: $(cos(\alpha_{\text{imp}}),\ sin(\alpha_{\text{imp}},\ 0))$

Where α_{imp} corresponds to the impingement half-angle.

4.7 Particle Tracing System

As already stated, the Lagrangian Particle Tracking method, introduced in Chapter 4.2, is required for spray modelling. This approach can be modelled in OpenFOAM by defining the properties of the particles and the way they interact with each other.

The `kinematicCloudProperties` dictionary is used to describe the additional information required for Lagrangian-Eulerian simulations. It specifies the particle injection rate and velocity, particle size and distribution, particle forces, and the empirical model to be used. The `kinematicCloudProperties` GitHub file includes comments on each variable and its significance in the simulation.

ParticleForces Type	Description
`sphereDrag`	Drag force for isolated spherical particles (very dilute particle density) at low Reynolds numbers.
`ShillerNaumannDrag`	Drag force for isolated spherical particles (very dilute particle density) from low to high Reynolds number.
`NonSphereDrag`	Similar to `SchillerNaumannDrag` but for non-spherical parcels. You need to specify the sphericity of parcels, phi, for this model.
`WenYuDrag`	Spherical drag force for multiphase flows from very dilute up to moderate density of particles.
`ErgunWenYuDrag`	Similar to `WenYuDrag` but for the whole range of fluid volume fraction.
`distortedSphereDrag`	Drag force for distorted particles.
`SaffmanMeiLiftForce`	Saffman-Mei particle lift force model applicable to spherical particles.
`pressureGradient`	Calculates the pressure gradient force.
`virtualMass`	Calculates the virtual mass force.
`gravity`	Calculates the gravity force.
`paramagnetic`	Calculates the particle paramagnetic (magnetic field) force.

Table 8 Different types of `particleForces` sub-dictionary in `kinematicCloudProperties` file, Source: Hamidreza Norouzi (2020)[41]

Table 8 collects several different types of `particleForces` sub-dictionaries that can be

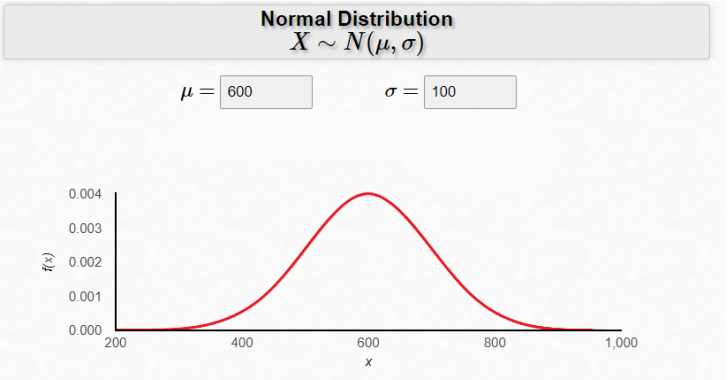
used to model the spray for our simulation.

The `wenYuDrag` model has been chosen to model the drag of our sphere particles as it can account for multiphase flows from very dilute up to moderate density particles, which is suitable for our case. In addition, the `gravity` model has also been included, even though gravity could be neglected as the particles are very small and have very little mass.

The particle's size is also specified in the `kinematicCloudProperties` dictionary. Reference [11] allows to establish a spatial distribution of mean diameter of water droplets produced by different nozzles. This data has been roughly used to create a normal distribution of diameters in order to recreate a real nozzle spray. Figure 24 shows the normal distribution defined in the `kinematicCloudProperties` dictionary as:

```
1       normalDistribution
2       {
3           expectation   600e-6;
4           variance      100e-6;
5           minValue      200e-6;
6           maxValue      900e-6;
7       }
```

(a) `normaldistribution` code



(b) Normal distribution function ($\times 10^{-6}$)

Figure 24 Normal size distribution of droplets, Sorce:[26]

4.8 Time Step Control

The Courant number is a dimensionless value that represents the time a particle stays in one cell of the mesh. Its definition can be seen below:

$$C_o = \frac{U\Delta t}{\Delta x} \tag{96}$$

Where Δx corresponds to the length of a cell, Δt is the time step and U is the velocity of the fluid.

Over a time step Δt, the flow moves a distance $U\Delta t$ across the cell. The bigger the time step, the further the flow is going to move across the cell. The Courant Number is the ratio of these two lengths, i.e. the fraction of the cell that the flow moves across it in a time step. For example, a Courant Number of 0.3 is telling us that in a time step, the flow is moving 30% of

the distance across the cell. However, if the Courant number is bigger than 1, it means that the flow is moving further than the length of the cell in a given time step.

When the Courant number is bigger than 1, instabilities are amplified throughout the domain and may cause divergence of the simulation. In order to decrease the Courant number we can either:

- Decrease the time step Δt
- Coarsen the mesh, i.e. increase Δx

CFD solvers use the maximum Courant Number for the stability calculations and for the time step adjustment. Maximum Courant Number recommendations may vary for different types of flow simulations, most often the maximum Courant number should be below 1.0. This maximum Courant number restricts the size of the time step that we can apply in the CFD simulation, if this is higher than 1, the time step will have to be reduced. There exist many ways to reduce the time step of the simulation.

The first thing we need to think about is what happens in a conventional transient simulation when we chose a fixed time step:

First you set up your CFD simulation and you choose a fixed time step size to run the simulation. Afterwards, the CFD solver will compute the flow field U and all the other fields and will compute the Courant Number field for all of the cells of the mesh. After that, it will evaluate the maximum C_o and report it on the screen so that the user can see it in the output file.[20]

Finally, as we are using a fixed time step, the next iteration will not modificate the Δt value and will repeat the previous steps repeatedly. If the C_o appears to be too large, the CFD code may diverge or even crash, making it impossible to find a solution. Even if it still converges to a solution, this would not be physically accurate and the time step would need to be reduced manually and run the simulation again.[20]

Nonetheless, there exists a way to adjust the time step automatically. When enabling an adjustable time step for the simulation we would start the process by choosing an initial time step size and a maximum C_o that should not be exceeded during the process. Like before, the CFD code would compute the velocity field U as well as all the other variables in order to compute the Courant Number field. Once again it would evaluate the maximum C_o for that field. The last step of an adjustable time stepping routine is comparing the maximum Courant number from the field and the maximum initial Courant number specified by the user. If the maximum Co in the field is too high, greater than the limit applied ($C_o > C_{o,max}$), the CFD code would automatically reduce the time step to bring the C_o down in the next iteration.[20]

In the `controlDict` dictionary an `adjustTimeStep` model has been applied using a maximum Courant number of 1 (`maxCo 1`) and a maximum time step of 10^{-3} (`maxDeltat 1e-3`)

4.9 Postprocessing and Results

The simulation analysis concludes by running the simulation and the results being evaluated and postprocessed. The main post-processing tool provided with OpenFOAM is the reader module to run with ParaView, an open-source, visualization application. `paraFoam` is a script that can be typed in the main case terminal and that launches ParaView using the reader module supplied with OpenFOAM.

Once within the ParaView application, the Properties window (Figure 25) for the case module includes the Parameters panel that contains the settings for mesh, fields and global controls.

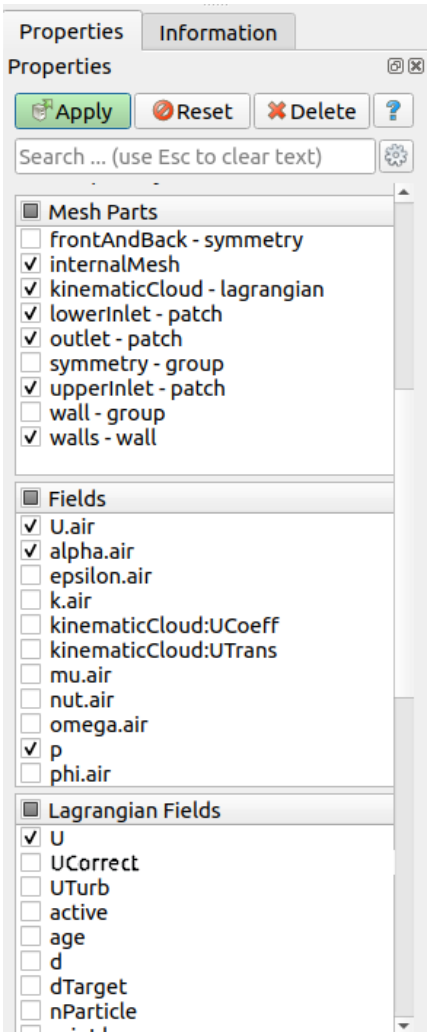


Figure 25 The properties panel for the case module, Source: Own

The user can select mesh and field data which is loaded for all time directories into ParaView. The current display is showing both `internalMesh` and `kinematicCloud-lagrangian` fields, as well as all the boundary patches except for the `frontAndBack` patch. after making any changes to any selections, the user must click the Apply button.

However, before diving into the results, we must ensure that the simulation reaches convergence (RQ-2). This can be done by using the residuals. In general, CFD codes use an iterative approach to calculate the solution field. As a result, calculating the final solution profile may need multiple iterations, with the initial iterations being a rough estimate and the later iterations being extremely near to the real value until convergence is achieved.

The residual is one of the most fundamental measures of an iterative solution's convergence, as it directly quantifies the error in the solution of the system of equations. In a CFD analysis, the residual measures the local imbalance of a variable in each control volume. Therefore, every cell in the model will have its own residual value for each of the equations being solved. In an iterative numerical solution, the residual will never be exactly zero. However, the lower the residual value is, the more numerically accurate the solution.[13]

According to reference [13], residual levels of 1E-4 are considered to be loosely converged, levels of 1E-5 are considered to be well converged, and levels of 1E-6 are considered to be tightly converged. For complicated problems, however, it's not always possible to achieve low residual levels

The residuals can be computed while the simulation is running by adding in the `system` file a dictionary called `residuals`, which includes the `p` and `U.air` fields.

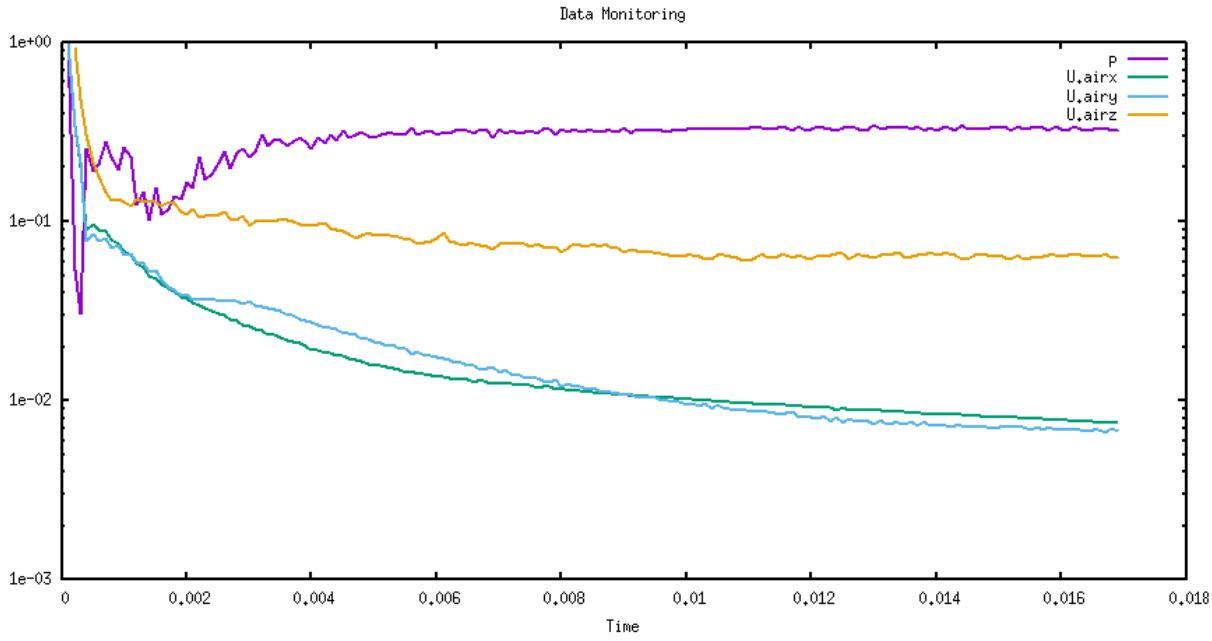


Figure 26 Solution residuals, Source: Own

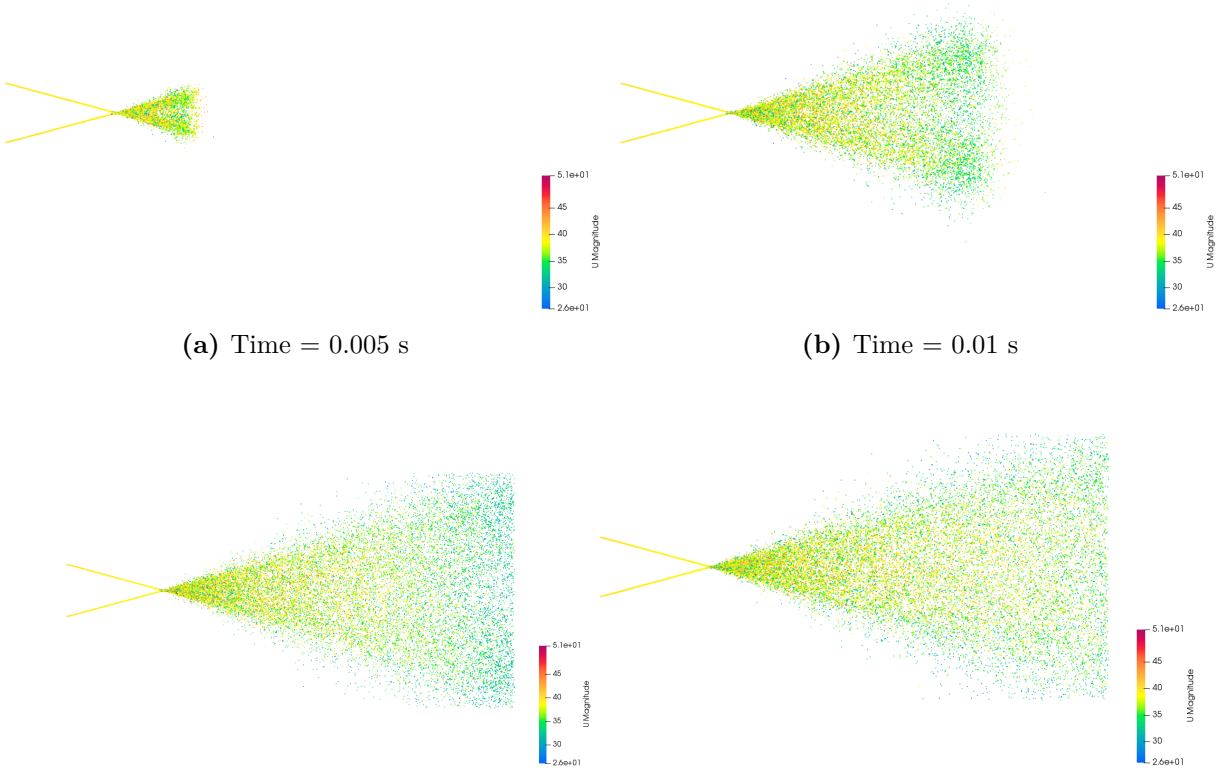
As can be seen in Figure 26, after the initial startup period, the solution imbalances gradually decrease as the solution progresses. However, the residual values are all higher than expected, which means that, even if the solution is slightly converged, it may not be a good representation of the true physical behaviour.

Down below, the different fields (Eulerian and Lagrangian) will be displayed individually in order to interpret and understand the resulting data from all the simulations. The `U.air` velocity field from the `internalMesh` mesh part, on the one hand, and the `U` velocity field from the `kinematicCloud-lagrangian` mesh part, on the other hand.

In addition, a bicoloured representation of the spray mixing will be displayed. This representation has been obtained by using the fluid dispersion visualizer code (see GitHub link in Appendix A) which gives a specific colour to each parcel or particle depending on its initial "y" position; red for $y > 0$ and blue for $y < 0$. The data of all the particle's coordinates for each time step has been obtained from the "*save data*" option in ParaView. The result, which can be seen in the following chapters, allows to see the mixing uniformity of the spray.

In case the animations do not work refer to Appendix B.

4.9.1 Impingement Angle =30°



(a) Time = 0.005 s **(b)** Time = 0.01 s

(c) Time = 0.015 s **(d)** Time = 0.02 s

Figure 27 Water droplets spray velocity field (30°), Source: Own

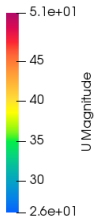


Figure 28 Animated water droplets spray velocity field (30°), Source: Own

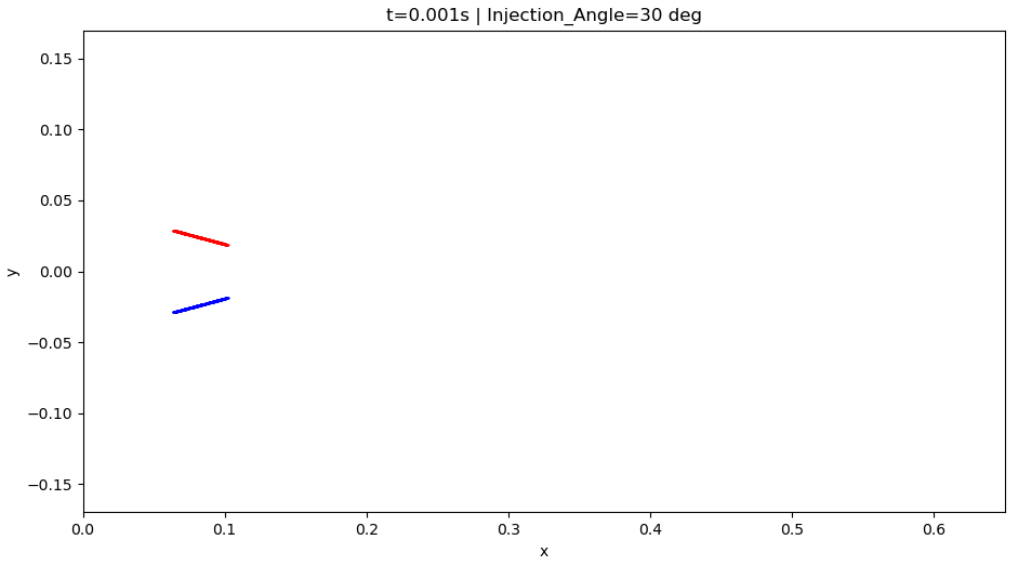
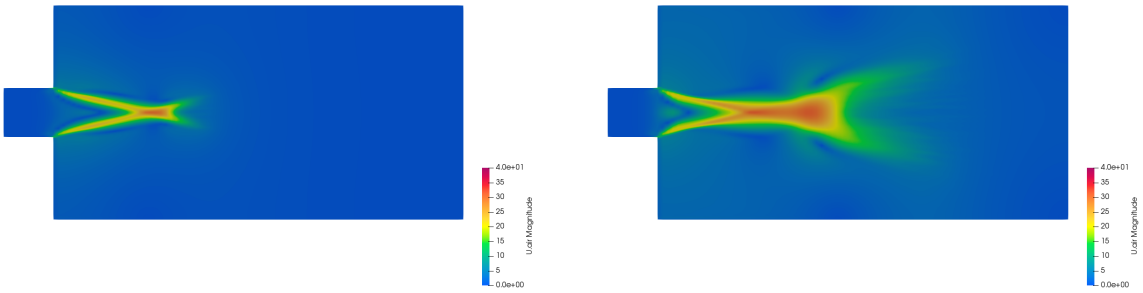
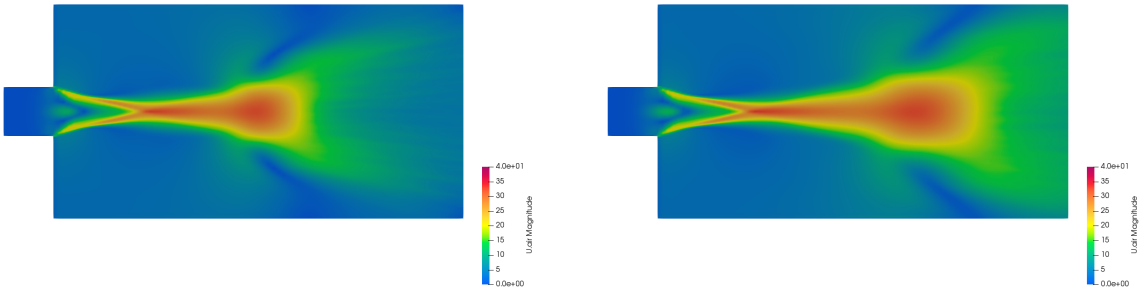


Figure 29 Animated water droplets spray bicolour mixture (30°), Source: Own



(a) Time = 0.05 s

(b) Time = 0.01 s

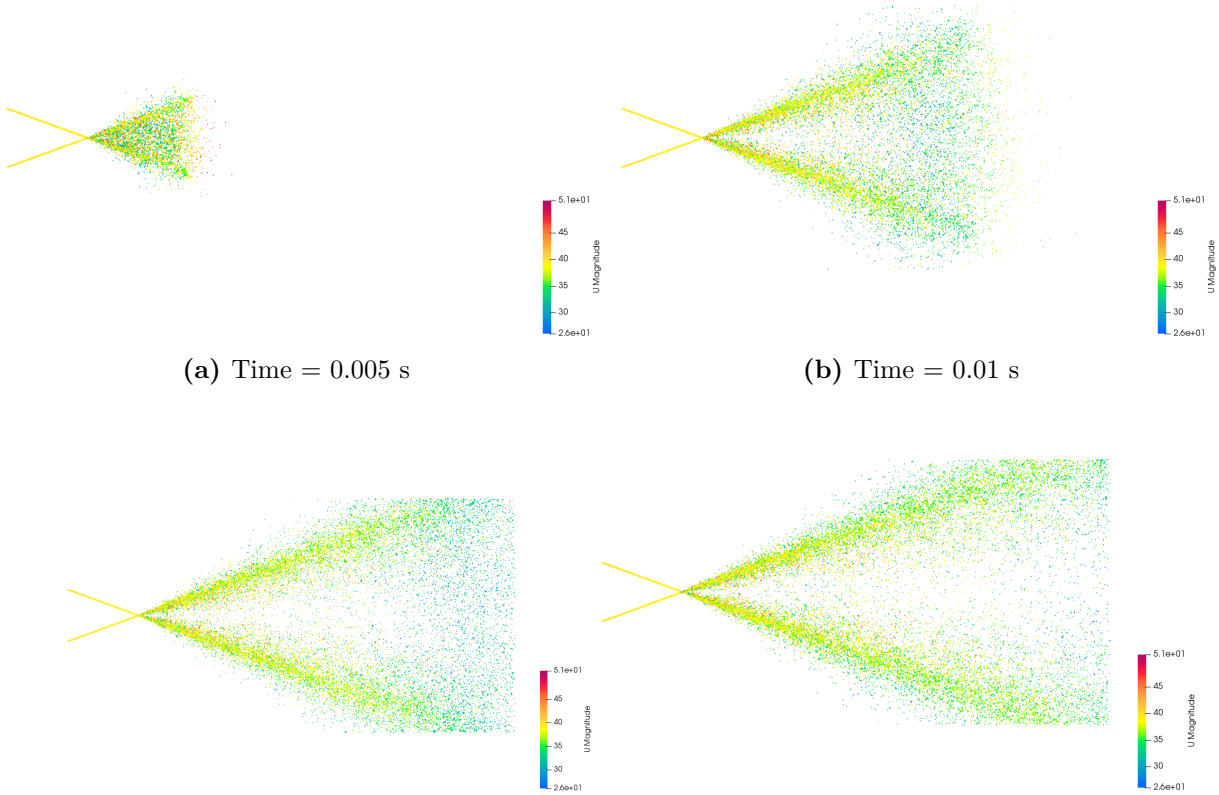


(c) Time = 0.015 s

(d) Time = 0.02 s

Figure 30 Air continuous phase velocity field (30°), Source: Own

4.9.2 Impingement Angle = 40°



(a) Time = 0.005 s **(b)** Time = 0.01 s

(c) Time = 0.015 s **(d)** Time = 0.02 s

Figure 31 Water droplets spray velocity field (40°), Source: Own

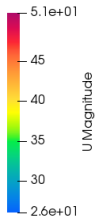


Figure 32 Animated water droplets spray velocity field (40°), Source: Own

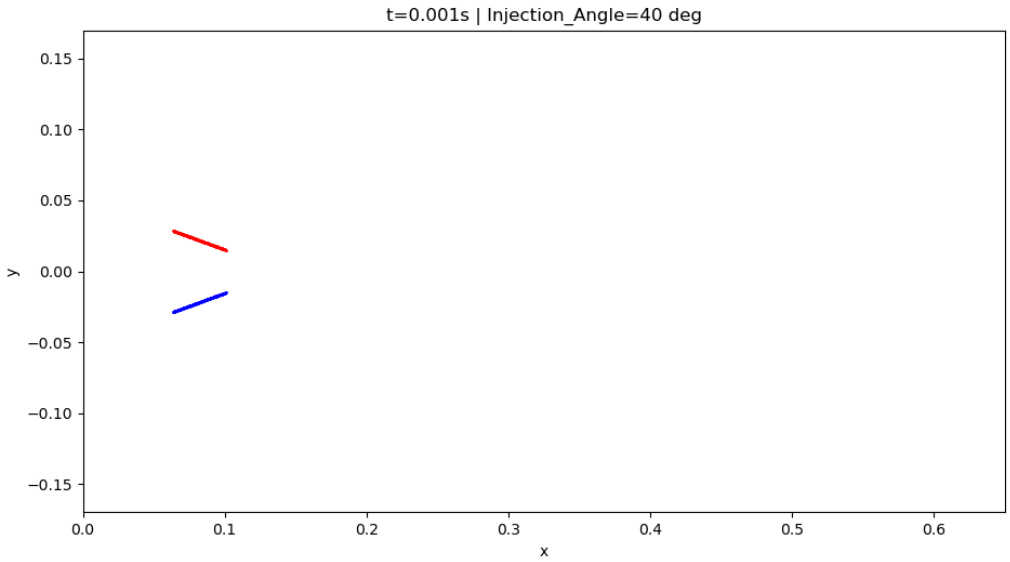
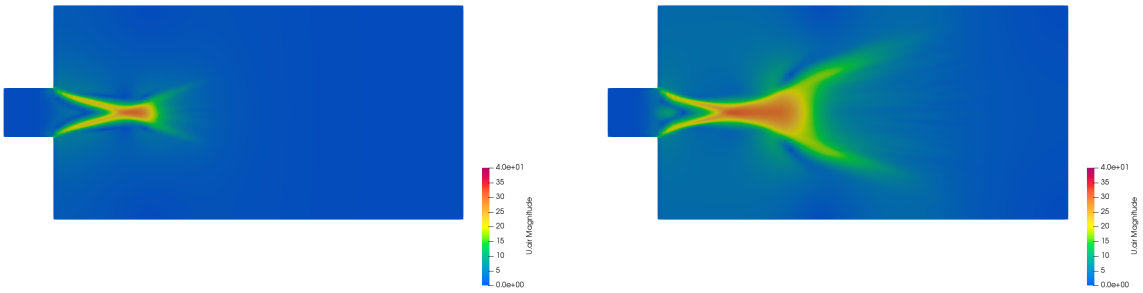
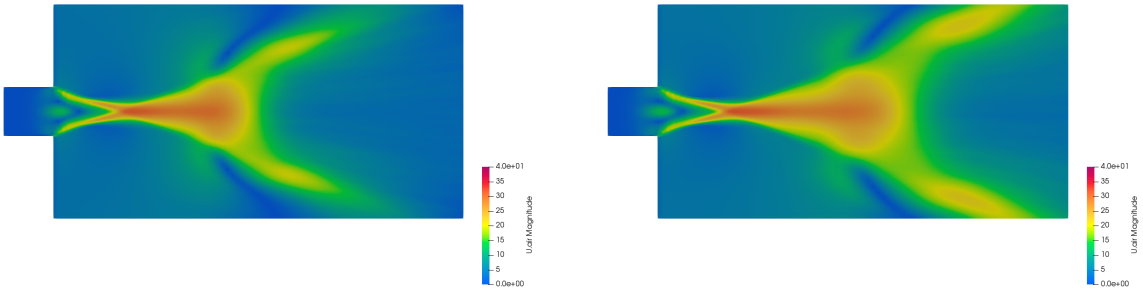


Figure 33 Animated water droplets spray bicolour mixture (40°), Source: Own



(a) Time = 0.05 s

(b) Time = 0.01 s

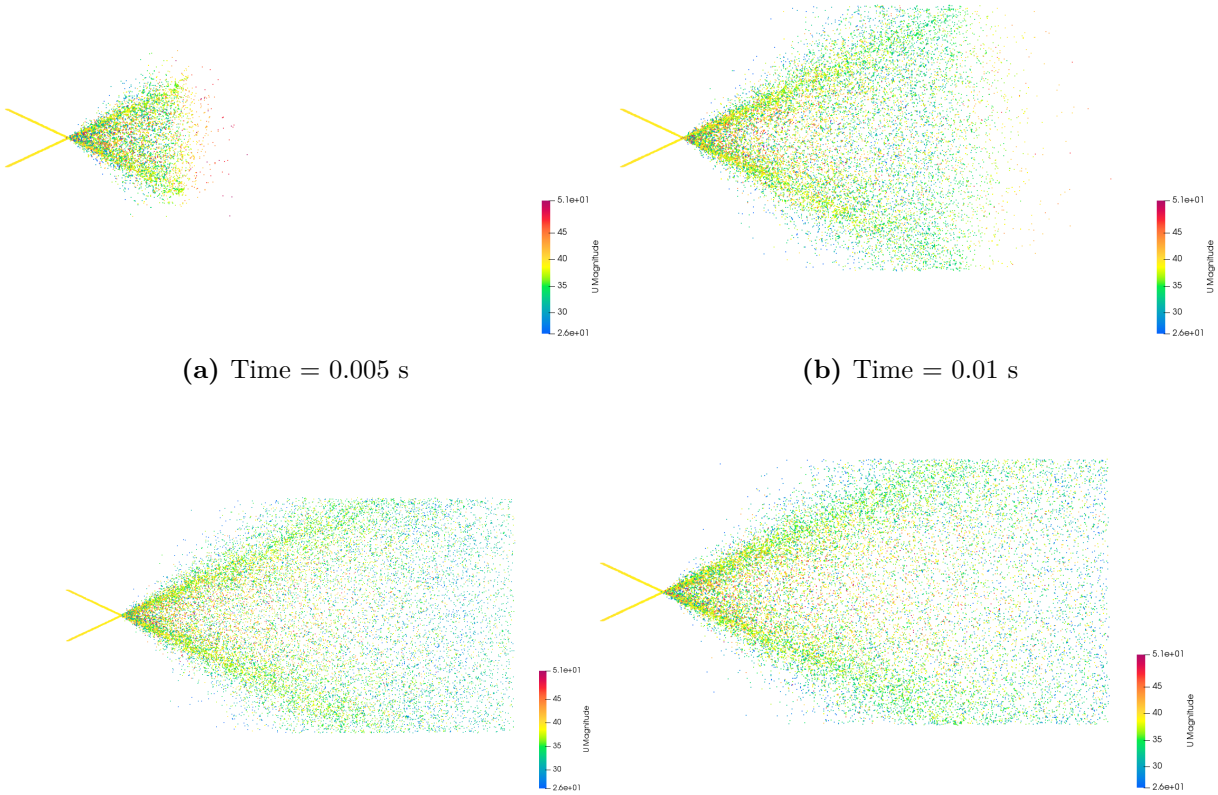


(c) Time = 0.015 s

(d) Time = 0.02 s

Figure 34 Air continuous phase velocity field (40°), Source: Own

4.9.3 Impingement Angle = 50°



(a) Time = 0.005 s **(b)** Time = 0.01 s

(c) Time = 0.015 s **(d)** Time = 0.02 s

Figure 35 Water droplets spray velocity field (50°), Source: Own

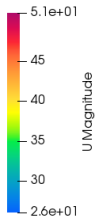


Figure 36 Animated water droplets spray velocity field (50°), Source: Own

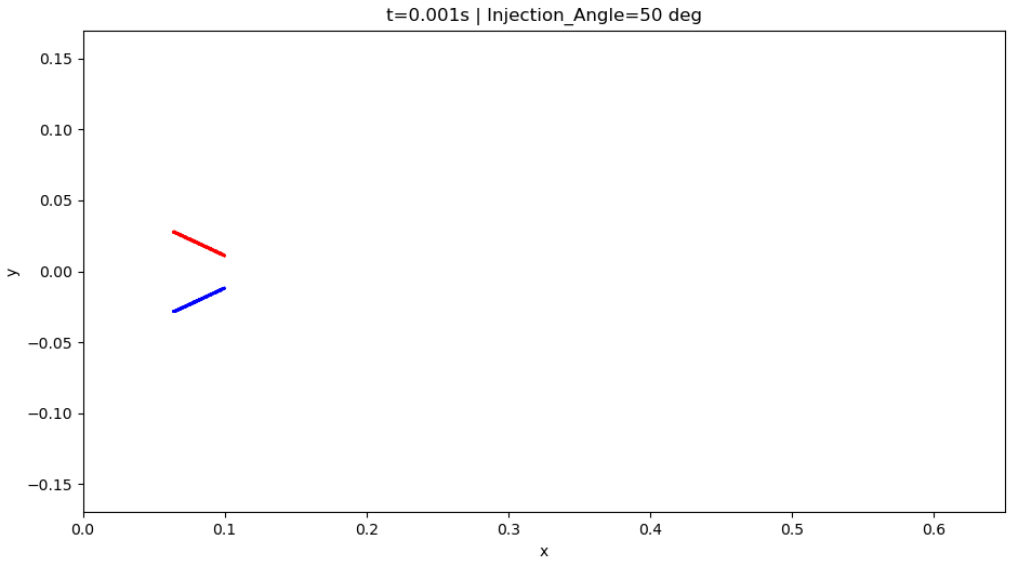
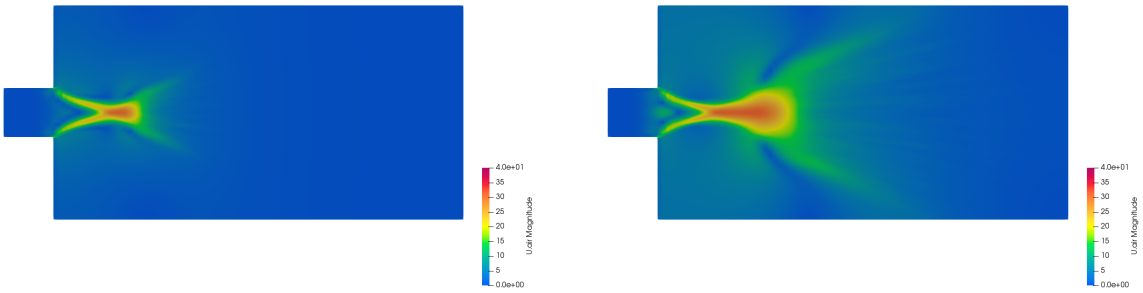
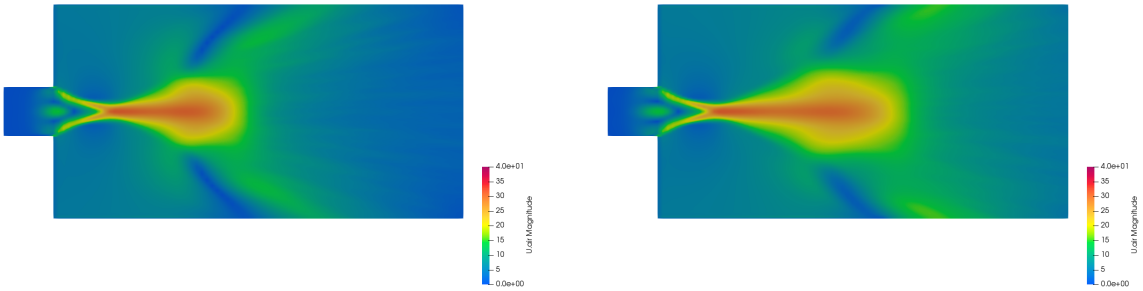


Figure 37 Animated water droplets spray bicolour mixture (50°), Source: Own



(a) Time = 0.05 s

(b) Time = 0.01 s

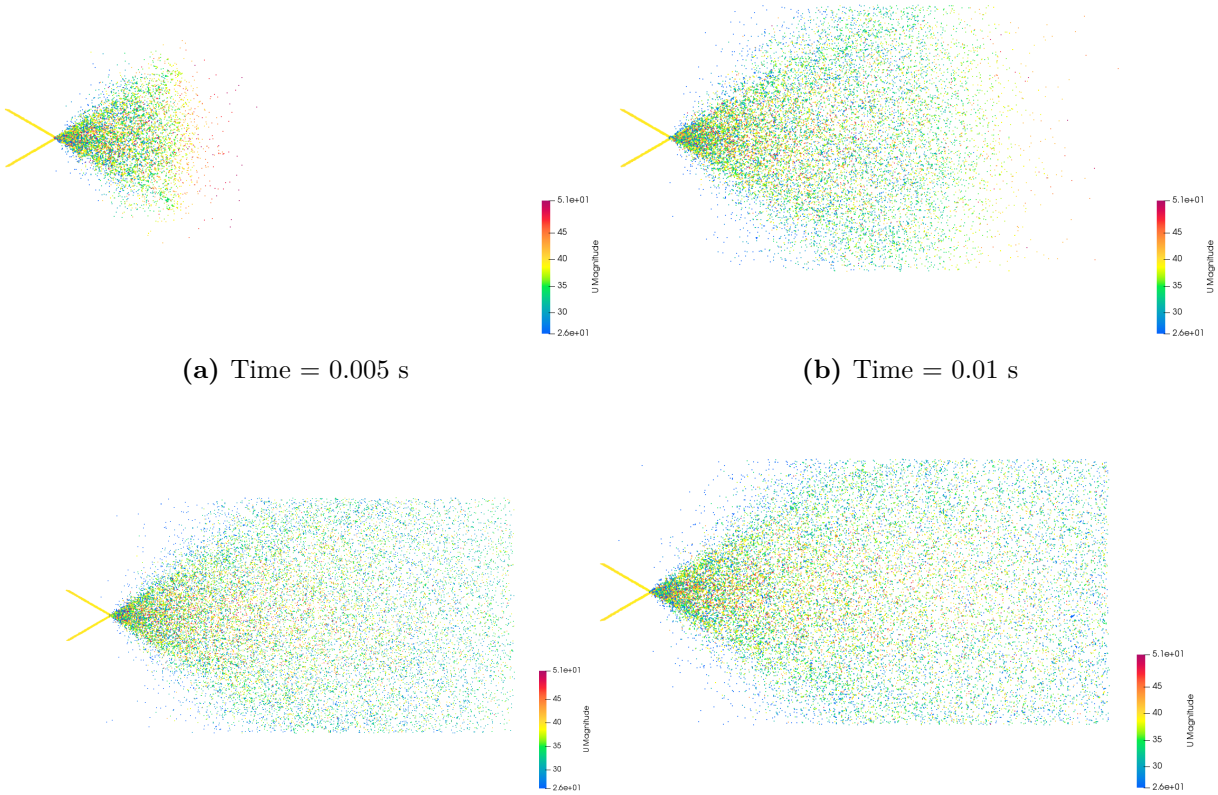


(c) Time = 0.015 s

(d) Time = 0.02 s

Figure 38 Air continuous phase velocity field (50°), Source: Own

4.9.4 Impingement Angle = 60°



(a) Time = 0.005 s **(b)** Time = 0.01 s

(c) Time = 0.015 s **(d)** Time = 0.02 s

Figure 39 Water droplets spray velocity field (60°), Source: Own

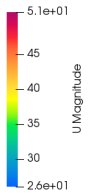


Figure 40 Animated water droplets spray velocity field (60°), Source: Own

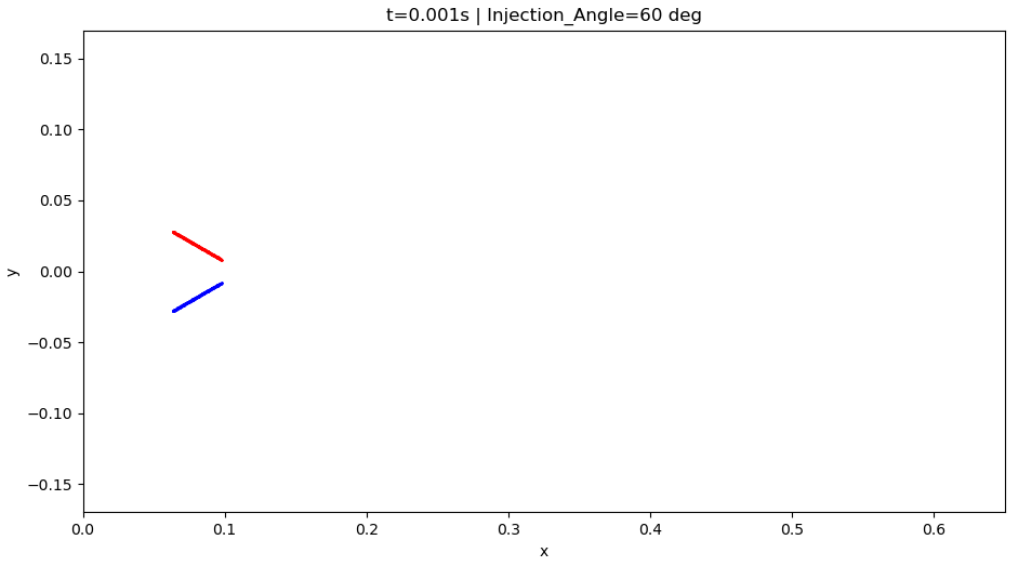
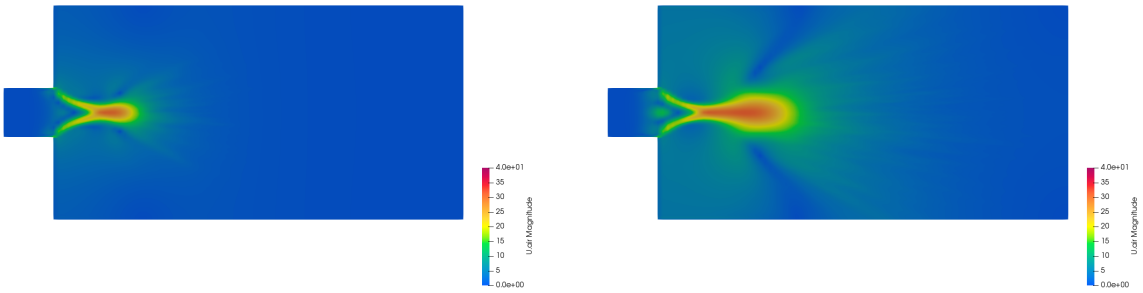
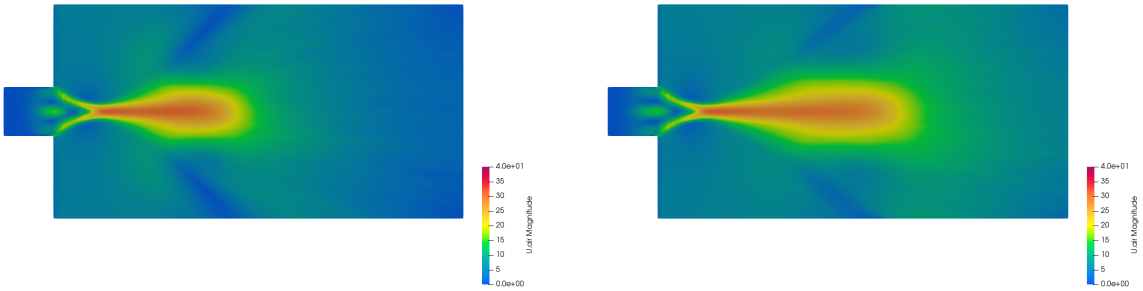


Figure 41 Animated water droplets spray bicolour mixture (60°), Source: Own



(a) Time = 0.05 s

(b) Time = 0.01 s

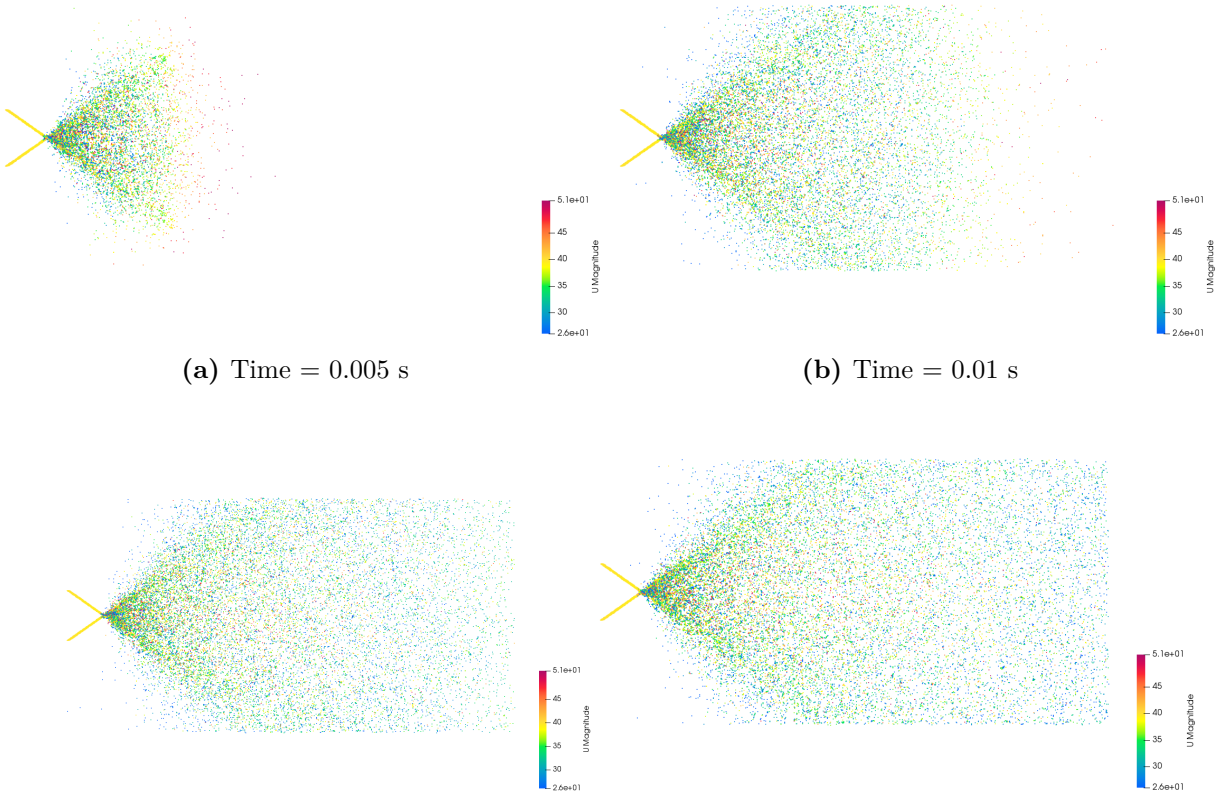


(c) Time = 0.015 s

(d) Time = 0.02 s

Figure 42 Air continuous phase velocity field (60°), Source: Own

4.9.5 Impingement Angle = 70°



(a) Time = 0.005 s **(b)** Time = 0.01 s

(c) Time = 0.015 s **(d)** Time = 0.02 s

Figure 43 Water droplets spray velocity field (70°), Source: Own

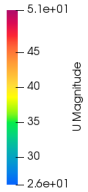


Figure 44 Animated water droplets spray velocity field (70°), Source: Own

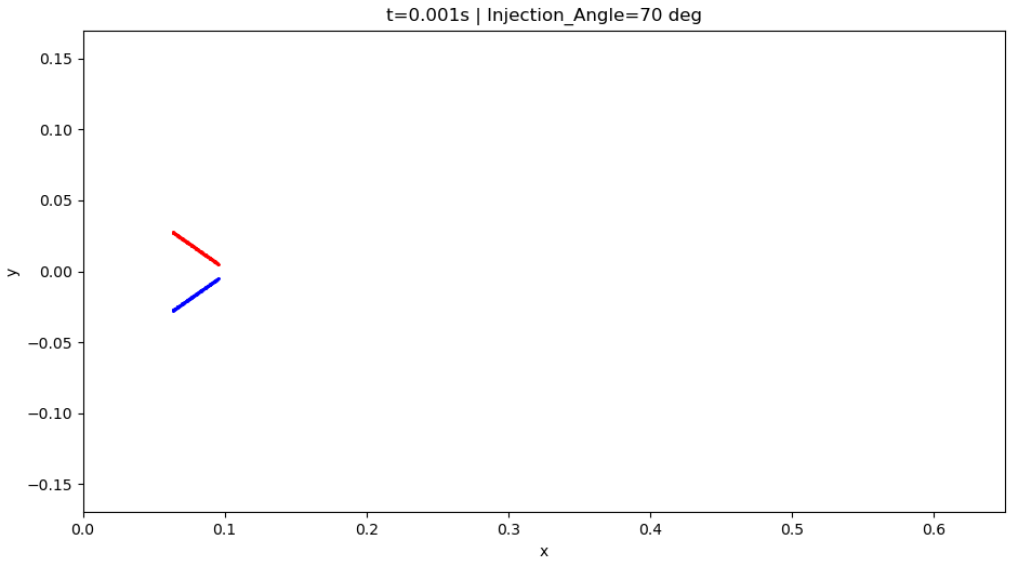
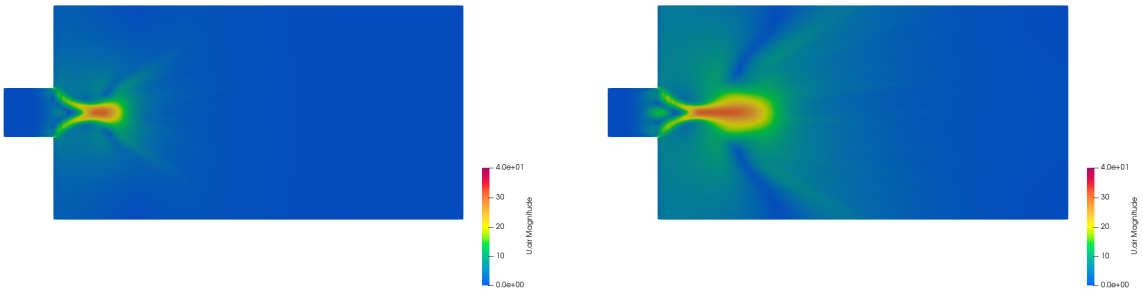
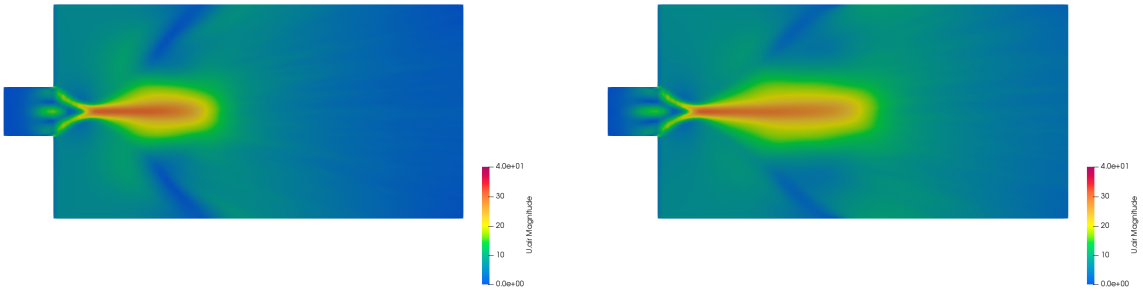


Figure 45 Animated water droplets spray bicolour mixture (70°), Source: Own



(a) Time = 0.05 s

(b) Time = 0.01 s

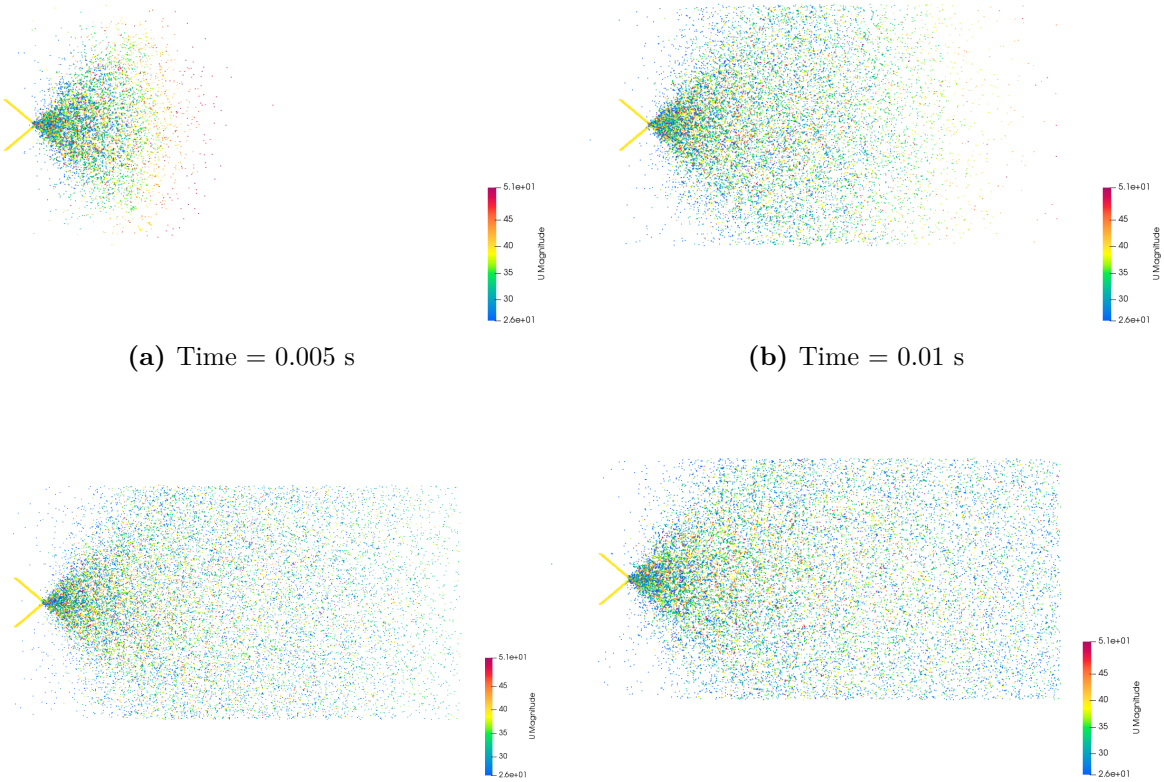


(c) Time = 0.015 s

(d) Time = 0.02 s

Figure 46 Air continuous phase velocity field (70°), Source: Own

4.9.6 Impingement Angle = 80°



(a) Time = 0.005 s **(b)** Time = 0.01 s

(c) Time = 0.015 s **(d)** Time = 0.02 s

Figure 47 Water droplets spray velocity field (80°), Source: Own

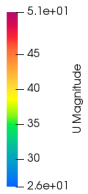


Figure 48 Animated water droplets spray velocity field (80°), Source: Own

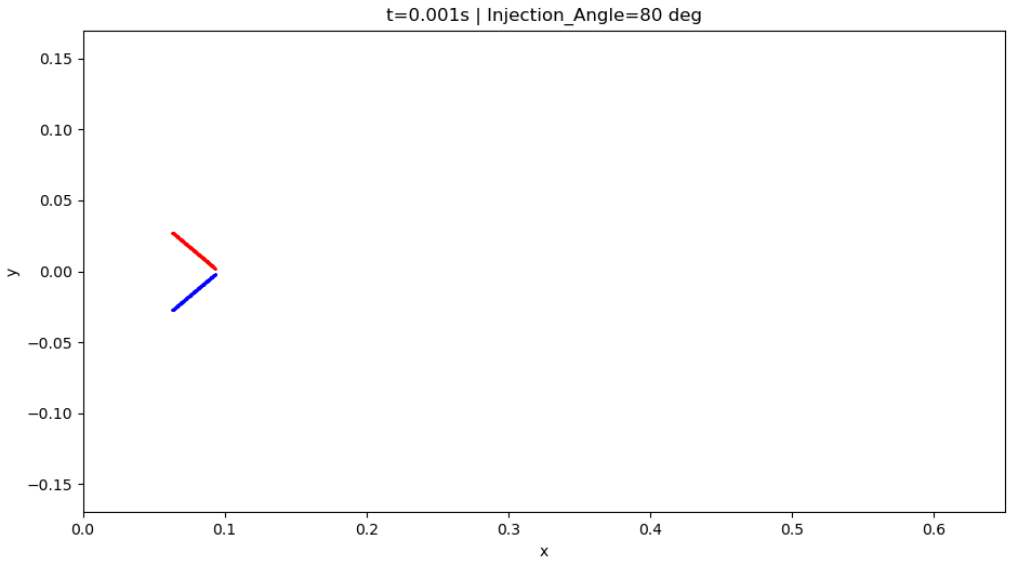
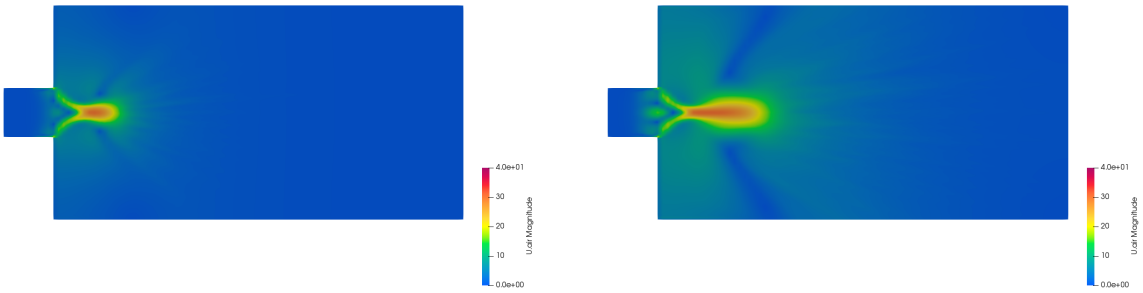
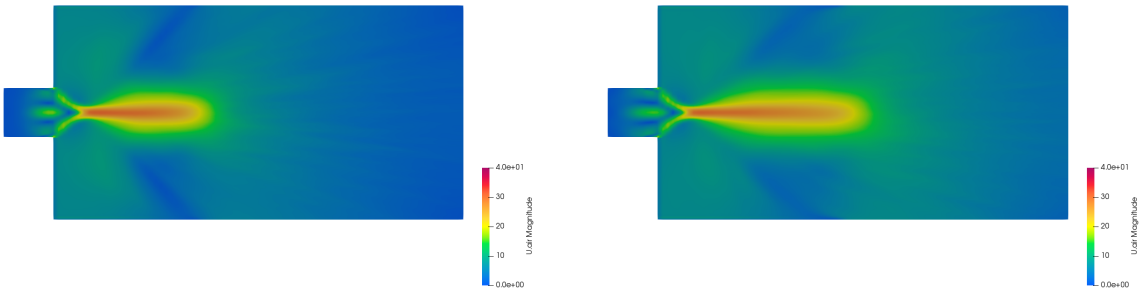


Figure 49 Animated water droplets spray bicolour mixture (80°), Source: Own



(a) Time = 0.05 s

(b) Time = 0.01 s

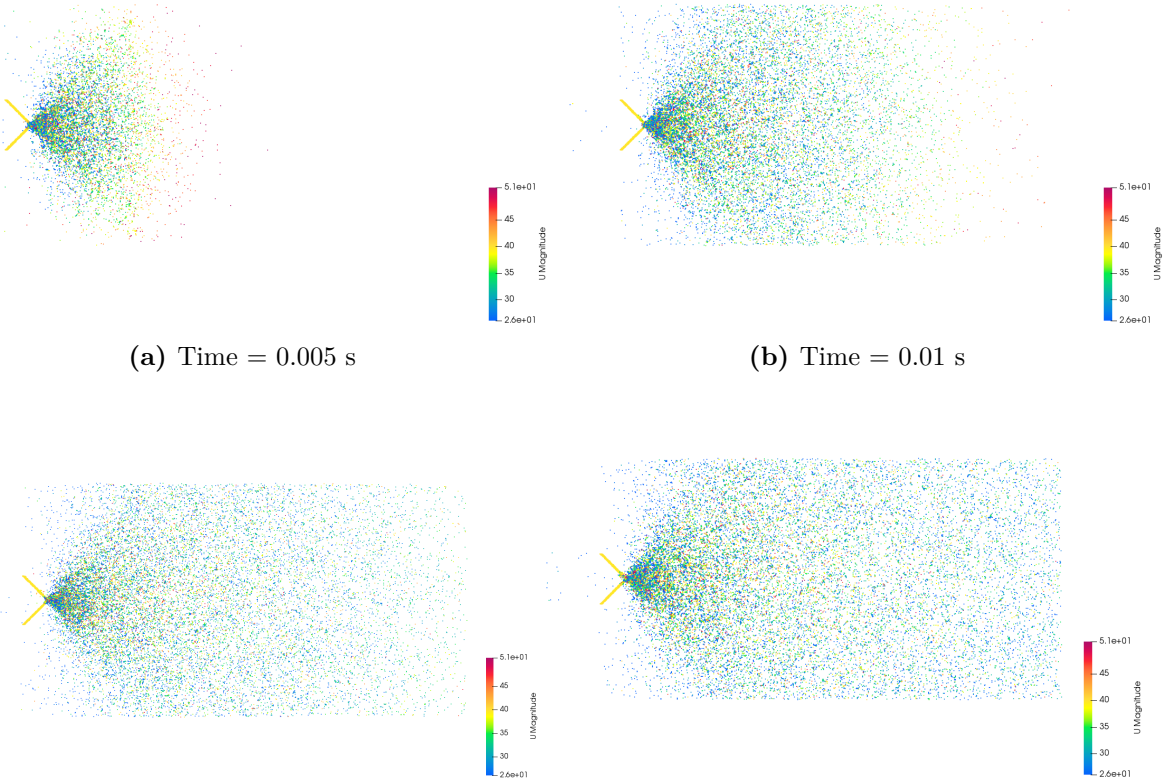


(c) Time = 0.015 s

(d) Time = 0.02 s

Figure 50 Air continuous phase velocity field (80°), Source: Own

4.9.7 Impingement Angle =90°



(a) Time = 0.005 s **(b)** Time = 0.01 s

(c) Time = 0.015 s **(d)** Time = 0.02 s

Figure 51 Water droplets spray velocity field (90°), Source: Own

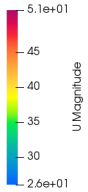


Figure 52 Animated water droplets spray velocity field (90°), Source: Own

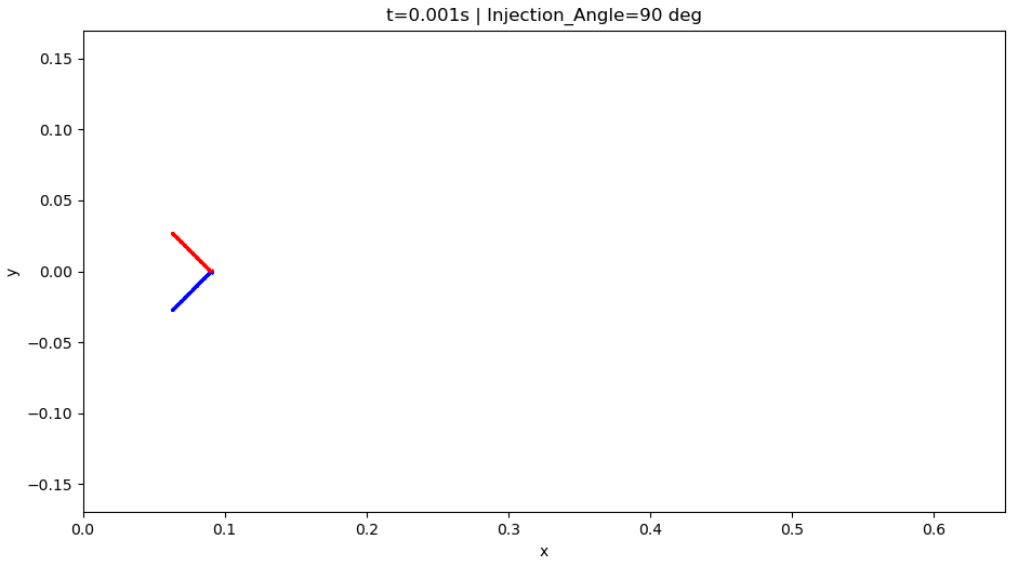
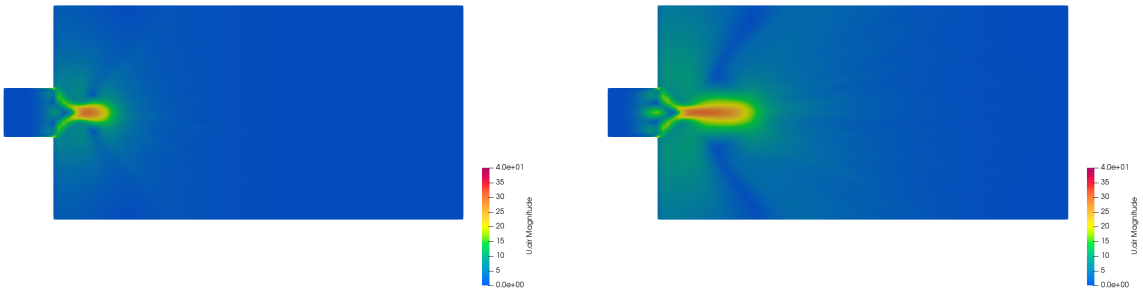
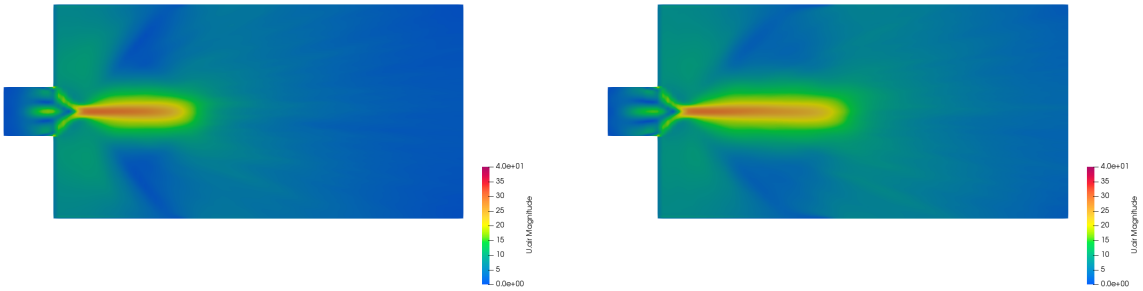


Figure 53 Animated water droplets spray bicolour mixture (90°), Source: Own



(a) Time = 0.05 s

(b) Time = 0.01 s



(c) Time = 0.015 s

(d) Time = 0.02 s

Figure 54 Air continuous phase velocity field (90°), Source: Own

4.10 Discussion

After the simulations have been completed and all the data has been gathered, the results can be discussed as well as extract some conclusions.

- The first thing to notice is that the mixing mode (see Figure 8) is transmissive for low angles and becomes well mixed at high angles. The bicoloured mixture images reveal that from 60°and above the spray has a more uniform concentration distribution than for lower angles, i.e. mixing uniformity is attained at angles greater than 60°. This would result in an homogeneous air-fuel mixing and improve the spray combustion efficiency. In addition, A concentration of droplets can be seen on the mixing zone, where the two jets collide. This zone was previously defined as the impingement point.

- One of the injection variables, the impingement distance, has not been taken into consideration. This distance varies with the angle of incidence; the lower the angle, the larger the impinging distance is. However, as the jets are already atomized when they exit the inlet (Euler-Lagrangian condition), the mixing uniformity will not depend on the impingement distance, hence this variable does not need to remain constant.

- It can be seen that the angular distribution of the atomized spray leaving the impingement point is also dependent on the impingement angle. The greater the angle, the greater the mass and droplets distributed at angles greater than 90°. According to the data, from 70°up to 90°there is a considerable increase in backflow, i.e. mass distributed at angles greaters than 90°. However, backflow would a big concern which, if real combustion occurred, could result in high heat flux to the injector face.

- The dispersed phase spray distribution spreads out at a much wider angle than the continuous phase distribution, which implies that the drag and other forces are deviating the particles from the continuous phase, i.e. the water particles do not follow the streamlines from the continuous phase. It is important to notice that the air has no initial velocity, so the continuous air field produced is a direct consequence from the water particles colliding with the surrounding air.
- It can also be observed a correlation between the cloud mean velocity and the impinging half-angle. The higher the angle is, the slower the particles tend to move in result of a more direct contact between them. This statement is noticeable by the spray velocity field colour; it gets more blue (less U magnitude) as the angle increases. The same can be observed on the air continuous phase field solutions; for high angle values, the air field solution tends to be slower.

Besides all the findings, it is important to take into consideration that the results may not be accurate enough nor comparable to the design of a real unlike impinging injector. This is the reason why a validation of the results will take place in an experimental analysis. In

addition, the simulation has been conducted in a 2D representation, which has enabled to get results much faster reducing the computational time at the expense of simplifying the problem. A 3D representation was planned to be simulated, however, it had a really high computational cost and most of the times it crashed and was unable to find a converged solution. It would have been an asset to implement this 3D simulation as the collisions between particles would also occur on the z axis and the spray would have been seen from different angles.

Recalling chapter 2.4.2, for unlike injectors, over the range of impingement angles from 40°to 80°, mixing homogeneity increases as the impingement angle decreases. This assessment cannot be verified because we are using a like injector configuration with two water jets. According to our results, mixing uniformity increases as the impingement angle increases. Most unlike impinging injectors are designed at angles of 60°, which is seen for our case to have a good mixing uniformity and will be put into test in the experimental study.

5 Experimental Analysis

The advantages of numerical modelling compared with experimental studies (e.g. reduced cost, flexibility, amount of detail etc.) are well known. However, theoretical studies where experimental validation is also presented provide an important added value to numerical investigations. For this reason, this chapter will be dedicated to validate the results obtained on the numerical analysis with a real life experiment that will test the mixing characteristics of an injector system design.

The mixing characteristics of liquid-rocket injectors are usually tested with nonreactive, immiscible liquids having physical properties similar to those of the actual propellants. The combination of injector parameters (e.g., impingement angle, impingement distance, nozzle type, etc.) which maximizes the mixing of the water jets is said to be "optimum" with respect to mixing.[43]

The main goal of this laboratory experiment is to design and effectively test a water-based doublet injector system. The impingement angle and type of nozzle will be the most important variables to test in order to get the best configuration. The experiment will have visual results, as a camera will be used to collect photos of the spray from various angles in order to determine the level of mixture in the spray.

The experiment will be divided in two separate tests. First, a little test will be conducted to analyse the spray properties of each stream nozzle in order to select the most adequate for the second test. The second test will involve experimenting with various angle combinations of an impinging doublet prototype using the previously chosen nozzles.

5.1 Design and Construction of the Prototype

The design of the prototype is based on a simple principle used in hydraulic presses and other quotidian objects such as pipettes called the Pascal's principle. This states that for any fluid at rest in a closed container, a pressure change in one part is transmitted without loss to every portion of the fluid and to the walls of the container. The principle was first enunciated by the French scientist Blaise Pascal.[28]

In our case, the force will be applied by an external air-pressurized tank to a piston, which will be afterwards transmitted to a neighbour water chamber and finally ejected through a nozzle. The exact geometry and the components of the design can be seen in Figure 55.

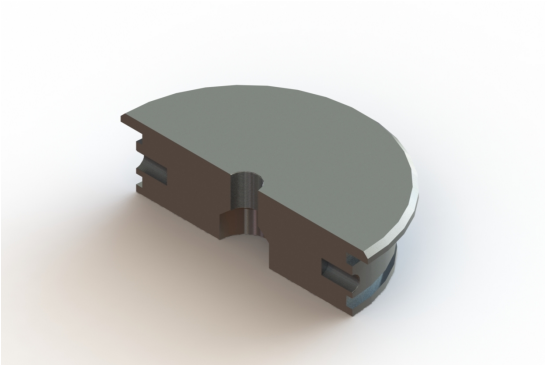
Figure 55 Test prototype with different components representation, Source: Own

All the components are made out of stainless steel and they need to be perfectly designed and constructed to ensure that they fit correctly inside the tube and that no fluid leaks occur.

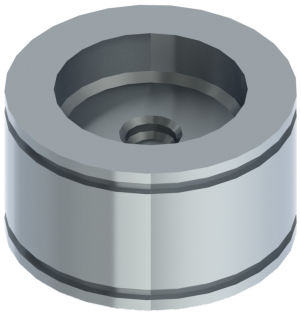
The tube measures 120 millimetres in length and 80 millimetres in diameter. A stainless steel cover will be placed over each open end and secured in place by eight screws. A 40 mm long piston will be placed on the inside of the enclosed tube and will be responsible for transferring air pressure to the water as well as isolating the two chambers to prevent leakage. A representation of these components can be seen in Figure 56.

In order to avoid that the injection system gets jammed during operation (RQ-4), lubrication and tightness are key elements for a successful performance of the prototype. This will be ensured by using rubber O-Rings at each prepared joint and lubricating grease to ensure tightness with the tube.

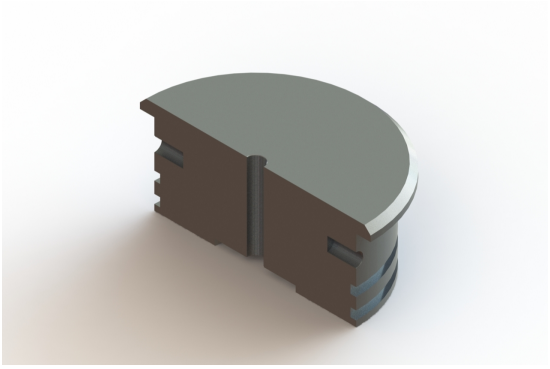
(a) Stainless steel body tube



(b) Stream nozzle water cover



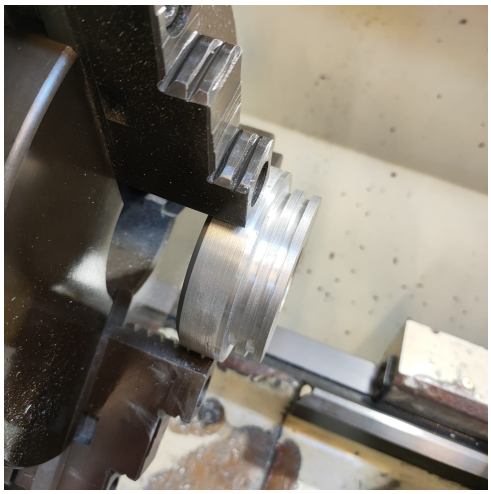
(c) Piston



(d) Pressurized air cover

Figure 56 *SolidWorks* renderings of the components, Source: Own

In order to keep reduced costs (RQ-6), machining considerations have been taken into account. With the cooperation of the project's director, Jaume Solé Bosquet, all of the components were fabricated in a workshop plant in SENER–NTE Cerdanyola del Vallès, Barcelona. The following images show the crafting process with the lathe machine.

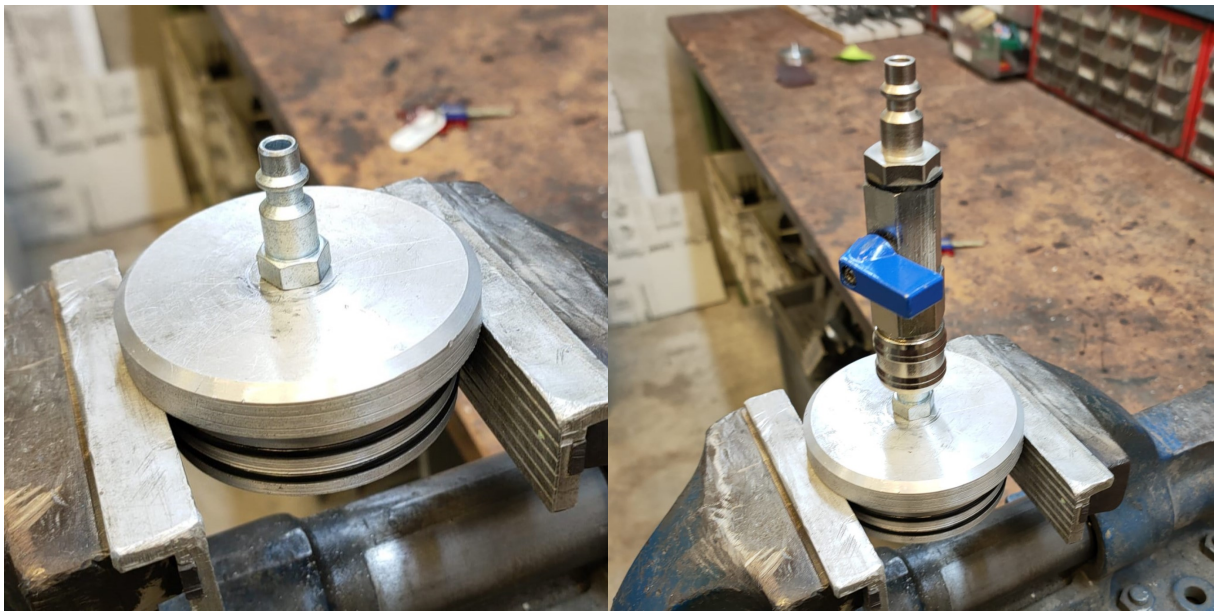


(a) Turning of the nozzle water cover

(b) Turning of the piston

Figure 57 Machining process of the different components, Source: Own

Once both covers and the piston are crafted, the tightening rubber O-Rings can be placed over the joints and the different stainless steel devices can be threaded in place. These devices are the air compressor hose quick connector BSP (British Standard Pipe) and the stream nozzle to be used at the moment.



(a) Air compressor hose quick connector **(b)** Shut-off valve for air compressor

(c) Stream nozzle installed

Figure 58 Different components threaded on their respective steel cover, Source: Own

Placing all the elements in place and drilling the fixing element holes in the tube is the final step in the creation of the prototype required to conduct the experiments.

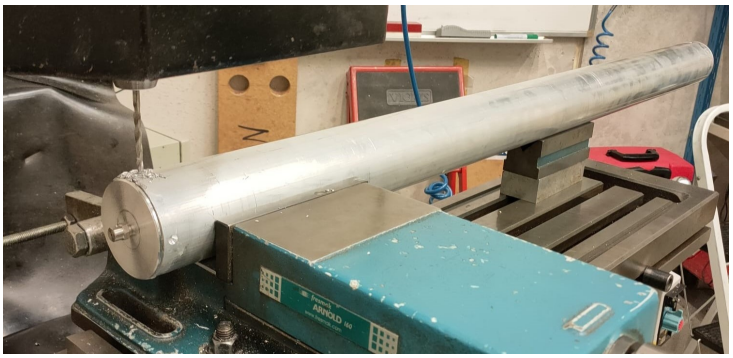


Figure 59 Drilling the holes on the tube, Source: Own

5.2 Stream Nozzle Study Experiment

As previously stated, the goal of this experiment is to test the different acquired nozzles in order to study the different spray patterns and select the most suitable one to recreate the doublet injector simulation. The procedure will be as follows. Each type of nozzle will be threaded onto the plate with the intention of testing it by allowing the water flow through it. In order to make this happen, the piston must be pushed backwards, and the water manually poured into the chamber. The air compressor hose quick connector can then be attached to the pressurised air line (Figure 60) and once the shut-off valve is switched on, air will begin to flow pushing the piston and therefore creating a water flow through the nozzle.

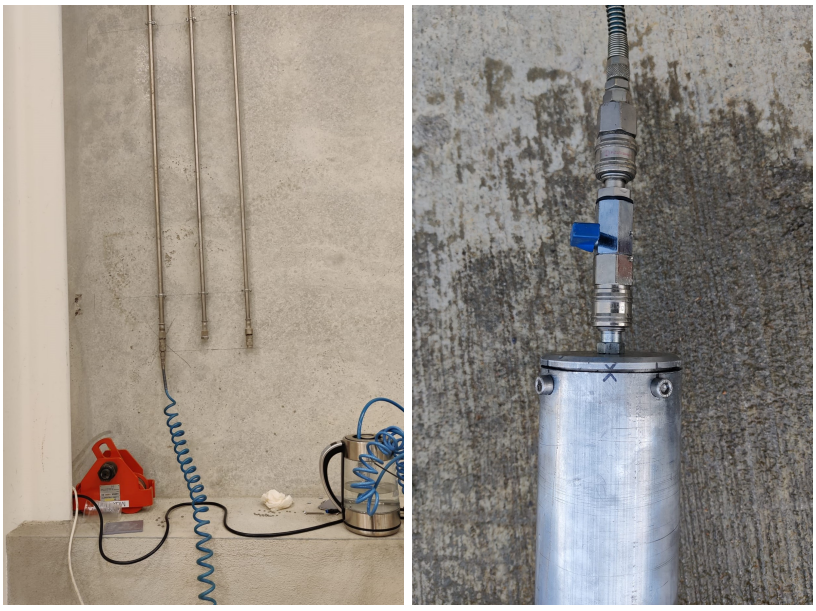
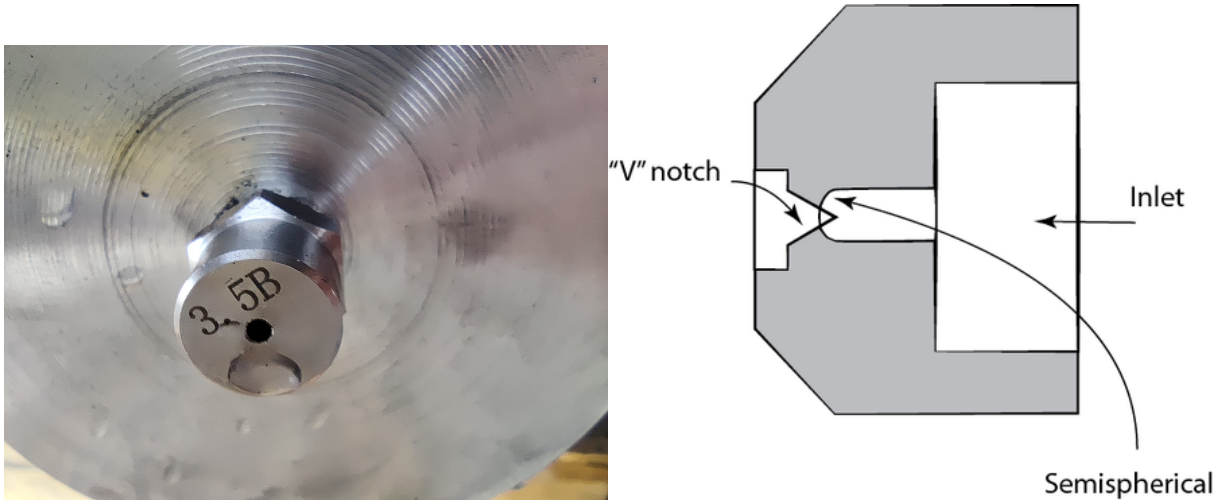


Figure 60 Pressurized air line connection, Source: Own

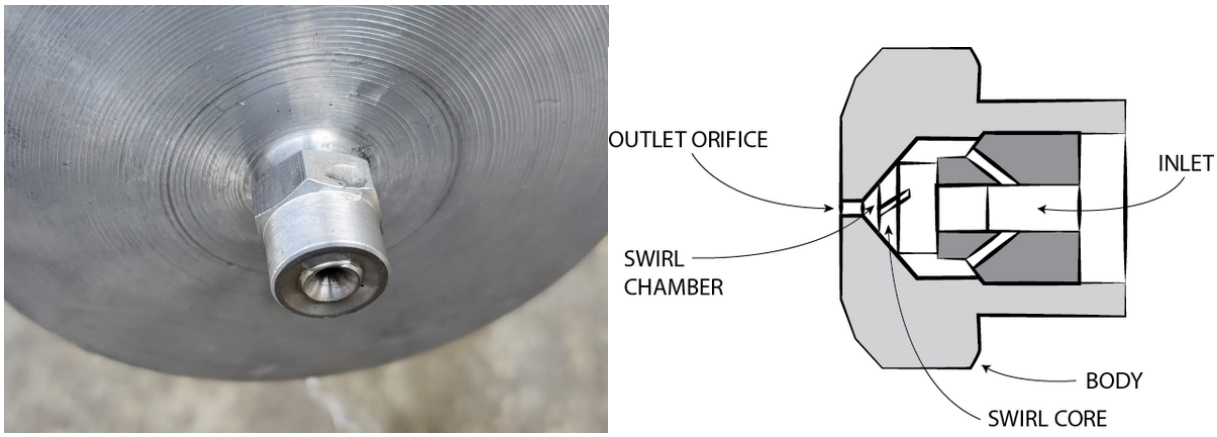
The following figures show the different types of nozzles used for this experiment.



(a) Stainless steel solid cone nozzle **(b)** Interior nozzle design[30]

Figure 61 Solid cone spray Nozzle, Source: Own

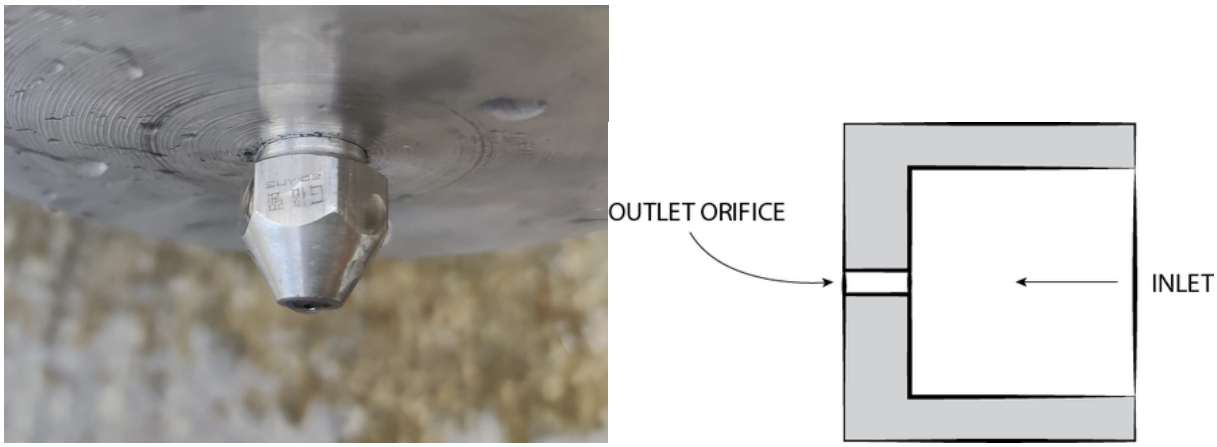
Solid cone spray nozzles (Figure 61) produce a solid cone-shaped spray pattern with a round or square impact area. They usually produce medium to large sized droplets.[27]



(a) Stainless steel hollow cone nozzle **(b)** Interior nozzle design[30]

Figure 62 Hollow swirl cone spray Nozzle, Source: Own

The hollow cone nozzles (Figure 62) produce spray patterns which are characterised by the high concentration of droplets around the edge of the spray cone while little or no fluid concentration in the middle of the cone. Hollow cone nozzles have the advantage of producing the smallest drop size, giving the spray a larger surface area and allowing for faster heat transfer.[27]



(a) Stainless steel straight jet nozzle **(b)** Interior nozzle design[30]

Figure 63 Plain-orifice straight jet nozzle , Source: Own

Straight jet or solid stream nozzle (Figure 63) is the simplest of all nozzles, being little more than a circular orifice at the end of a funnel. Solid stream nozzles give the highest impact of any spray pattern as the whole momentum of the liquid is concentrated into a small region. In this type of nozzles, the liquid is not atomised, for this reason droplet size is irrelevant.[27]

Following, the different nozzles will be put under test conditions so that their spray characteristics can be studied. Figure 64 shows the final setup used for this experiment. The user (myself) can be seen holding the tube and activating the shut off valve to enable the flow of pressurized air through the line.

(a) Shut off valve disabled **(b)** Shut off valve enabled

Figure 64 Prototype testing operation, Source: Own

5.2.1 Results and Discussion

Following the order from the previous pictures regarding the different types of nozzles, the first nozzle that has been put into test is the solid cone nozzle. The resulting spray of this nozzle can be seen in Figure 65. As expected, the spray has adopted a very well defined cone pattern, with very little, if any, water dispersed away from the cone solid shape. The solid cone angle was around 60°, and the resulting spray was extremely atomized, only reaching a distance of around one metre before becoming a nebulized spray. It was easy to detect that the interior of the cone was uniform and had a homogeneous distribution when touched with the hand, which is a particular property of this kind of nozzles.

Figure 65 Solid cone nozzle spray pattern, Source: Own

The second one is the hollow cone nozzle (see Figure 66). The resulting spray of this nozzle was similar to the solid cone nozzle but a wider angle was noticeable. In this case, the spray was dispersed at an approximate angle of 80°and the cone boundaries were not as clear as in the first case. Moreover, the spray did not reach as far as the one from the solid cone creating a sort of cloud less than one meter away from the nozzle. This indicates that the water was quickly atomized, with very small droplets. When touched with the hand it could be sensed that the cone was not uniform on the inside and that the majority of water droplets were dispersed on the outer layer of the cone, which is in line with the expectations for this type of nozzles. During the testing of this nozzle, a few leakage issues occurred; the screws were not properly tightened, and the nozzle cover was unable to seal the high pressure from the liquid, as shown in Figure 67.

Figure 66 Hollow cone nozzle spray pattern, Source: Own

Figure 67 Prototype cover sealing malfunction, Source: Own

The solid jet nozzle (see Figure 68) is the final of the three nozzles that have been tested. The resulting spray of this nozzle is nothing compared to the other two; it produces a solid jet ejected at a high velocity with no visible atomization or droplets. The water is ejected with such a force and momentum that it reaches a distance of approximately 4/5 metres before striking the ground, and when touched, it feels like touching solid wire. This kind of jet spray is very promising since it can be employed to recreate the injector design used in the simulation. Two of these nozzle can be positioned in collision trajectory at a certain angle to create an impinging injector.

Figure 68 Solid jet nozzle spray pattern, Source: Own

In conclusion, the selected nozzle to create an impinging injector is the solid jet nozzle as the other two are inadequate for the type of application we are seeking for. Their spray characteristics, on the other hand, are really interesting and could be tested to develop an injector because the spray is already atomized and it could have a promising mixing efficiency.

The last experiment could not be completed due to a lack of resources and time, hence it is a task pending to be conducted (see chapter 6.3.3).

6 Project Review

6.1 Safety Analysis

Despite the fact that this project was conducted using non-reactive fluids (water), it is essential to take into consideration the physical hazards when handling propellants and when using the turning machines. Rocket injectors for liquid fuel rocket engines are a complex and nuanced matter and certain elementary safety precautions must be observed when testing them. the following general safety considerations should be kept in mind:

When operating a lathe machine: [22]

(a) Wear appropriate clothes, certified safety glasses, and anti-cut gloves as objects may fly off the work and can be very dangerous.

(b) Keep the floor free from obstructions, or slip hazards and free of oil and grease.

(c) Shut off the power supply to the motor before mounting or removing accessories and before taking measurements of any kind.

(d) Use a vacuum, or brush to remove cuttings only after the lathe has stopped moving.

(e) Secure and clamp the piece being worked.

(f) Use a barrier guard when operating the lathe in semi-automatic or automatic mode.

(g) Remove all tools, measuring instruments, and other objects from the saddle or lathe bed before starting the machine.

(h) Make sure all lathe cutting tools are sharp.

Figure 69 Lathe machine guidelines ANSI (wall sign), Source: [18]

When operating with small liquid rocket engines: [1]

(a) The operator should be protected by a suitable barricade located some distance (at least 6 meters) from the test unit and the control of valves during engine ignition operation should be by remote means.

(b) A large chemical fire extinguisher or a supply of water should always be on hand.

(c) The operator should not view the test unit directly, but should use a mirror arrangement or use a thick layer of reinforced safety glass attached to the operator's barricade.

(d) It should be taken into account that separating fuel and oxidizer storage minimises the risk of fire and explosion

(e) Warning signals should be given prior to tests to notify personnel that the area is hazardous. A test must never be conducted until the operator has assured himself that all personnel are behind safety barricades or otherwise protected.

(f) Personnel should be permitted to work in the test area only if fuel and oxidizer are separated and not pressurized.

(g) Personnel handling propellants should wear safety equipment such as gloves, face shields, or rubber aprons.

(h) No smoking is ever permitted anywhere near a test area when propellants are present.

6.2 Environmental Impact

Despite the fact that the project was not planned as a commercial product but rather as a potential low-cost design, it had an environmental impact. On the following pages, the carbon footprint will be tracked. The test had no environmental impact because it was conducted using non-contaminating elements. However, the environmental consequences of the design, transportation, and production of the project and prototype must be considered. First of all, the Greenhouse Gases will be considered.

The production of CO_2 can be understood by the hours spent using the computer while writing the report, simulating and designing. The working setup is composed of a laptop and a second screen. Checking the specifications online a nominal power consumption of 150W for the laptop and 15W for the screen can be assumed, which combined have a power consumption of 165W, i.e. 0.165kW. Assuming that at least 500 hours were spent on the project's development, the total amount of energy expended is:

$$0.165kW \cdot 500h = 82.5kW\,h \tag{97}$$

According to reference [21] the greenhouse gases (GHG) generated in Spain due to electrical

consumption can be calculated by simply typing the energy dispense in their website:

$$20.60kg \text{ of } CO_2 \tag{98}$$

This amount can be added to the project's second source of $CO2$ emissions, transportation and logistics. According to the same source, the amount of $CO2$ can be determined by entering the kilometres driven, the type of automobile, and the number of passengers. Estimating a total travelled distance of 70km by car:

$$13.40kg \text{ of } CO_2 \tag{99}$$

Next, the GHG produced by the lathe machine must be taken into account. The machine has an average power consumption of 4kW. The time spent mechanising the prototype is around 90 min, hence, the power consumption can be calculated as:

$$4kW \cdot 1.5h = 6kWh \tag{100}$$

This value can be once again introduced to reference [21] and the amount of CO_2 generated due to the lathe is:

$$1.50kf \text{ of } CO_2 \tag{101}$$

The total amount of CO_2 produced during the project sums up to **35.5kg**

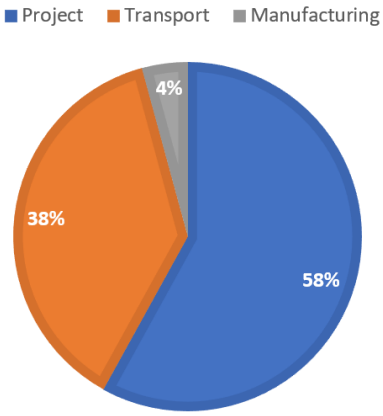


Figure 70 Project emissions distribution, Source: Own

6.3 Conclusions - Improvements - Future

6.3.1 Conclusions

The project's objective was to design, simulate and test a low-cost doublet rocket motor injector. The prototype tests were intended to validate the simulated results and propose a final solution to the initial problem.

Regarding the experimental analysis the project can be called a success, despite the fact that the final test could not be performed. It has been proven that the prototype is feasible and durable, thus it works properly and it performs its designated task, which is to create a pressurized flow through the nozzle.

From the simulation perspective, it has been a hard task to achieve. Nonetheless, it can be called a success, as the results are really visual and indicative that mixing uniformity can be achieved by using a pair of nozzles placed at an angle higher than 60°. Once the `doubletInjector` case file had been successfully programmed it was very straight forward to repeat the simulation with different angular parameters and the computing time ended up being a bit less than one hour.

The other perspective is that the prototype tube helped to obtain more empirical and valuable results for the study. The design was meant to be as simple as possible to reduce the machining time and complete both experiments. Unfortunately, the workshop plant in SENER-NTE has been extremely busy over the past months, making it really difficult to make progress with the prototype.

From a personal point of view, the project is really extensive and well-documented, encompassing both theoretical and experimental approaches. From studying all the maths and physics to generate a CFD simulation along with designing a 2D Lagrangian-Eulerian multiphase simulation, going through postprocessing all the results and even externally programming a python code in order to create a bicoloured mixture representation that can be seen in the GitHub link (Appendix A). Following with an experimental approach, machining all the components and successfully assembling and testing them.

6.3.2 Improvements

To whom the design and development of the simulations and testing of the test model may concern, the following improvements have been outlined:

- Regarding the residuals graph of the numerical simulation of the impinging injector it can be seen that the simulation is slightly converged. Typically this happens when the mesh

is not optimal and far too coarse. A possible way through this problem may be reducing the size of the cells, i.e. refining the mesh. Other possibilities may be that the turbulent dissipating rate (ε) parameter is extremely high and the simulation could be run with a laminar model. However, a laminar simulation model has already been tested and the residuals' values do not improve. It must be taken into account that all the parameters have been depicted through trial and error in order to get coherent results. For this reason, other configurations, as well as other Lagrangian solvers such as the `sprayFoam`, might result in more converged results.

- With respect to the visualization of the bicoloured spray in chapter 4.9 the ParaView post-processing tool was not capable of tracing down the inlet from which each particle came from, or at least, it was not capable of assigning a different colour to each inlet flow of particles. For this reason, an external code has been programmed tracing down the particle's position while they entered the domain and assigning a colour (blue or red) depending on their y axis value. This code is applied to an array of coordinates corresponding to all the different particles' position in each time step. However, when a particle exits the domain it gets deleted and the array changes its size. Consequently, all the colour values assigned to each row lose track of their particle, resulting on a chaotic coloured spray. In order to tackle this problem, the domain has been enlarged, at the expense of having to simulate again all the files, and it was possible to track the particles up to 0.009 seconds before they started exiting the domain.

- With regard on the experimental analysis, multiple leakage problems were detected when enabling the shut off valve. The design of all the components needed to be correctly measured in order to fit inside the tube. We decided that a smaller diameter would be sufficient to seal the liquids while also making it easier to put in and take out the covers when necessary. This did not go as planned since the cover's diameter was slightly inferior to optimum, allowing water to leak out.

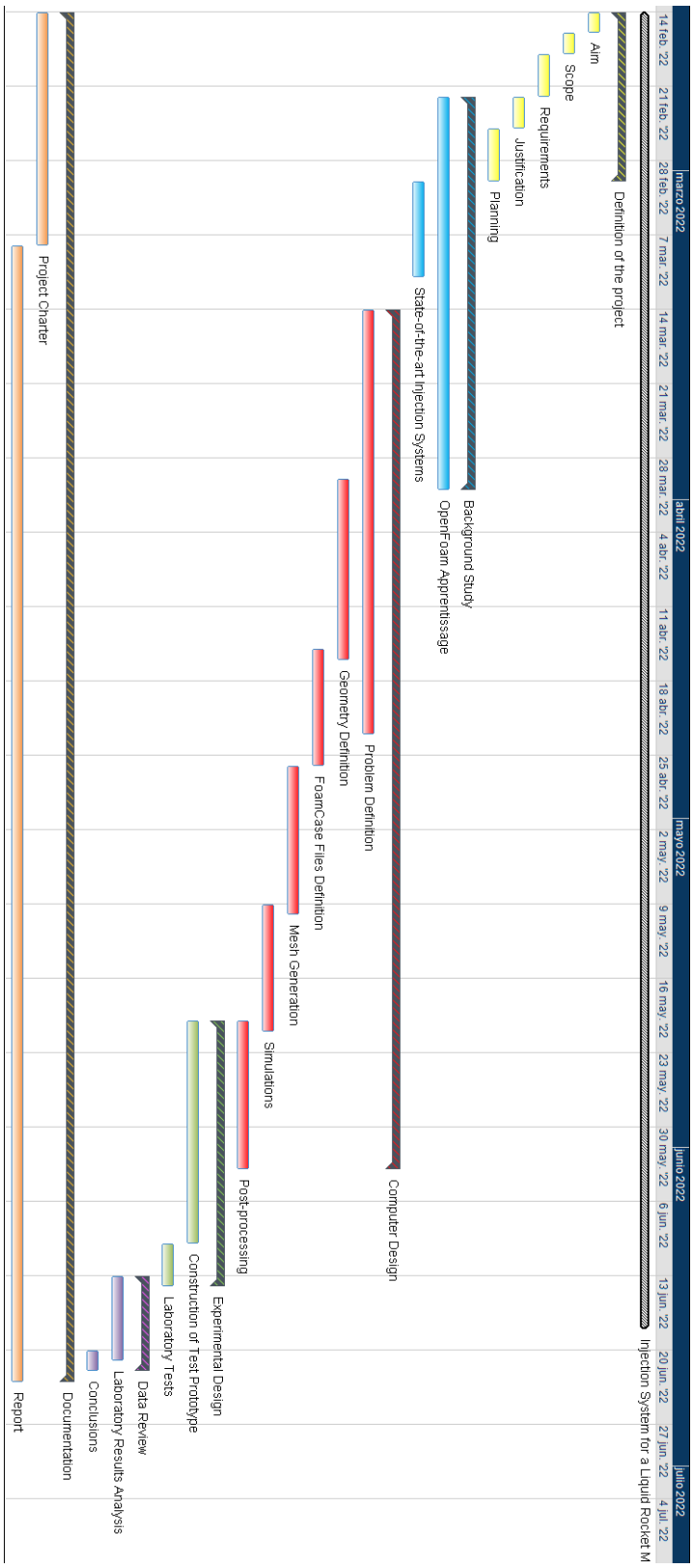
6.3.3 Future Development

The future work of this project is to design and test the second experiment. This consists in creating a new design capable of feeding two solid jet nozzles at the same time, angled at a collision trajectory. This redesign of the prototype could be created by cutting in half the last testing tube in order to create two separate tubes which could be filled up with water and pressurized by two different quick air lines.

Implementing this second experiment along with the improvements commented before would be very valuable for the project.

6.3.4 Project Planning

This is the schedule that this project has followed (from top to bottom):



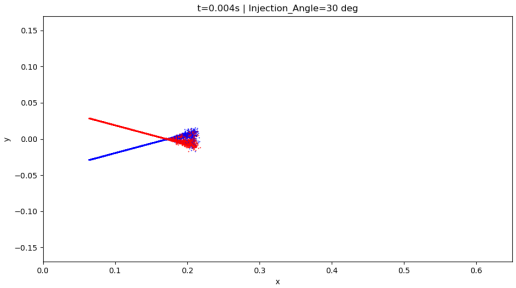
A Appendix I: OpenFOAM case (`doubletInjector`)

- GitHub `doubletInjector` case link:

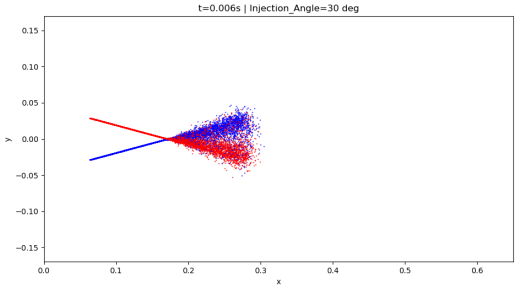
  github.com/adriaGonzalezCano/TFG—Adria-Gonzalez

B Appendix II: Water Droplets Spray Bicolour Mixture

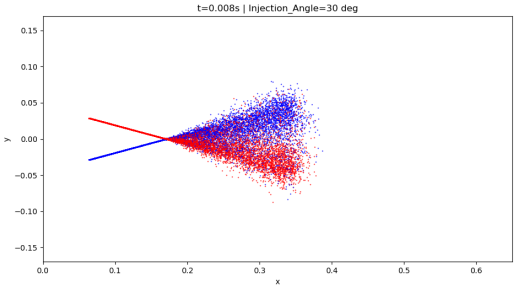
B.1 Angle: 30°



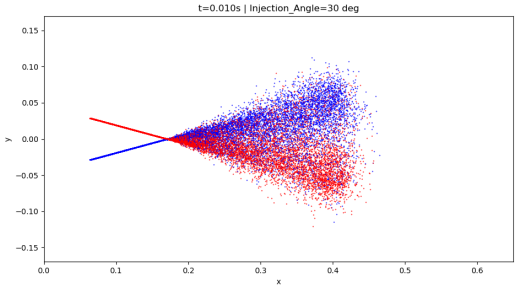
(a) Time = 0.003 s



(b) Time = 0.005 s

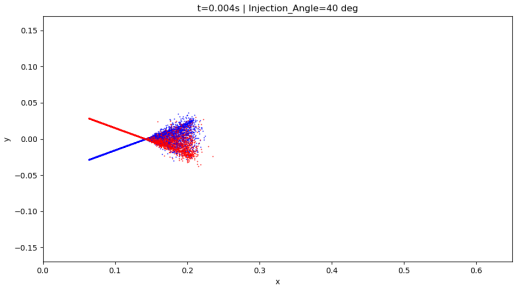


(c) Time = 0.007 s

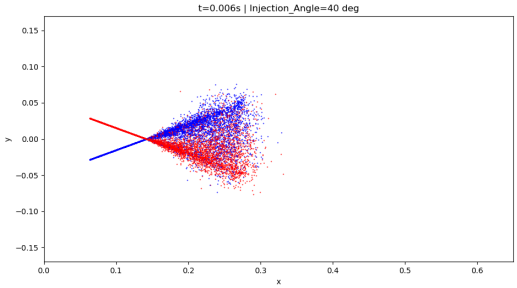


(d) Time = 0.009 s

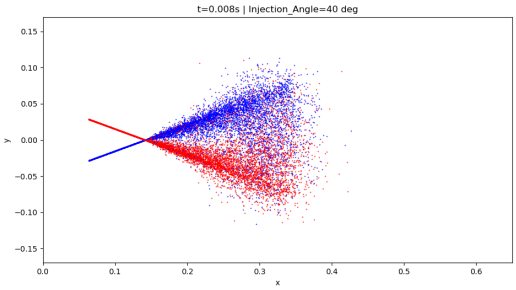
B.2 Angle: 40°



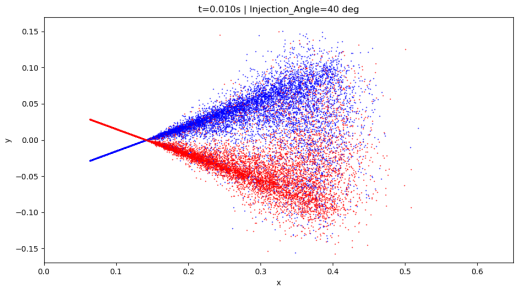
(a) Time = 0.003 s



(b) Time = 0.005 s

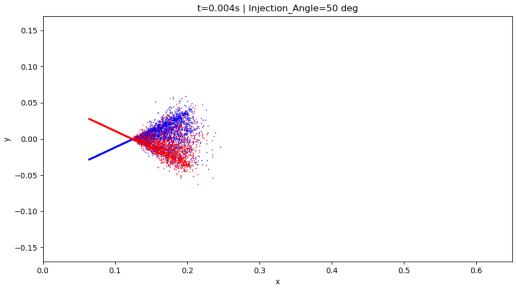


(c) Time = 0.007 s

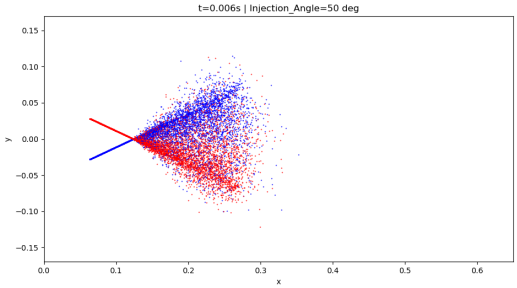


(d) Time = 0.009 s

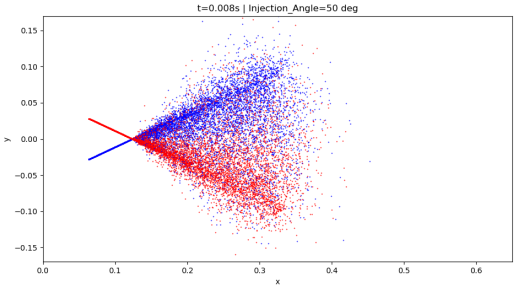
B.3 Angle: 50°



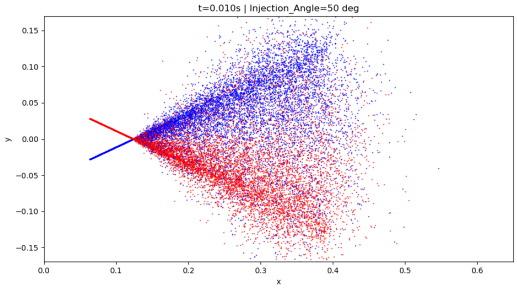
(a) Time = 0.003 s



(b) Time = 0.005 s

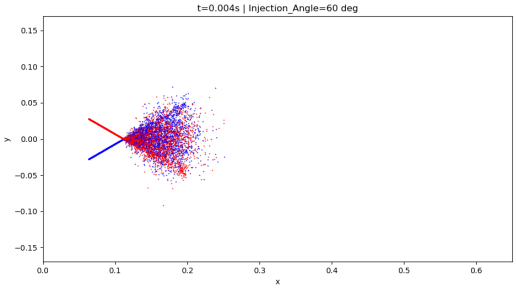


(c) Time = 0.007 s

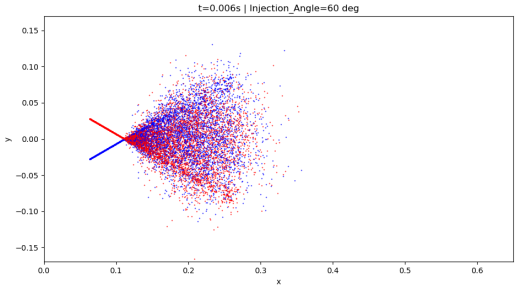


(d) Time = 0.009 s

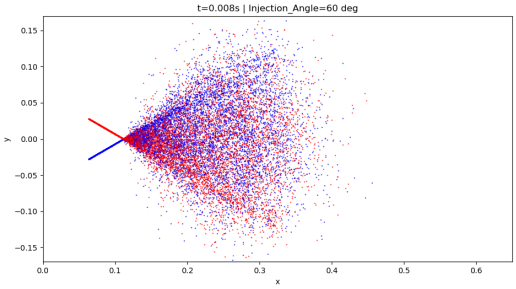
B.4 Angle: 60°



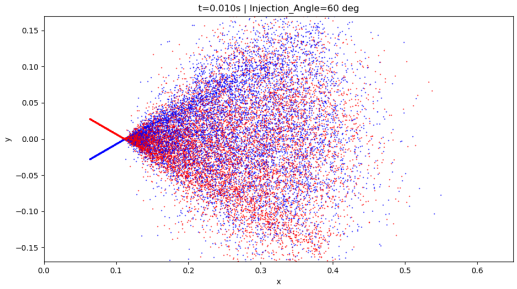
(a) Time = 0.003 s



(b) Time = 0.005 s

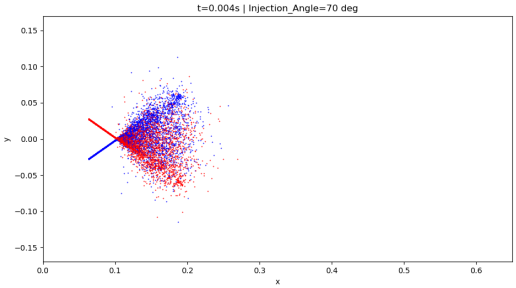


(c) Time = 0.007 s

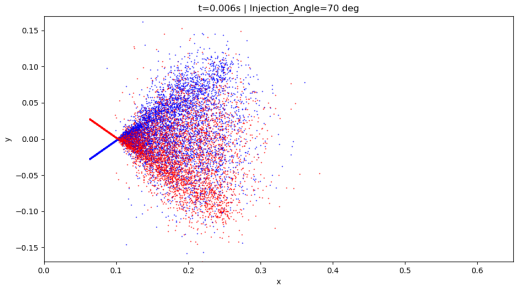


(d) Time = 0.009 s

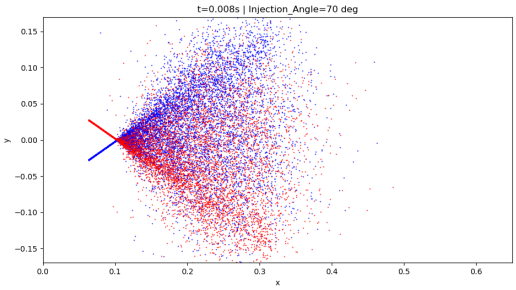
B.5 Angle: 70°



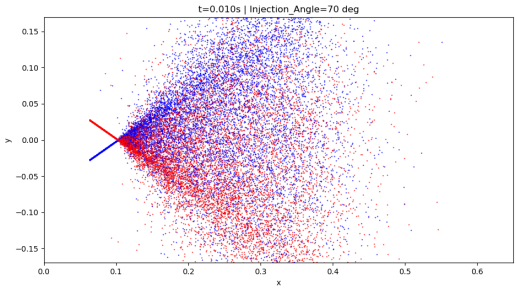
(a) Time = 0.003 s



(b) Time = 0.005 s

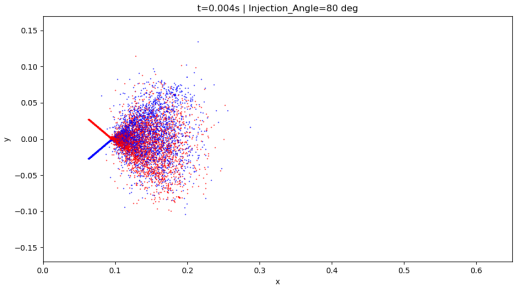


(c) Time = 0.007 s

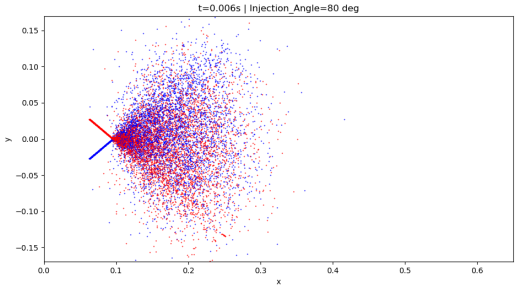


(d) Time = 0.009 s

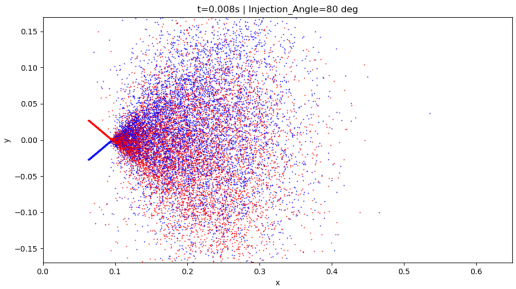
B.6 Angle: 80°



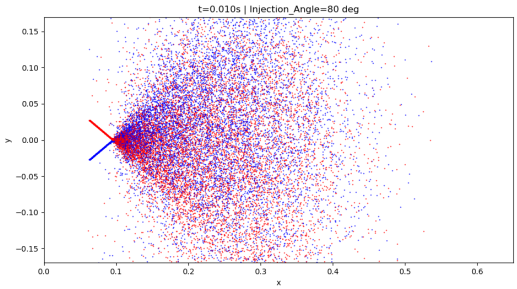
(a) Time = 0.003 s



(b) Time = 0.005 s

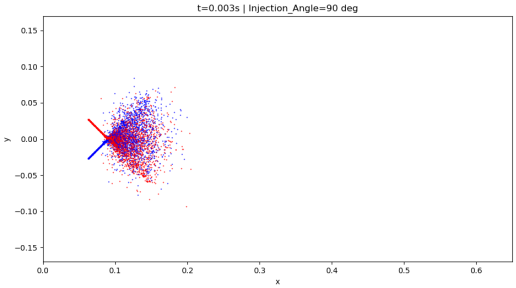


(c) Time = 0.007 s

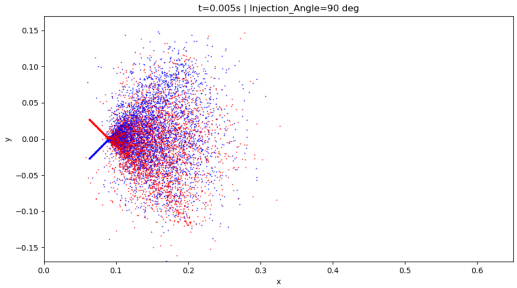


(d) Time = 0.009 s

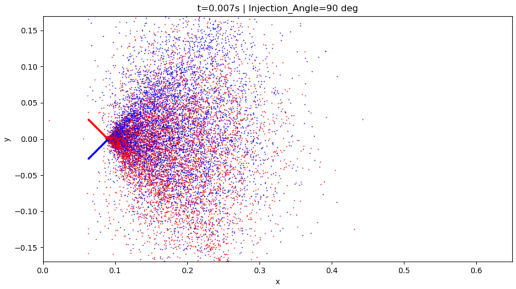
B.7 Angle: 90°



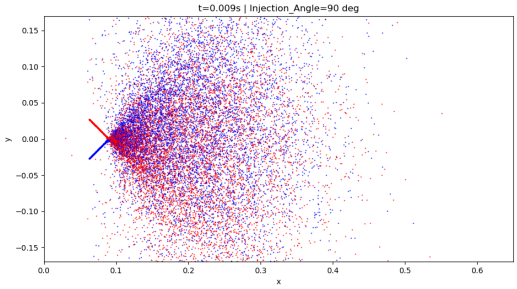
(a) Time = 0.003 s



(b) Time = 0.005 s



(c) Time = 0.007 s



(d) Time = 0.009 s

References

[1] Leroy J. Krzycki. *How to design, build and test small liquid-fuel rocket engines*. Rocketlab, 1967. URL: https://spacha.github.io/How-to-Rocket/#injectors.

[2] Howard W. Douglass et al. "Liquid Rocket Engine Injectors". In: (Mar. 1976).

[3] Wangda Zuo. "Introduction of Computational Fluid Dynamics". In: (2005).

[4] Weeratunge Malalasekera Henk Kaarle Versteeg. "An Introduction to Computational Fluid Dynamics: The Finite Volume Method". In: (2007). URL: https://books.google.es/books?id=RvBZ-UMpGzIC&printsec=frontcover&hl=ca#v=onepage&q&f=false.

[5] N. Ashgriz, W. Brocklehurst, and D. Talley. "Mixing Mechanisms in a Pair of Impinging Jets". In: *https://doi.org/10.2514/2.5803* 17 (3 May 2012), pp. 736–749. ISSN: 07484658. DOI: 10.2514/2.5803. URL: https://arc.aiaa.org/doi/10.2514/2.5803.

[6] Vincent Notaro. "Mixing Analysis of Like Doublet Injectors in High Pressure Environments for Gelled Propellant Simulants". In: (2013).

[7] Robert Kasper. "Particle Simulation with OpenFOAM Introduction, Fundamentals and Applications". In: (2017).

[8] M. Hahn D. W. Savin V. A. Kuznetsov P. A. Trubaev. "Convective heat transfer in the near-wall turbulent gas stratum". In: *Materials Science and Engineering* (2019). DOI: 10.1088/1757-899X/552/1/012005.

[9] Milad Mottaghi, Hooyar Attar, and Mehrdad Torabzadegan. "Assess proper drag coefficient models predicting fluidized beds of CLC reactor by utilizing computational fluid dynamic simulation". In: (2020).

[10] Sadanandam Anupoju. "Dimensionless Numbers and Their Importance in Fluid Mechanics". In: (2021). URL: https://theconstructor.org/fluid-mechanics/dimensionless-numbers-importance-fluid-mechanics/34265/.

[11] R. Sijs et al. "Drop size measurement techniques for sprays: Comparison of image analysis, phase Doppler particle analysis, and laser diffraction". In: *AIP Advances* 11 (1 Jan. 2021). ISSN: 21583226. DOI: 10.1063/5.0018667.

[12] OpenCFD Ltd 2022. *Open FOAM User Guide*. URL: https://www.openfoam.com/documentation/user-guide (visited on 05/24/2022).

[13] *3 Criteria for Assessing CFD Convergence | Engineering.com*. URL: https://www.engineering.com/story/3-criteria-for-assessing-cfd-convergence (visited on 06/02/2022).

[14] *Application of Reynolds Number | nuclear-power.com*. URL: https://www.nuclear-power.com/nuclear-engineering/fluid-dynamics/reynolds-number/application-of-reynolds-number/ (visited on 05/17/2022).

[15] Cadence CFD. *Methods for Grid Generation in CFD Simulations | System Analysis Blog*. URL: https://resources.system-analysis.cadence.com/blog/msa2021-methods-for-grid-generation-in-cfd-simulations (visited on 05/17/2022).

[16] *Conservation Laws*. URL: http://hyperphysics.phy-astr.gsu.edu/hbase/conser.html (visited on 05/2022).

[17] NASA Cristopher Rumsey. *Wilcox k-omega Model*. URL: https://turbmodels.larc.nasa.gov/wilcox.html (visited on 05/20/2022).

[18] *Danger: Lathe Machine Guidelines ANSI - Wall Sign | 5S Today*. URL: https://www.5stoday.com/danger-milling-machine-guidelines-ansi-wall-sign/ (visited on 06/12/2022).

[19] *Darcy Friction Factor*. URL: https://www.nuclear-power.com/nuclear-engineering/fluid-dynamics/major-head-loss-friction-loss/darcy-friction-factor-2/ (visited on 05/17/2022).

[20] *Fluid Mechanics 101 - YouTube*. URL: https://www.youtube.com/channel/UCcqQi9LT0ETkRoUu8eYaEkg

[21] *Home - Calculo GEI CeroCO2*. URL: https://www.ceroco2.org/calculadoras/home (visited on 06/18/2022).

[22] *Metalworking Machines - Lathes*. URL: https://www.ccohs.ca/oshanswers/safety_haz/metalworking/lathes.html (visited on 06/12/2022).

[23] *Multi-phase flow simulations in OpenFOAM*. URL: http://www.cfdyna.com/Home/of_multiPhase.html (visited on 05/31/2022).

[24] *Navier-Stokes Equations | Introduction to CFD*. URL: https://cfd.blogs.upv.es/introduction/navier-stokes-equations/ (visited on 05/2022).

[25] *Navier–Stokes Equation | Clay Mathematics Institute*. URL: https://www.claymath.org/millennium-problems/navie-stokes-equation (visited on 05/2022).

[26] *Normal Distribution Applet/Calculator*. URL: https://homepage.divms.uiowa.edu/~mbognar/applets/normal.html (visited on 05/31/2022).

[27] *Nozzles for all industries and applications*. URL: https://www.spray-nozzle.co.uk/home (visited on 06/10/2022).

[28] *Pascal's principle | Definition, Example, Facts | Britannica*. URL: https://www.britannica.com/science/Pascals-principle (visited on 06/10/2022).

[29] Rashid. *Types of fluid flow – Fluid Mechanics*. URL: http://www.mechanicalwalkins.com/types-of-fluid-flow-fluid-mechanics/ (visited on 05/15/2022).

[30] *Spray nozzle - Wikipedia*. URL: https://en.wikipedia.org/wiki/Spray_nozzle (visited on 06/10/2022).

[31] *src/finiteVolume/fields/fvPatchFields - openfoam · GitLab*. URL: https://develop.openfoam.com/Development/openfoam/-/tree/master/src/finiteVolume/fields/fvPatchFields (visited on 05/24/2022).

[32] *Swirlers – Copenhagen Suborbitals*. URL: https://copenhagensuborbitals.com/swirlers/ (visited on 06/05/2022).

[33] *Turbulence free-stream boundary conditions – CFD-Wiki, the free CFD reference*. URL: https://www.cfd-online.com/Wiki/Turbulence_free-stream_boundary_conditions (visited on 05/31/2022).

[34] *Viscosity of Air, Dynamic and Kinematic*. URL: https://www.engineersedge.com/physics/viscosity_of_air_dynamic_and_kinematic_14483.htm (visited on 05/31/2022).

[35] Inc ANSYS. *ANSYS Fluent User's Guide*. November 2013. URL: https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/main_pre.htm.

[36] Hrvoje Jasak. *Error analysis and estimation for the finite volume method with applications to fluid flows*. Imperial College London (University of London), January 1996. URL: http://hdl.handle.net/10044/1/8335.

[37] John D. Anderson Jr. *[Computational Fluid Dynamics - The Basics with Applications*. February 2022. URL: https://www.askbooks.net/2022/02/pdf-computational-fluid-dynamics-basics.html.

[38] R M Knight and W H Nurick. "Interim Report, Correlation of Spray Dropsize Distribution and Injector Variables". In: (September 1969).

[39] Johan C. Kok. “Resolving the dependence on freestream values for the k-omega turbulence model”. In: *AIAA journal* 38 (7 July 2000), pp. 1292–1295. DOI: 10.2514/2.1101.

[40] Jet Propulsion Lab. “The Liquid Phase Mixing of a Pair of Impinging Streams”. In: (August 1953). URL: https://arc.aiaa.org/doi/pdf/10.2514/3.28966.

[41] Hamidreza Norouzi. “OpenFOAM Utilities Tracing Particles to Find RTD Compatible with”. In: (June 2020).

[42] Guozhu Liang Qiang Wei. “Coupled Lagrangian impingement spray model for doublet impinging injectors under liquid rocket engine operating conditions”. In: (June 2017). URL: https://www.sciencedirect.com/science/article/pii/S1000936117301504.

[43] Robert W. Riebling. “Criteria for Optimum Propellant Mixing in Impinging-Jet Injection Elements”. In: (June 1967). URL: https://arc.aiaa.org/doi/pdf/10.2514/3.28966.

[44] Ideen Sadrehaghighi. *Unstructured Meshing for CFD*. May 2022. URL: https://www.researchgate.net/publication/339285304_Unstructured_Meshing_for_CFD.

[45] Brian Launder B.I. Sharma. “Application of the energy-dissipation model of flow near a spinning disc”. In: (January 1974). URL: https://www.researchgate.net/publication/284652999_Application_of_the_energy-dissipation_model_of_flow_near_a_spinning_disc.