U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Department of Mechanical Engineering

Morphing Autonomous Underwater Vehicle - Hydrodynamic Analysis

Author:
Ricardo Cordeiro - up201808938

Supervisors:
Prof. Dr.ª Catarina Castro
Dr. Nelson Gonçalves

Dissertation submitted to the Faculty of Engineering of University of Porto in partial satisfaction of the requirements for the degree of Master in Mechanical Engineering.

October 26, 2020

Acknowledgements

After this long journey I would like to thank my supervisor from the Faculdade de Engenharia da Universidade do Porto (FEUP), Prof. Dr.ª Catarina Castro for all the advice, dedication and tireless effort that gave me and my dissertation through all it's development.

Next, I would like to thank Dr. Nelson Gonçalves, my supervisor from the Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial (INEGI), for all the support and hours of dedication to my dissertation. He made the greatest work mentoring my path, specially transmitting is knowledge and expertise in an easy and comprehensive way, but also as a person that was always ready to help when I needed the most.

Also I would like to give a special thank to Prof. Dr. Carlos António, from Faculdade de Engenharia da Universidade do Porto (FEUP), for all his support on the Artificial Neural Network and optimization developments.

I would like to thank all my friends and colleagues that have been present in my academic journey, they really have proportioned me moments that I will never forget.

I would like to give my kind thank to my girlfriend, Adriana, for all the incessant and limitless love and kindness that she gives me.

Lastly I would like to thank my family, my sister Cátia and my parents Maria da Conceição and Augusto for their love, support and afection. They are my foundation.

(This page was intentionally left blank)

Abstract

This dissertation concerns the hydrodynamics of an Autonomous Underwater Vehicle (AUV) capable of changing it's shape in order to obtain the best performance in different situations. The demand for high-performance AUVs is growing in the field of ocean engineering due to increasing activities in ocean exploration and research. New generations of AUVs are expected to operate in harsh and complex ocean environments. We propose a hybrid design of an Autonomous Underwater Vehicle based on a Bio-inspired form.

The study was performed by creating a bi-dimensional shape using seven input variables. This body was object of a hydrodynamic study in order to get two output variables. The input variables are geometrical, these are respective to the body dimensions: Length (L), Diameter (D), Front Length (L_f), Rear Length (L_r), Front Radius (R_f), Rear Radius (R_r) and Middle Radius (R_m). The output variables represent the Drag Coefficient (C_D) and the Turbulent Kinetic energy (k) produced by the body in its locomotion. The physical evaluation was done using the simulation software Ansys and by changing the Flow Velocity (U_{in}) from 1 m/s to 10 m/s. The domain of each geometrical variable was restrained to a finite interval. Then the values of each of the geometrical variables were uniformly distributed through the domain by utilizing the Uniform Design Method (UDM). In order to get the best results, i.e. minimizing both values of the output variables by changing the input variables values, an optimization procedure was taken. To do so, the software Fortran was used to utilize an Artificial Neural Network (ANN) with the application of a Genetic Algorithm (GA).

The results obtained show that there are two geometrical variables with a major impact in this optimization problem: Diameter (D) and Rear Radius (R_r). The optimization procedure achieved an improvement of 15.2 % in the Drag Coefficient (C_D) and 0.83 % in the Turbulent Kinetic Energy (k).

(This page was intentionally left blank)

Resumo

Esta dissertação tem como objectivo estudar a hidrodinâmica de um Veículo Autónomo Subaquático (AUV) que seja capaz de alterar a sua morfologia de forma a atingir a melhor performance possível em diferentes situações. A necessidade de veículos autónomos subaquáticos de elevada performance tem vindo a crescer na área de engenharia marítima devido ao aumento de atividades de exploração e investigação marítima. É esperado que as novas gerações deste tipo de veículos sejam capazes de operar em condições marítimas adversas e complexas. Nesta dissertação é proposto um veículo autónomo subaquático com um design híbrido inspirado numa forma biomórfica.

De forma a desenvolver este estudo, foi criado um corpo bi-dimensional composto por sete variáveis. Este corpo foi alvo de um estudo hidrodinâmico com o intuito de obter duas variáveis de saída. As variáveis de entrada são geométricas e correspondem à geometria do corpo em estudo: Comprimento (L), Diâmetro (D), Comprimento Frontal (L_f), Comprimento Traseiro (L_r), Raio Frontal (R_f), Raio Traseiro (R_r) e Raio Médio (R_m). As variáveis de saída representam o Coeficiente de Atrito (C_D) e a Energia de Turbulência Cinética (k), produzidas pelo movimento de deslocação do corpo. A avaliação dinâmica foi realizada utilizando o software de simulação Ansys, através da alteração da Velocidade do Escoamento (U_{in}) desde 1 m/s até 10 m/s. O domínio de cada variável geométrica foi limitado a um intervalo fixo. Posteriormente os valores de cada variável geométrica foram distribuídos uniformemente pelo dominio considerado, utilizando o Método de Design Uniforme (UDM). De forma a obter os melhores resultados possíveis, i.e minimizar o valor das duas variáveis de saída alterando o valor das sete variáveis de entrada, um método de otimização foi utilizado. Deste modo,para a realização deste método, o software Fortran foi aplicado de forma a pôr em ação uma Rede Artificial Neuronal (ANN) com a aplicação de um Algoritmo Genético (GA).

Os resultados obtidos mostram a existência de duas variáveis geométricas com maior peso para o processo de otimização: Diâmetro (D) e Raio Traseiro (R_r). O processo de otimização foi capaz de alcançar uma melhoria de 15.2 % para o Coeficiente de Atrito (C_D) e de 0.83 % para a Energia de Turbulência Cinética (k).

(This page was intentionally left blank)

Résumé

Cette dissertation poursuit l'étude de l'hydrodynamique d'un véhicule autonome sousmarin (AUV), capable de changer sa morphologie afin d'atteindre la meilleure performance possible selon les différentes situations. La nécessité de véhicules autonomes sousmarins de performance élevée s'est accrue dans le domaine de l'ingénierie maritime, dû à l'augmentation d'activités d'exploration et de recherche maritimes. Les nouvelles générations de ce type de véhicules sont censées opérer dans des conditions maritimes adverses et complexes. Cette dissertation propose un véhicule autonome sous-marin (AUV) avec un design hybride, inspiré d'une forme biomorphique.

Un corps bidimensionnel, composé de sept variables, a été conçu avec l'objectif d'approfondir cette étude. Ce corps a fait l'objet d'une étude hydrodynamique qui a pour but obtenir deux variables de sortie. Les variables d'entrée sont géométriques, et celles-ci correspondent à la géométrie du corps ici observé: Longueur (L), Diamètre (D), Longueur Frontal (Lf), Longueur Arrière (Lr), Rayon Frontal (Rf), Rayon Arrière (Rr) et Rayon Moyen (Rm). Les variables de sortie représentent le Coefficient de Frottement (CD) et l'Énergie Cinétique Turbulente (k) produits par le déplacement du corps. L'évaluation dynamique a été réalisée en utilisant le software de simulation Ansys, à travers la modification de la Vitesse d'Écoulement (Uin) dès 1 m/s jusqu'à 10 m/s. Le domaine de chaque variable géométrique a été limité à un intervalle fixe. Ensuite, les valeurs de chaque variable géométrique ont été réparties uniformément sur le domaine considéré, en appliquant la Méthode de Design Uniforme (UDM). La méthode d'optimisation utilisée consiste à réduire les valeurs des deux variables de sortie en changeant les sept variables d'entrée, dans le but d'atteindre les meilleurs résultats. Ainsi, le software Fortran a mis en œuvre un Réseau Neuronal Artificiel (ANN) avec l'application de l'Algorithme Génétique (GA).

Les résultats obtenus démontrent qu'il existe deux variables géométriques plus déterminantes dans le processus d'optimisation: Diamètre (D) et le Rayon Arrière (Rr). Le processus d'optimisation a atteint une amélioration de 15.2% pour le Coefficient de Frottement (CD) et de 0.83% pour l'Énergie Cinétique Turbulente (k).

(This page was intentionally left blank)

Contents

List of Figures

(This page was intentionally left blank)

List of Tables

Chapter 1

Introduction

In this chapter a brief contextualization of the following work to be developed is given in order to get the reader's attention to the reasons that guide the author to do this work. The objectives stipulated to this work are also presented as well as this dissertation outline.

1.1 Thesis Context

The ocean as been a source of inspiration for inventors and researchers over the last three millennia. The greatest part of Earth's biodiversity, approximately 90%, lives in the ocean and studies have proven that species are always adapting and evolving to overcome their habitat difficulties and thrive. That kind of behaviour as been studied by humans in their attempt to adopt some natural features to science and engineering. The ocean exploration started very soon in human civilization, since 4500 B.C with divers in Greek and Chinese cultures to the genesis of ship-borne deep-sea research in the 17^{th} Century by the hands of Sir James Clark Ross and has continued until today. The issues and goals are different nowadays and they are a lot more difficult to achieve. We are talking about issues like aquaculture monitoring to ocean mapping and prospecting, this kind of challenges require sophisticated technology capable of satisfying the requirements of these tasks so they can be solved the most efficient and sustainable way [44].

Aquaculture has grown very rapidly during the past 20 years and the tendency is to continue it's growth because of the need of protein required to feed humans and other animals. Recent studies have predicted that fish consumption in developing and developed countries will increase by 57 % and 4 %, respectively [41]. This demand can cause problems in product quality because of the development of fish diseases due to fish concentration and nutrition [42]. To avoid this problem and to estimate the fish population development, constant monitoring is needed and to do so human divers are required, although they are being submitted to this possibly infected environment and their actions are restricted in low-visibility environments [36].

Ocean mapping and prospecting are activities done in deep waters, so humans divers are incapable of playing this role. To perform this kind of activities and others like inspection of structures of oil and gas extraction, inspection of offshore wind turbine structures, military missions and so on, Remotely Operated Vehicles (ROVs) are used. The major drawback is that they are expensive and require human command, becoming very time-consuming, which increases the overall costs and most important they are ineffective in a group of tasks because of their shape and dimensions [55].

To overcome this issue and be able to execute the tasks mentioned above, humans have developed Autonomous Underwater Vehicles (AUVs) with shapes resembling marine species. This attempt to copy and adapt bio-inspired forms is based on the fact that fishes possess capabilities such as: high energy efficiency, high velocity, silent swimming, high manoeuvrability and stability. All these characteristics are goals to achieve on the design and engineering of this kind of technology, but it's impossible to cover them all with one kind of AUV because of its shape. This means that a vehicle projected to perform a long-distance and high-velocity mission is going to underperform on a mission requiring high manoeuvrability in a confined space, this requires a vehicle able to switch its form depending on the mission objectives [19].

This capability will revolutionize the AUV industry. Having a vehicle able to excel in opposite tasks by changing its form would guaranty efficiency and sustainability throughout its mission, reflecting in a higher payoff to the investors and a smaller ecologic step for the environment.

1.2 Objectives

Conceptual biomimetic AUVs are being designed and built all over the world because of their obvious advantages over traditional AUVs. TECMAR, one of the INEGI groups that works specifically with technologies and procedures that contribute for the appreciation of the marine resources, understands the potential of this concept to develop new vehicles and is pursuing a realistic model of an AUV capable of morphing its shape to provide a self-sustainable autonomous operating vehicle. The objective of this work is to get some initial insight on what shapes are best suited for the diverse operating modes and how to optimize those shapes, collecting relevant data of hydrodynamic variables that characterize their performance. To perform Computational Fluid Dynamics (CFD) simulations, the software Ansys was used and to perform sensibility analysis and optimization, the softwares MATLAB and Fortran were utilized. This dissertation studies numerically a bi-dimensional torpedo shape to evaluate the impact of geometrical variables and flow velocity on its performance. Afterwards, an optimization of the previous mentioned variables values was done by recurring to an Artificial Neural Network (ANN). This results in an optimal body shape for each of the ten velocities.

1.3 Outline

The present study case is divided in six chapters and eight appendices.

Chapter 2 gives a review of the concepts and technological developments of the main subjects of this work.

Chapter 3 introduces the methods and tools used to model the problem.

Chapter 4 includes a detailed description of the process developed to solve the presented problem.

Chapter 5 presents and discusses the results obtained in this work.

Chapter 6 presents the conclusion and provides insights of a possible future work.

Appendix A presents figures from the Ansys Design Modeler work station.

Appendix B presents the figures from the Mesh construction procedure developed in Ansys Mesh.

Appendix C presents figures that represent the step-by-step configuration procedure utilized in Ansys Fluent.

Appendix D presents the tables utilized to apply the Uniform Design Method (UDM) as well as the tables obtained for the present work.

Appendix E presents the MATLAB code developed to build an Artificial Neural Network (ANN) with the application of Genetic Algorithm (GA).

Appendix F presents the tables from Ansys Fluent: input and output variables.

Appendix G presents figures from the results obtained for the Sensitivity Index (Sobol)

Appendix H presents the figures obtained for the optimization procedure

(This page was intentionally left blank)

Chapter 2

State of the Art

In this chapter a revision and identification of each one of the following themes is done. The objective is to give the reader a basic knowledge about the state of development, methods, applications and existing models of each of this topics: Autonomous Underwater Vehicles (AUV), Swimming Modes, Uniform Design Method (UDM), Artificial Neural Networks (ANN) and Genetic Algorithm (GA).

2.1 Autonomous Underwater Vehicles

2.1.1 Overview

The need for underwater exploration and exploitation has driven researchers and investigators to build a machine able to perform those tasks. That's where Unmanned Underwater Vehicles (UUV) were born. First vehicles to be born were Remotely Operated Underwater Vehicles (ROV) that, as the name says, are remotely controled vehicles that can be manipulated at long ranges and steepness. Due to the risks to human life in underwater operations, these vehicles have gradually replaced divers and manned submersibles. Most of the developed systems are designed to be controlled by an experienced user, who manipulates it from the surface. In addition to safety, ROV's have offered a more effective and low-cost method for underwater research and sea exploitation. This is the primary reason for the rapid development of numerous vehicles over the past few years [38].

2.1.2 Technology development

In order to reduce time consumption, resources and to optimize data collection a new kind of vehicle has been developed, an Autonomous Underwater Vehicle (AUV). This machine is independent of human direct control by having a computer on board programmed to follow a pre-programmed course and is able to navigate using arrays of acoustic beacons on the seafloor or a combination of Ultra Short Base Line (USBL) acoustic communication, GPS positioning and inertial navigation. This technological achievement is revolutionising our ability to map and monitor the marine environment [56].

Unlike submarine gliders, which are propelled using a buoyancy engine and have an undulating trajectory, AUV's are able to maintain a linear trajectory through the water and are therefore well suited to geo-science applications requiring constant altitude, such as seabed mapping and sub-bottom profiling. Typically deployed from a surface vessel, this machine can operate independently of that vessel for periods of a few hours to several days

[56]. Remotely Operated Vehicles (ROVs) remain tethered to the host vessel, although this enables them to draw more power and communicate real-time data. Their speed, mobility and spatial range are very limited compared with an AUV. The wholly autonomous nature of some AUVs means that the deploying vessel can be used for other tasks while the AUV is in the water, dramatically increasing the amount of data that can be collected for a given amount of time.

Nowadays most of the existing underwater vehicles are propeller driven. Even though they are easy to control, their propulsion mode has some drawbacks, such as loud noise, large size, low efficiency, as well as poor manoeuvring at low speed. In addition, another significant drawback related to marine aquaculture applications is the acoustic noise caused by the propeller. It inevitably interrupts the cluster of fish and results in low yield [36]. Most AUV's are torpedo-shaped (e.g. the NERC Autosub6000 AUV; Figure 2.1), but some have a more complex configuration allowing them to move slowly and across complex terrain, e.g. the WHOI ABE (Figure 2.2) and SENTRY (Figure 2.3) AUVs [56].

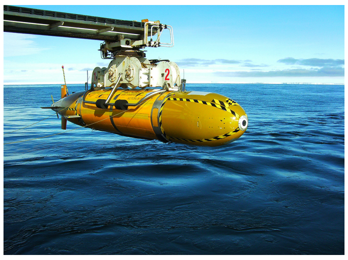


Figure 2.1: NERC Autosub6000 AUV

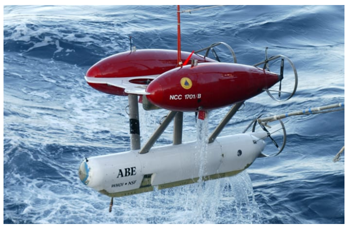


Figure 2.2: WHOI ABE AUV

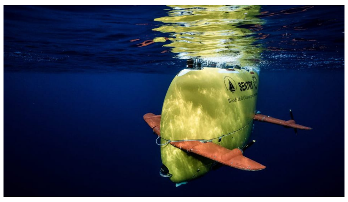


Figure 2.3: SENTRY AUV

The aspects to be optimized on AUV's are mainly their autonomy and safety conditions [1]. In furtherance to increase travelling time, the vehicle needs to spend less energy in its locomotion and to do so its hydrodynamic shape is a crucial criterion. A way to achieve this optimal shape is to use biomimetic technologies in order to develop a fish-like robot, since this animals have thrived and excel in this environment. That's when the concept of Bio-inspired AUV's was born.

There are some existing prototypes such as: MantaDroid (Figure 2.4), that emulates a manta ray. This model excels in endurance and silence swimming, it has autonomy up to ten hours and can be used in aquaculture surveillance [45]. The Tunabot, this prototype mimics a Yellowfin tuna shape and locomotion achieving nearly equivalent speeds when compared with live specimens (Figure 2.5) [22]. Another model is called the Eelume, and replicates a eel. Eelume vehicle is basically self-propelled robotic arm whose slender and flexible body can transit over long distances and carry out Inspection, Maintenance and Repair (IMR) equipment in confined spaces not accessible by conventional underwater vehicles. It's most relevant characteristic is it's amazing maneuverability (Figure 2.6) [21].

AUVs have already proven their efficacy and superiority to other UUV in many circumstances, but it's still a developing technology although it's almost sixty five years of existence [27]. With the aim of producing the best aquatic vehicle possible information and inspiration must be withdrawn from aquatic creatures. The spotlight of creation will

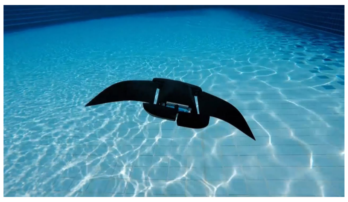


Figure 2.4: MantaDroid AUV

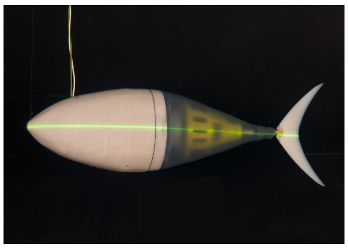


Figure 2.5: Tunabot AUV

Figure 2.6: Eelume AUV

of course differ according to the vehicle function, but as mentioned before, the main reason of producing this kind of vehicles is its autonomy and efficiency. That means they should also be able to operate in different tasks, adding the component of versatility to their core characteristics. This is a simple definition to an abstract answer, considering that there are many environmental factors that will clout this kind of vehicles performance, besides it's shape and propeller method, like currents, different pressure zones, different temperatures, animals and so on. Since these variables can not be changed or contained, our job is to do the best in every controlled design parameter. An AUV design capable of morphing between different bio-shapes and propeller methods would vastly increase it's range of applications and would be able to excel it's operation in different situations.

2.2 Swimming Modes

2.2.1 Overview

Fish are marine animals that have adapted and evolved to thrive in a particularly hospice environments. Due to natural selection the mechanisms developed by these are highly efficient with regard to the mode of life, activity and habitat for each species. Some fish are highly manoeuvrable, some are fast long-distance swimmers, others are power-efficient endurance swimmers and others are short burst of speed swimmers (Figure 2.8). Despite their differences, all of them are equally adapted to their way of living and that is reflected in their body characteristics [16].

The main properties of water as a locomotion medium are its incompressibility and its high density. Since water is an incompressible fluid, any movement performed by an aquatic animal will set the water surrounding in motion and vice-versa. Due to its high density, nearly equal to the animal body density, it counterbalances the force of gravity so the weight is not of primary importance.

2.2.2 Swimming mechanics

Swimming consists in the transfer of momentum from the fish to the surrounding water (and vice-versa) via drag, lift and acceleration reaction forces.

Swimming drag can be decomposed into three components: friction or viscous drag, form drag and vortex or induced drag. Friction or viscous drag is due to skin friction between the fish and the boundary layer of water, it arises as a result of the viscosity of water in areas of flow with large velocity gradients. Form drag is caused by the distortion

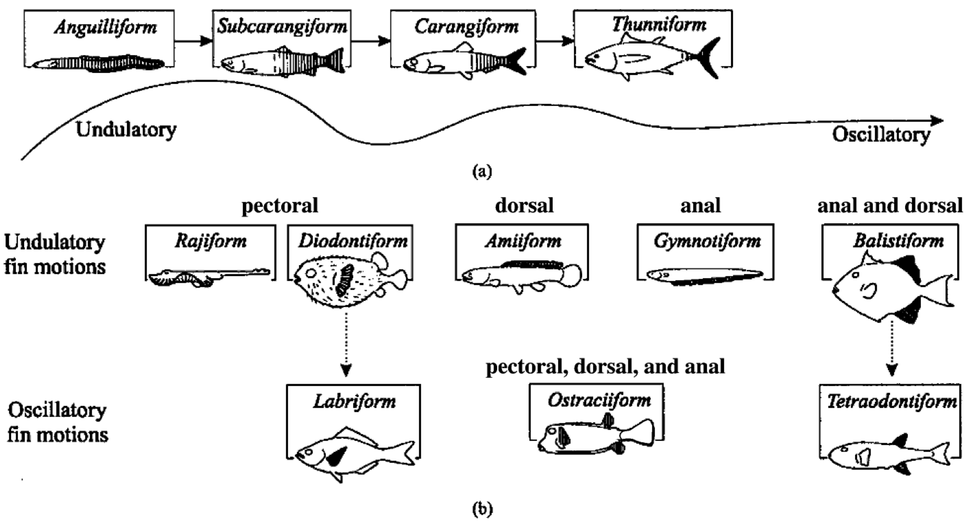


Figure 2.7: BCF (a) and MPF (b) swimmers

of flow around solid bodies. Vortex or induced drag results from the energy lost in the vortexes formed by the caudal and pectoral fins as they generate lift or thrust. Logically, this characteristics are antagonistic and so it's impossible to fully satisfy each one of them, at the same time, with the same stationary body [26].

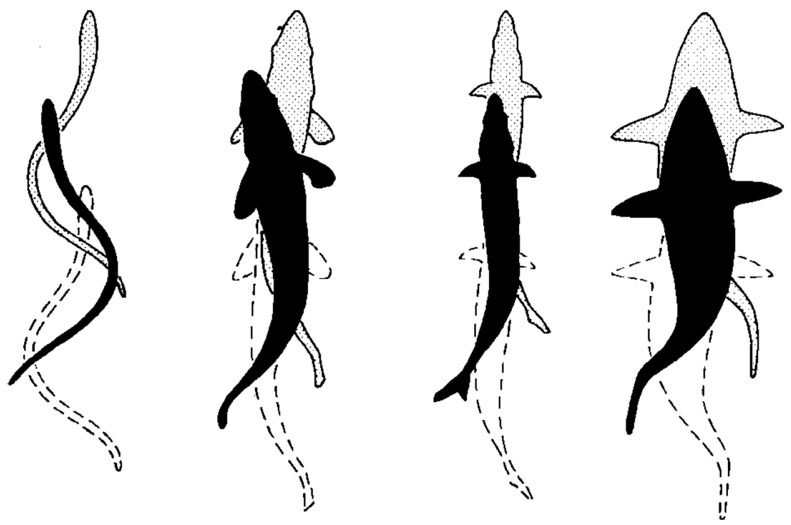


Figure 2.8: Different type of marine animals locomotion methods

Lift forces are caused by asymmetries in the flow, as a fluid moves past an object, the pattern of flow may be such that the pressure on one lateral side is greater than on the opposite side. Lift is then exerted on the object in a direction perpendicular to the flow direction. Acceleration reaction is an inertial force, generated by the resistance of the water surrounding a body or an appendage when the velocity of the latter relative to the water is changing [51].

It's possible to summarize fish locomotion capability by looking into two aspects in their morphology: body shape and fins. Body shape is defined by the structure and form of the fish, it will influence the animal interaction with its locomotion surroundings. A simple way to compare different shapes and their respective capacity, is by their cross-section: there are fusiform animals that are considered cruising specialists, compressed animals with tall elliptical shapes that are considered manoeuvre specialists, depressed animals that are able to sit on the ocean floor, and circular animals that are accelerating specialists. Fins are swimming tools developed by fish so they can propel and stabilize their body's in the water, see Figure 2.9. Although these functions differ from species to species, in most marine animals, pectoral fins are mainly used for steering and locomotion, pelvic fins for stabilizing and braking, dorsal and anal fins are used for balance, stabilization and to prevent pitching and spinning and caudal fins are used for thrust or propel [35].

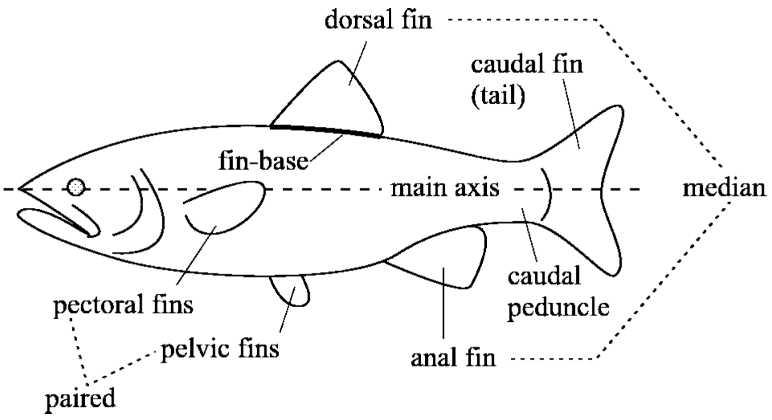


Figure 2.9: Fish morphology terminology

Most fish generate thrust by bending their bodies into a backward-moving propulsive wave that extends to its caudal fin, a type of swimming classified underbody and/or caudal fin (BCF) locomotion. Other fish have developed alternative swimming mechanisms that involve the use of their median and pectoral fins, termed median and/or paired fin (MPF) locomotion. Because BCF relies on powerfull caudal body structures that can thrust only rearwards, this form of locomotion is particularly effective for accelerating quickly and cruising. In MPF swimmers, propulsive forces are generated by multiple fins, this ables them to coordinate their fins motion to execute elaborated turns, so this type of swimming is well adapted for high manoeuvrability [26].

Inside these two major groups, there are twelve different kinds of sub-groups. Belonging to the BCF family there is anguilliform, subcarangiform, carangiform and thunniform. Inside MPF there is rajiform, diodontiform, amiiform, gymnotiform, balistiform, labriform, tetraodontiform and ostraciiform. Those types can be seen in Figure 2.7 [16].

With the aim of producing a bio-inspired AUV, there are a few determining factors that can be found in each one of this types of swimming modes, however it's impossible to gather them all in only one prototype, so the main focus would be redirected to assemble the main characteristics of thunniform and anguilliform swimmers, both belonging to the BCF group. This option is supported on the fact that thunniform swimmers have the most efficient locomotion mode developed in the aquatic environment, that allows high cruising speeds to be maintained for long periods of time. Along with the power of the caudal propellers (more than 90%), the well-streamlined body shape and weight distribution, besides reducing pressure drag, ensures that the recoil forces are effectively minimized. This design accomplishes an optimized form for high-speed swimming, nonetheless is not well suited for actions like slow swimming, turning manoeuvres or rapid acceleration. On the opposite side there are anguilliform swimmers, they are slow speed swimmers but they are accelerating and manoeuvre specialists. Because of their flexible bodies they are capable of changing their direction and navigate through confined spaces very easily. A combination of this two kind of body shapes would be able to perform exquisitely in a vast number of situations and conditions [51].

2.3 Selection of Training Datasets (Uniform Design Method)

2.3.1 Overview

Classical experimental designs are mostly based on analysis of variance (ANOVA) models that involve main effects, interactions and random error. The objective is to provide a good estimation for all parameters with a reasonable number of experiments. When the number of factors increases, the number of parameters in this method increases exponentially, this means that the number of experiments will do too. This fact brings a problem related with computer processing due to extensive time-consumption [23].

2.3.2 Uniform Design Method creation

The optimal regression design is based on a pre-specified regression model.

$$Y = \sum_{i=1}^{k} \beta_i g_i(x_1, ..., x_s) + \epsilon \tag{2.1}$$

where $x_1, ..., x_s$ are s input factors, β_i's are unknown parameters, ϵ is the random error and g_i's are unknown functions, so Equation 2.1 can be represented as

$$Y = h(x_1, ..., x_s) + \epsilon \tag{2.2}$$

where the function h is unknown. The goal is to estimate the average value $E(h(x))$ over the experimental domain, where $h(x)$ is the output parameter of the experiment. Without loss of generality, is assumed that experimental domain is the unit cube C^s, so

$$E(h(x)) = \int_{C^s} h(x)dx \tag{2.3}$$

It usually can be estimated by

$$\bar{h} = \frac{1}{n} \sum_{x \in P} h(x) \tag{2.4}$$

where P is a set of n experimental points over the domain. The goal is to achieve an experimental design that estimates $E(h(x))$ in an efficient way [23].

Fang (1980) proposed the Uniform Design Method (UDM), that allocates experimental points uniformly scattered on the experimental domain. UDM is a space filling experimental design in a deterministic uniform fashion, that selects points from the center of cells. A distinctive feature of this method is the introduction of number-theoretic method, it's target is to find a set of points that are uniformly scattered over an s-dimensional unit cube and this set is used instead of random number in Monte Carlo method. This way UDM requires one-dimensional balance and s-dimensional uniformity [13].

This method is distinguished from the traditional ones in the way that it simultaneously deletes insignificant variables and estimates the coefficients of significant variables. In fact, the UDM can be considered as a kind of experimental design with the aim of minimizing discrepancy. It possesses an oracle property (it is robust to the underlying model assumption), which means that it performs as well as if the true model were known in advance. Besides that, within a small number of experimental runs, a significant amount of information can be obtained for exploring the relationships between the response and the contributing factors. The theoretical backup of uniform design rests on the theory of numbers and quasi-Monte Carlo method [37].

2.3.3 Discrepancy concept

The Koksma-Hlawka inequality gives the upper error bounds of the estimate of $E(h(x))$

$$|E(h(x)) - \bar{h}| \leq D(P)V(h) \tag{2.5}$$

where $V(h)$ is a measure of the variation of h and is independent of the design points. $D(P)$ is the discrepancy of P. So, given a bounded $V(h)$, Equation 2.5 indicates that the more uniform a set P of points is over the experimental region C^s, the more accurate $\bar{h}$ is as an estimator of $E(h(x))$. Therefore, one should choose a set of experimental points with smallest discrepancy among all possible designs for a given number of factors and experimental runs. This is the fundamental idea of UDM. Note that the UD is robust against changes of the function h for which $V(h)$ remains unchanged, this fact indicates that UD provides a good estimate of $E(h(x))$ for a very large class of $h(x)$. The key issue to be addressed then is how to find n points in C^s with minimum discrepancy [23].

Suppose there are s factors of interest over a standard domain C^s. The goal here is to choose a set of n points P_n such that these points are uniformly scattered on C^s. $M(Pn)$ is a measure of the non-uniformity of Pn, the goal is to find a set of points that minimize M. From the Koksma-Hlawka inequality (Equation 2.5), a natural choice of M is the discrepancy $D(P)$. Being $F_n(x)$ the empirical distribution function of P_n

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} I(x_i \leq x) \tag{2.6}$$

where $I(\cdot)$ is the indicator function. Then the discrepancy D_p can be written as:

$$D_p(P_n) = \left[\int_{C^s} |F_n(x) - F(x)|^p dx\right]^{\frac{1}{p}} \tag{2.7}$$

where $F(x)$ is the uniform distribution function on C^s. The popular D_∞ discrepancy can be obtained by taking $p = \infty$ in Equation 2.7 is called the star discrepancy or discrepancy for simplicity. This is the most commonly used measurement for discrepancy and can be reexpressed as follows:

$$D(P_n) = \sup_{x \in C^s} |F_n(x) - F(x)| \tag{2.8}$$

The discrepancy has been universally accepted in quasi-Monte Carlo methods and number-theoretic methods.

2.3.4 Application and development

A UDM table is denoted by $U_n(q^s)$, being U the uniform design, n the number of samples, q the number of levels of each input variable, and s the maximum number of columns of the table. There is an accessory table for each UDM table, including recommendations of columns with minimum discrepancy for a given number of input parameters [17].

Details of the algorithm for constructing a $U_n(q^s)$ table are given as follows:

- For a given n, find the set $H_n = (h_1, h_2, \dots h_m)$ with m. d. c. $(n, h_i) = 1$ and $h_i \leq n$, $i = 1, \dots, m$, with $m = \phi(n)$, where ϕ is the Euler function

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)...\left(1 - \frac{1}{p_t}\right) \tag{2.9}$$

  and $n = p_1^{r_1} p_2^{r_2} ... p_t^{r_t}$ is the prime decomposition of n.

- For any s distinct elements of H_n, generate an $n \times s$ table where $u_{ij} = ih_j \pmod{n}$ (for $i = 1, \dots, n$ and $j = 1, \dots, s$) and $0 < u_{ij} \leq n$, this is, $u_{ij} = n$ if $ih_j = kn$ for $k \in \mathrm{N}$.

Finally, the UDM table must be transformed into a hyperrectangle region corresponding to the input variable domain by linear transformation. The idea is to obtain a relationship between the design variables on the interval $[\bar{\pi} - 2\alpha\bar{\pi}, \bar{\pi} + 2\alpha\bar{\pi}]$ and the output values, being $\bar{\pi}$ the mean value of the domain and α a constant value for each variable defined as the Variation Coefficient [17].

The uniform design method has the following advantages: (1) it is able to produce samples with high representativeness in the domain; (2) it imposes no strong assumption on the model; and (3) it accommodates the largest possible number of levels for each factor among all experimental designs. Due to these advantages, the Uniform Design Method has been applied to the fields of chemistry and chemical engineering, pharmaceutics, quality engineering, system engineering, survey design, computer sciences and natural sciences [13].

2.4 Modeling (Artificial Neural Network)

2.4.1 Overview

A Artificial Neural Network (ANN) is, as once said by Dr. Robert Hecht Nielsen, the inventor of first neurocomputer: a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs [12]. ANN are processing devices (algorithms or actual harware) that are loosely modeled after the neuronal structure of the cerebral cortex but on a much smaller scale. The intention of this working networks is not to replicate the operation of the biological systems but to make use of what is known of the biological networks for solving complex problems.

The attractiveness of ANNs comes from their remarkable information processing characteristics, mainly to their non-linearity, high parallelism, robustness, fault and failure tolerance, ability to handle imprecise and fuzzy information (noise tolerance), learning and generalization capabilities [10].

Artificial models possessing such characteristics are desirable because (i) nonlinearity allows better fit to the data, (ii) operation and the analogy between ANNs and bionoise-insensitivity provides accurate prediction in the presence of uncertain data and measurement errors, (iii) high parallelism implies fast processing and hardware failure-tolerance, (iv) learning and adaptivity allows the system to update (modify) its internal structure in response to changing environment, (v) generalization enables application of the model to unlearned data.

ANNs are computational modeling tools that have emerged and found extensive acceptance in many disciplines for modeling real-world problems. They were first used in the fields of cognitive science and engineering but recently their usage has significantly increased in other fields such as sonar target recognition, car navigation, image compression, signal prediction and forecasting, recognizing hand-written ZIP codes, speech recognition and even backgammon. In micro biology, they have been used extensively ranging from modeling, classification, pattern recognition and multivariate data analysis. The main goal of an ANN-based computing is to develop mathematical algorithms that will enable ANNs to learn by mimicking information processing and knowledge aquisition in the human brain, it is the foundation of Artificial Intelligence (AI) [34].

2.4.2 Biological and Artificial Neural Networks

The human nervous system consists in billions of neurons of various types and lengths relevant to their function. In Figure 2.10 is represented a schematic draw of a biological neuron composed by three major funtional units: dendrites, cell body and axon. The cell body incorporates the nucleus that contains cell data and information, the dendrites are responsible for receiving signals from other neurons and pass that information to the cell body and the axon, which branches into collaterals, receives signals from the cell body and carries them to the dendrites of the neighboring neurons through a microscopic gap named synaptic gap (Figure 2.11).

An impulse, in the form of electric signal is received at the dendrites of one neuron and travels through the cell body towards the axon until finally reaches the synaptic gap. Upon the arrival of the signal to the membrane a chemical neurotransmitter is released in quantity proportional to the strength of the incoming signal. This signal is received by the dendrites of the upon neuron, forcing it to generate a new electrical signal and do the same process from the earlier neuron and so on. This simplified mechanism is the

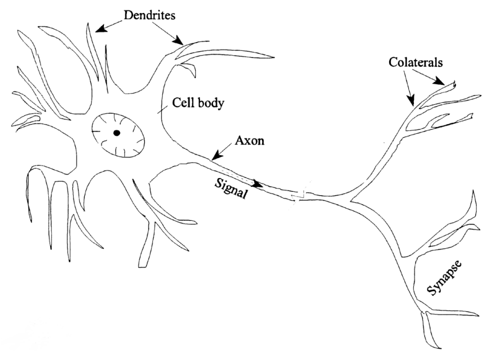


Figure 2.10: Schema of a biological neuron

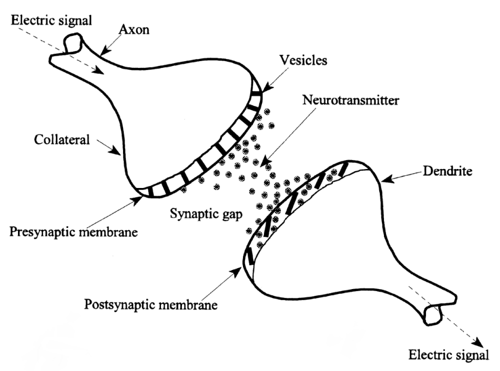


Figure 2.11: Schema of a synaptic gap

foundation of the neurocomputing development and the operation of the building unit of an ANN [10].

The analogy made between biological and artificial neurons is that connections between the artificial nodes represent the axons and dendrites, the synapses are replaced with the connection weights and the cell body and respective nucleus are replenished by a mathematical function. As Figure 2.12 ilustrates there can be n artificial neurons with various signals of intensity x and connection weight w feeding into a neuron with a threshold of b, also called bias. This is similar to a biological neuron, and both networks learn by incrementally adjusting the magnitudes of the weights or synapse's strengths.

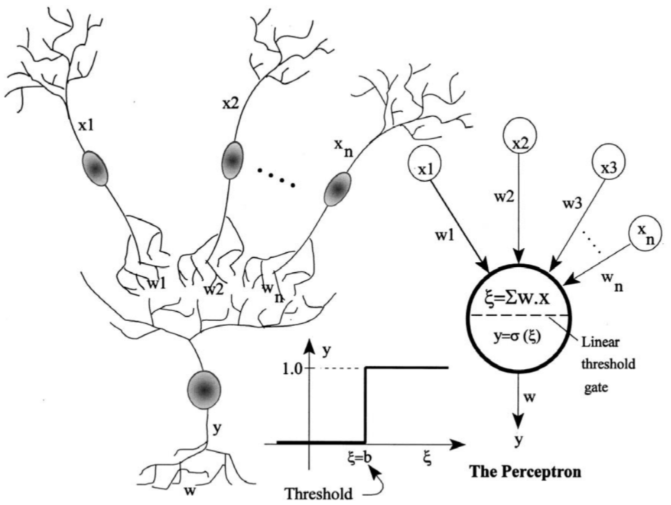


Figure 2.12: Connections in biological and artificial neurons

An artificial neuron receives inputs from the environment and combines them in a peculiar way to form a net input (ξ), passes that information through the threshold and emits an output (y) to the next neuron of the network. Only when ξ value is greater than the neuron threshold or bias (b) the neuron is activated. The network input is computed as the inner product of the input signals (x) and their respective strengths (w). For n

signals the artificial neuron or perceptron neuron operation can be expressed as:

$$y = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w_i x_i \geq b; \\ 0, & \text{if } \sum_{i=1}^{n} w_i x_i < b \end{cases} \tag{2.10}$$

where when the output signal is equal to one the neuron is activated and when the value is zero it isn't. Positive connection weights ($w_i > 0$) excite the neuron and enhance the net signal ξ, on the other hand when the connection weights are negative ($w_i < 0$) they reduce the neuron activity and inhibit the net signal.

The perceptron can be trained on a set of examples using a special learning rule (Hecht-Nielsen, 1990). The connection weigths are changed in proportion to the difference of the network output (y) and the correct or target output (Y), this difference defines the error function. This function forms an irregular multidimensional complex hyperplane with many peaks, saddle points and minima. Using a particular search technique the learning process strives to obtain a group of connection weights that correspond to the global minimum.

In order to obtain results for nonlinear separable problems, the network as the need of additional layer(s) between the input and the output layers. This architecture is called multilayer perceptron (MLP). The learning of a MLP is not so simple and direct as the two layer perceptron. For example a Backpropagation MLP trained by the delta learning rule is a well known and applied example. Both this elements will be discused later in this section.

In a perceptron, learning can be discribed as the process of updating the internal representation of the system in response to external stimuli so that it can perform a specific task. This means it's necessary morphing the network architecture, this consists in adjusting the connection weights, pruning or creating some connection links and change some firing rules for each neuron. This learning process is performed iteratively when the network is confronted with new training examples. An ANN is said to have learnt if it can handle imprecise, fuzzy, noisy and probabilistic information without noticeable adverse effect on it's response and when the network is capable of generalize from the known learned tasks to the unknown ones.

2.4.3 Computational problems

There are seven categories of important problems that an ANN outperform other computacional tools, these are: pattern classification, clustering, function approximation, forecasting, optimization, association and control.

- Pattern classification

This category deals with the problem of assigning an unknown input pattern to a certain class based on properties that characterize that class. Unlike other tools an ANN does not require the linearity assumption, so it can be applied to non-linear separable classes.

- Clustering

The cluster problem consists in the construction of classes by exploration of the similarities between the input patterns based on their inter-correlations. The network will designate similar patterns to the same cluster/class.

- Function approximation

The function approximation or modeling involves the problem of training the ANN on input and output data so as to approximate the underlying rules relating the inputs to the outputs.

- Forecasting

The forecasting problem consists in the prediction of one behavior, to do so, the computacional tool must be trained on samples from a time series representing a certain phenomenon at a given scenario and then using that phenomenon to forecast the behaviour for other scenarios.

- Optimization

Optmization consists in finding the best solution for the current problem, wether can be finding a value that maximizes or minimizes the objective function subject to a group of problem or real world constrains.

- Association

Association can be dealt by developing a network using noise-free data in order to use that network to evaluate noise-corrupted data and to correct or reconstruct the corrupted or missing data.

- Control

This last problem consists in designing a network capable of assuming an adaptive control system to generate the required control inputs, such that the system will follow a certain path based on it's own feedback.

2.4.4 Classification

There are a series of parameters to classify a operating network, they can be based on the degree of connectivity of the neuron in the network, the function that the network is design to serve, the direction of the flow of information within the network, the type of learning algorithm, the learning rule and the degree of learning.

There are three types of learning: supervised learning, this method consists in training the network with the target outputs for each example and using the error between the network solution and them to adjust the connections weights; reinforcement learning is a kind of supervised learning but instead of the network having access to the correct answer it is provided with a critique on correctness of the desired output; unsupervised learning does not require the correct answer for each training example, however the network, through exploring the data and correlation between the examples, organizes the data into clusters based on their similarity; there is also hybrid learning that combines supervised and unsupervised learning.

It's also important to discuss the kind of existing learning rules and their respective utilization in each type of learning. There are basically four types of learning rules and these define the way how the network weights are updated between training cycles. The error-correction learning (ECL) rule is used in supervised learning in which the arithmetic difference or error is used to modify the connection weights in a way to reduce is effective value. The Boltzmann learning (BL) rule is a stochastic rule similar to ECL but is based on Boltzmann statistical distribution which renders learning extremely slower. The Hebbian learning (HL) rule is the oldest one and it's based on the synchronous activation of both neurons on one synapse and due to the repetition of this event the connection strength is increased, therefore unlike the previous mentioned methods this method learns locally by adjusting the synapse weight based on the neurons activity. For last there is Competitive learning (CL) rule where all the neurons are forced to compete against each other and only one will activate, this way the connection weights are updated according to this activity.

2.4.5 Backpropagation

This kind of networks are the most widely used and are considered the workhorse of artificial neural networks [34]. A Backpropagation (BP) is a multilayer network constituted by an input layer, an output layer and one or more hidden layer/s, these last are responsible to capture the nonlinearity in the data, see Figure 2.13. They use a supervised learning method with an error-correction learning rule (ECL) and are capable of learning the mapping from one data space to another using known examples. The term backpropagation is due to the reverse direction of the error in the network architecture, this means that when the network as finished the process and given a solution to the problem, the difference between the obtained solution and the optimum solution, the so called error, travels from the output layer until it reaches the input layer so that the connection weights can be modified. The collective effect of all the nodes is summed up by action of the product between each node value and it's respective connection weight, once the net effect on one node is determined the activation at that node is calculated using a transfer function, for example the sigmoidal function, to yield an output between 0 and +1 or -1 and +1. The amount of activation obtained represents the new signal that is going to be transfered to the following layer and this is aplied for all the layers constituing the network. The forward and backward processes are perfomed repeatedly until the ANN solution is within the tolerance for the optimum solution. These networks are very versatile and can be used and excel in almost all computational problems.

2.4.6 Development

The development of an operational and successful network consists in a cycle of six phases: problem definition and formulation, system design, system realization, system verification, system implementation and system maintenance. The process starts with a good problem definition and formulation, this is the foundation of the network and it's very relevant to the following work. The next step is system design where the type of network and the learning rule are established, this is also where the data is collected and pre-processed to fit the type of ANN used, this includes a statistical analysis and division of the contained data into three subsets: training, test and validation. The following step, system realization, where the training of the network occurs. After this section is complete it's time for system verification where the network is examined for its generalization capability, this means that the network needs to be capable of respond accurately

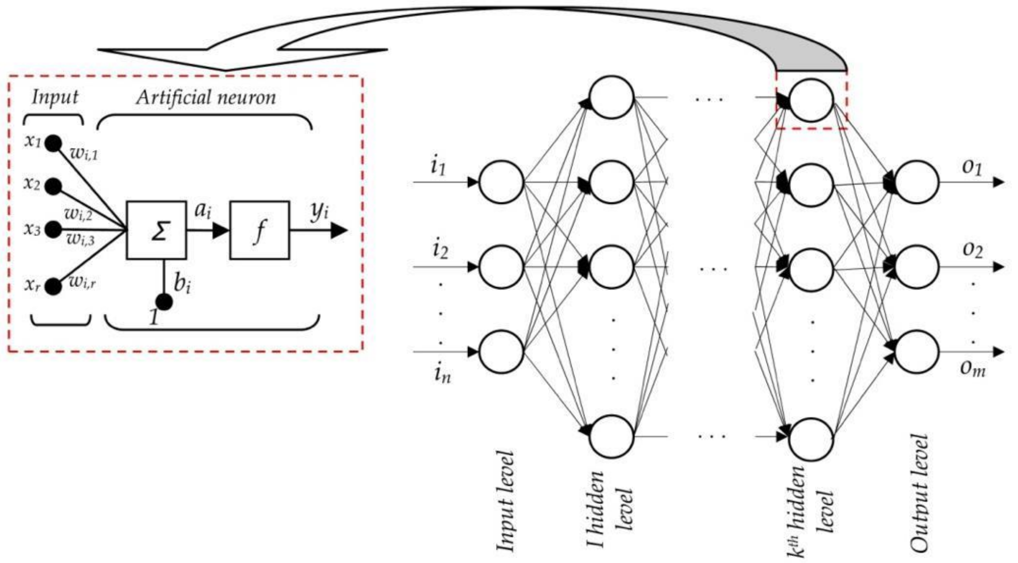


Figure 2.13: Connections in an artificial neuron with the backpropagation method

to examples never used on its development. If the network succeeds, the following step is the system implementation where the network is installed in a computer program or a hardware controller. After all these steps, the system is operating, but to continue on it's excellent level another step is required, this is called system maintenance and involves updating the developed system as changes in variables occur, this closes the development cycle shown in Figure 2.14.

The are common issues when developing and implementing a BP ANN, such as: database size and partitioning, data preprocessing, balancing and enrichment, data normalization, input and output representation, network weight initialization, learning rate (η) , momentum coefficient (μ), transfer function (σ), convergence criteria, number of training cycles, training modes, hidden layer size and parameter optimization [10] .

The database size problem happens because since the network must be capable of generalizing, the data should be sufficiently large so it covers all possible variations in the problem domain. The data partitioning, as said before, must be done into three subsets: training, test and validation. The training subset must include all the data from the problem domain and is used to adjust the connection weights between the nodes. The test subset must be filled with different data from the training subset, although both data must be inside the same domain. The validation subset must be constituted by examples different from the ones from the previous subsets so the best network can be chosen and it's accuracy can be evaluated.

Before beginning the network training a group of techniques must be done like noise removal, reducing input dimensionality and data transformation, treatment of non-normally distributed data, data inspection and deletion of outliers. All this processes constitute the data preprocessing. Balacing the data is particulary important in classification problems and is based on distributing the data evenly between the various clusters in order to prevent the network for being biased to one or more over-represented classes. Data enrichment

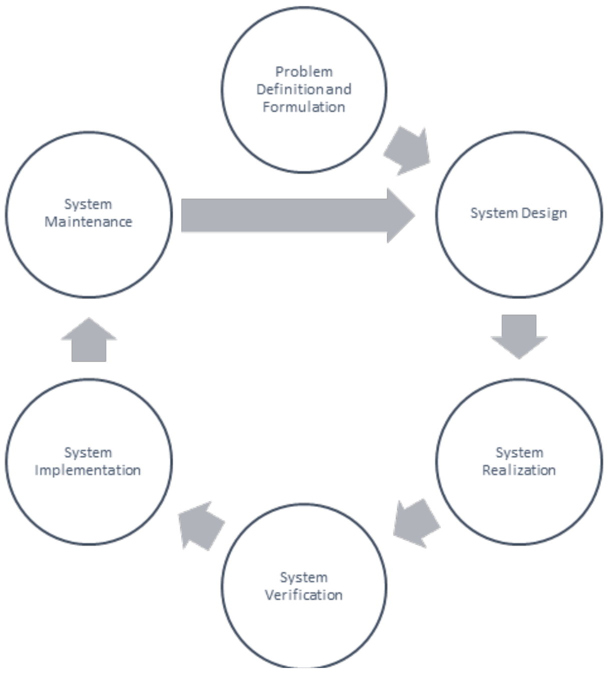


Figure 2.14: Development of an operational network cycle

consists in adding data to the domain so that the ANN robustness increases, this can be done by adding data or random noise data.

Data normalization is crucial for preventing larger numbers to override smaller numbers and to avoid premature saturation of hidden nodes which will restrict the network learning process. Input and output representation is based on the language used by the network, a binary terminology is very useful in extracting rules from a trained network and also increases the dimensionality of the vectors, and so the data discretization is more complete.

Network weight initialization it's still an undefined case that researchers disagree in it's relevance to the overall convergence and architecture of the network. On the other hand, the learning rate (η) is a well defined subject. If the learning rate is high the training process will be faster, the weight update will change significantly, this may cause oscillation on the error surface and the system may never converge, and also it increases the risk of overshooting a near optimal connection weight value for each node. Contrarily, if the learning rate is small it will drive steadily to the optimum solution, but in a very slow process. There is also a constant learning rate that is in between the two previous mentioned. In order to get the best of both characteristics, the adaptive learning rate [$\eta(t)$] was created, and as the name says, it will vary along the training process and adapt the velocity/step of the learning rate to the required at the moment, normally a bigger step is needed in the beginning of the training process and it should decrease when the process is closer to the final value.

The momentum coefficient (μ) is used in the weight updating to help the system escape local minima and reduce search instability, although it could lead to overshooting the solution when it's value is high [10]. In order to rectify that, in a similar way to the learning rate, a variable momentum coefficient [$\mu(t)$] as been created and it changes along the process to satisfy the system needs. The transfer function (σ) is necessary to transform the weighted sum of all signals onto a neuron so as to determine its firing intensity. The convergence criteria may differ for each network and can be based on the training error, on the gradient of error or on cross-validation, being the last one the more reliable but with the consequence of being more computational demanding and requiring abundant data from the network. The most used stop criterion is the sum-of-squared-errors (SSE) calculated as

$$SSE = \frac{1}{N}\sum_{p=1}^{N}\sum_{i=1}^{M}(t_{pi} - O_{pi})^2 \tag{2.11}$$

where O_{pi} is the actual solution, t_{pi} is the target solution, i represents the numeral of the output node, p is the numeral of the example, N is the number of total training examples and M is the number of output nodes. Normally the error decreases significantly with the increment of the number of nodes that constitute the hidden layer or with the growing number of training cycles, although when this parameters are oversized they can lead to network memorization and overtraining. The learning curves for this process can be seen in the Figure 2.15.

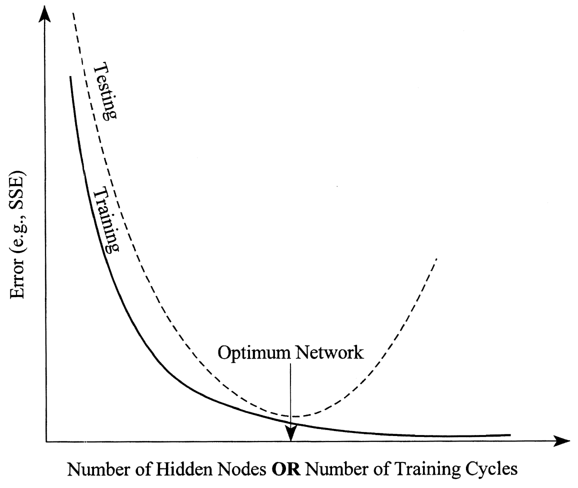


Figure 2.15: Learning curve of an artificial neural network

There isn't a concrete number of training cycles for every network, this must be achieved by trial and error, training a network for a long period of time could result in a over trained network that will only serve as a look-up table, this fenomenon is called overtraining or memorization, despite the fact that, theoretically speaking, a large number of training cycles would result in a near-zero error. Although hard to specify the determination of the appropriate number of hidden layers and nodes in each layer is one of the most crucial tasks in a ANN design. A network with few nodes will be incapable of differentiating between complex patterns and this will lead to a linear approximation of the actual trend. On the opposite side a network with to many nodes will result in overparameterization and follow the noise, this will cause a poor generalization of the training data and will require extensive processing time[10].

Training data is presented to the network and treated by one or both the following training modes: example-by-example training (EET) and batch training (BT). In EET mode, the connection weights are updated immediately after the processing of each training example, this leads to a small storage requirement and a better stochastic search. The disadvantage of this method is that training may become stuck in a bad training example wich may lead the network in the wrong direction. Otherwise, in the BT mode the weight update is done after all the training examples are processed by the network, this reflects in a better estimation of the error gradient vector and a more representative measurement of the required weight change, but it requires a larger network storage and is more likely to be stuck in a local minimum [12].

The concept of parameter optimization passes through the selection of the ideal values for each of the six quantifiable parameters mentioned previously. The number of hidden nodes if set to large will lead to overfitting and no generalization and set to small will lead to underfitting and slow training. The same analogy can be done to the learning rate (η), a high rate will lead to an unstable network that oscillates around the optimal solution and a low rate to a slow training. For the momentum coefficient (μ) when is set for a high value it will reduce the network risk of being stuck on local minima, speeds the training process but also increases the risk of overshooting the solution value which leads to instability, on the other hand, when this parameter is set for a low value it will contribute for a entrapment in local minima and slow training. A large number of training cycles will cause network memorization and bad generalization, the opposite situation will cause underrepresentation of data. The size of the training subset also affects it's performance, when using a large training subset the network will be capable of generalization but the opposite situation will result on the opposite effect. Also, regarding the test subset, a large size will give the network the ability to confirm it's generalization and a small test subset will restrict the network to do such self-evaluation.

An artificial neural network is an efficient and widely used computational tool due to his processing and learning capabilities, his increased utilization is linked to his ability to recognize and identify the underlying relations between data regardless explicit relation, nonlinearity and dimensionality. Its noise and corrupted data tolerance makes them versatile and attractive to numerous situations. This being said, it doesn't mean this method does not have it's limitations such as lack of clear rules or guidelines for an optimal architecture design, unclear explanation of the process through which the solution was obtained by the network, lack of physical concepts, and the most relevant factor, that ANN sucess is based on the quality and quantity of data used in the process.

2.5 Optimization (Genetic Algorithm)

2.5.1 Overview

Optimization can be defined as the selection of the best elements from a specific group with a set of constrains related to them. Select these individuals will lead to better results and then optimization will be achieved. This is a simple definition to a hard and vast job that sometimes cannot be accomplished. In mathematics, optimization can be seen as maximizing or minimizing a certain target function. In order to do so, rules must be satisfied otherwise the obtained solution will not be plausible.

A genetic algorithm (GA) is a heuristic search method that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offsprings of the next generation. This is an iterative process and the main goal is to achieve a generation with the fittest individuals, this is sustained on the fact that by passing the best genes the next generation, this generation will be better and have a greater chance at surviving.

The use of this method for solving computational problems is not new. It was introduced in 1970's by J. H. Holland when it has proved to be a significant development for scientific and engineering applications. Besides this method low speed processing and it's randomness, that leads to a problem of performance assurance, the research work in this field has grown exponentially due to the easy availability of low-cost fast processing computers and so this method has reached a stage of maturity. When applying this method, former complex and conflicting unsolved problems that require simultaneous solutions, can now be solved [40].

2.5.2 Basic concepts

A genetic algorithm cannot be considered as a mathematically guided algorithm, because it doesn't have any stringent mathematical formulation. This method is just a stochastic, discrete event and a nonlinear process and the optimal results obtained arise from the previous mentioned natural selection mechanism. In order to select such individuals a objective value must be introduced in such a way that a fitness function can be created to evaluate the performance of each individual, i.e., it's capability to compete against other individuals, and, prevail.

Four phases are considered in a genetic algorithm cycle: initial population, selection, crossover and mutation [9]. The process begins with a random set of individuals which is called initial population. Each individual represents a solution to the problem and is characterized by a set of parameters (variables) known as genes, a string of genes will form a chromosome (solution). In a genetic algorithm, the set of genes of an individual is usually represented using a binary string for reasons that will be explained later on this chapter.

The next phase is called selection and consists in the selection of the fittest individuals so they could pass their genes to the next generation. The pair of individuals (parents) are selected, as mentioned before, by the fitness function that will attribute a fitness score to each individual of the population.

After selection, comes the most significant phase in a genetic algorithm, the crossover. For each pair of parents to be mated, a crossover point or points is chosen randomly from within the genes, offsprings are created by crossing the genes of their parents among themselves until the crossover point is reached, and so, the new offspring are added to the population.

In some cases of the previous formed offspring, some of their genes can be subjected to a change in a process called mutation, that happens with a low random probability. This

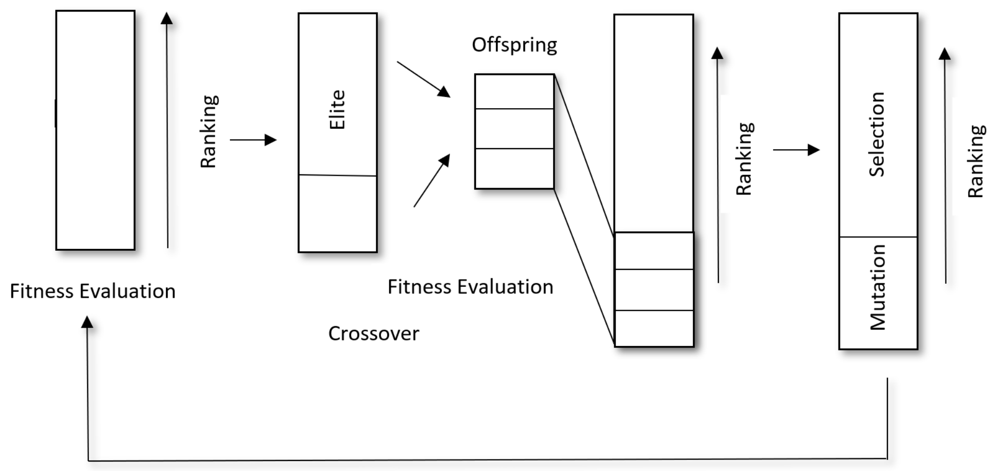


Figure 2.16: Schema of the genetic operators cycle

implies that some parameters values from the previous solution are modified. The main goal of mutation is to maintain diversity within the population and prevent premature convergence to a local minimum.

The algorithm comes to a final solution when the population has converged, i.e. the new offspring is not significantly different from the previous generation. Then it is said that the genetic algorithm has provided a set of solutions to the existing problem. But this is not the only way to stop the iterative cycle. It also can stop when a certain target value for the fitness function is reached or when a stipulated number of runs/iterations is finished.

2.5.3 Design and Construction

After establishing the GA cycle steps it's important to clarify a few criteria that need to be established. First thing to be discussed is the language utilized in this method to established the problem solutions. The bit string encoding is the classical and most commonly used approach used by GA researchers because of it's simplicity and traceability, although it could lead to difficult and sometimes unnatural solutions in some optimization problems [40].

Next, crossover and mutation rate criterion will define the probability of each operation to happen. The choice of such rate values is complex and nonlinear from one problem to another but an established guideline is that for a large population size (one hundred individuals) the crossover rate is smaller than a small population size (thirty individuals). This is given because the crossover operation tends to conserve the genetic information present in the strings and so it's capacity to generate new building blocks is small, although it increases the chance of disrupting some good chromosomes. On the other hand, the mutation rate in a large population is greater than in a small population size due to it's great capacity of generating new building blocks and it's non-conservative information operation, however it has the capacity of reintroduce some lost genetic material. Logically this happens because in a larger population the need to maintain the initial information is reduced when comparing with a smaller population, because when that happens to the smaller population there's a risk of losing crucial information to find the problems best suited solution.

The fitness function is the main link between the genetic algorithm and the operating system. It receives a chromosome and produces an objective value, normally in least square form, as a measure of it's performance. In order to maintain uniformity between the range of values over various problem domains, there are three common scaling operations: linear scaling, power law scaling and sigma truncation. In the linear scaling the fitness value of one chromosome has a linear relationship with the objective value:

$$f_i = a \cdot O_i + b \tag{2.12}$$

where i is the chromosome identification, a and b are problem constants used to enforce the equality of the average objective value O_i and the fitness value f_i and cause maximum scaled fitness to be a specified multiple of the averaged fitness. In the power law scaling the fitness value is obtained as the k^{th} power of the objective value:

$$f_i = O_i^k \tag{2.13}$$

where k is problem dependent or even varying during each run. In the sigma truncation, the fitness value is obtained by:

$$f_i = O_i - (\overline{O} - cx\sigma) \tag{2.14}$$

where c is a integer, $\overline{O}$ is the mean value of the objective values, and σ is the standard deviation in the population [40].

As mentioned before the probability of one chromosome be a parent depends on it's fitness score, and to measure that performance there are a few selection algorithms. Bias is a kind of algorithm that defines the absolute difference between actual and expected selection probabilities of each individual. Alternatively there's the spread algorithm that quantifies the range of possible number of trials that an individual may achieved. There is also efficiency that is related to the overall time complexity of the algorithm. One of the most used selection techniques is called roulette wheel selection. This mechanism starts with the sum of the fitness of all the population members (N), then a random number (n) between zero and N is generated and them the algorithm returns to the first population member whose fitness added to the fitness of the subsequent generations is equal or greater than n. This technique tends to give zero bias but could potentially use spread unlimitedly. Other commonly used selection technique is stochastic universal sampling (SUS) that has minimum spread, zero bias but as a low level of efficiency, that means that the time complexity is high. There are other methods that introduce a different approach to chromosome classification: instead of selecting a chromosome due to it's evaluation value the choice is based on their proportional rank, this is called ranking scheme. This has brought some developments in avoiding premature convergence and speeding up the process when the population is near the final convergence.

In the mutation phase it was said that this phenomenon could occur in one or more points of the chromosome genes. Although one-point crossover was stipulated because of natural processes, it has a major drawback because it's not possible to obtain certain combinations in some situations [40]. That's when the multi-point crossover as been introduced, this exponentially improved the generation of offspring. There's also uniform crossover, this approach uses a randomly generated crossover mask so it does not have a fixed number of crossover points. This method exchanges bits instead of segments so it can combine features regardless of their location. This skill can overcome the fact of destroying building blocks and potentially lose some important data. There is not a monochromatic view from researchers for which approach is better because it depends on each problem. The only well established conclusion is that one-point crossover is the least skilled method.

After creating the new generation is time to be decide what to do with the previous one. There are a few strategies that can be followed. First there's total replacement where every individual from the former generation is replaced with the newborns. Of course this can only happen when the number of offspring's generated by the algorithm is the same as the population before. This strategy is normally associated with a elitist strategy because of the possibility of the best chromosome/s failed to produce offspring and this way they can be implemented in the next generation. This strategy may induce the algorithm to increase the domination of a super chromosome but it undeniably increases the population performance. When the algorithm does not produce enough offspring's to completely substitute the previous generation, logically, only a percentage of them is going to be replaced. To choose which are going to be left out, normally the algorithm picks the worst ones but it can also substitute directly the parents for their respective offspring maintaining the diversity in the population.

As known one of the major applications of the GA is for time dependent problems. This means that the problem environment is changing throughout the process. In an effort to get the best solution, the algorithm must be adaptive to the system behaviour or it must be able to remain inviolable to such disturbance. To do so, there are two basic strategies: (1) expand the GA memory, so it can build up more data and consequentially more qualified responses; (2) combining three mechanisms called random immigrants, triggered hypermutation and statistical process control. This gives the algorithm more capacity for generate diversity in the population. The random immigrants replace some individuals from the population, the triggered hypermutation is an adaptive mechanism that increases the mutation rate whenever the best time-average performance of the population deteriorates and the statistical process controls the GA population performance so it can adapt to the nonstationary environment.

As quoted before one of the key negative points of GA method is the time spent in the computation process. With the aim of minimizing that effect and knowing that GA as already an intrinsic parallelism architecture, a parallel computational framework can be implemented. There are a few existing methods that can enhance the computational speed and they can be classified into three categories: global, migration and diffusion [40].

The global GA, represented in Figure 2.17, treats the entire population as only one breed and creates a master-slave relationship between the selection and fitness assignment (master) and the crossover, mutation and function evaluation (slave). The master can have has many slaves as needed to solve the problem, but one big disadvantage of this method is that the slaves embrace a lazy attitude while the master does his selective work.

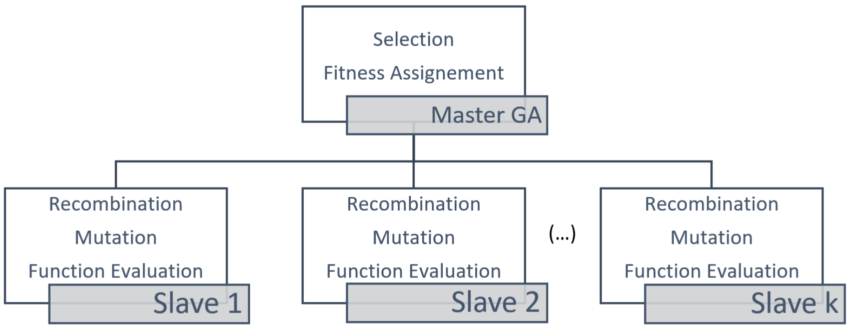


Figure 2.17: Schema of a global GA

Another approach is the migration GA, this method divides the population in subgroups which are treated as a different breeding from each other, then single individuals migrate to another subgroup in such a manner that good genetic material spreads through all population. There are three kinds of migration associated with this method: ring migration where individuals migrate according to a certain direction as represented in Figure 2.18; neighborhood migration where migration occurs to the nearest subgroups, Figure 2.19; and unrestricted migration where individuals can migrate to every existing group, 2.20.

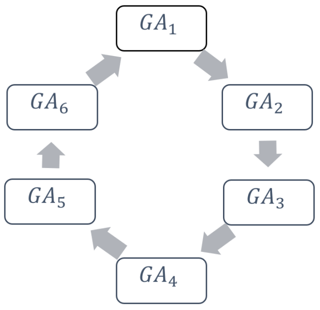


Figure 2.18: Schema of a ring migration GA

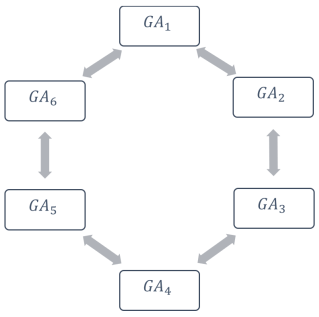


Figure 2.19: Schema of a neighborhood migration GA

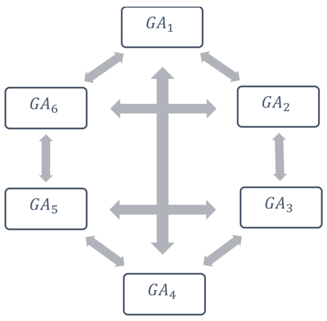


Figure 2.20: Schema of an unrestricted migration GA

Last there's diffusion GA, a method that considers the whole population as a single breed and scatters all individuals in a bi-dimensional network in which individuals are able to breed with other individuals in a small neighborhood (see Figure 2.21).

On the other hand one of the key positive points of GA method is it's capacity to solve multiobjective problems. The solution set of a multiobjective optimization problem consists not in a individual parameter optimization but on a group optimization, this means that not all parameters are going to be simultaneously and equally improved. This concept is knowm as the Pareto-optimality. A genetic algorithm as demonstrated it's capacity to obtain the Pareto-optimal set instead of a single parameter optimization.

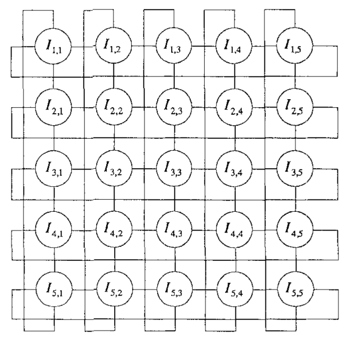


Figure 2.21: Schema of a diffusion GA

2.5.4 Advantages and shortcomings

After looking at all this methods and mechanisms adopted in a way to get the better and best suited genetic algorithm for every situation, is now reasonable to sum all it's positive aspects and give a series of reasons why this algorithm has become so popular and utilized:

- It can simplify problem constrains by traducing them into a chromosome coding language
- It's architecture can be updated and design to solve multiobjective problems the best way possible
- It can solve multimodal, nondifferientable, noncontinuous problems due to it's independence of the error surface
- It's a very simple and easy to understand technique
- It can be easily interfaced to existing models and simulations

As any other system it also has some negative factors. In some situations depending of the nature of the objective functions it can also generate bad chromosomes by combining good building blocks, this phenomenon is called deception [40]. Other negative episode is called genetic drift or bias, i.e. there are no guarantees of obtaining the global optimal solution, because the algorithm can be stuck in a sub-optimal point a prematurely converge towards a sub-optimal solution. As cited before, a genetic algorithm is not well suited for cases where response times must be guaranteed because of the it's large variance. Last but not least, it's important to consider this method randomness in such a way that the population performance will increase but that doesn't mean a specific individual performance will do too, because of such factor is not wise integrate a GA directly to a real system without utilizing a simulation model before.

2.5.5 Applications

Due to is characteristics this algorithm as been implemented in a wide range of applications in the area of industrial electronics, such applications are among parameter and system identification, control, robotics, pattern recognition, speech recognition, engineering designs, planning and scheduling and also system classification.

Regarding the work to be developed in this thesis, the application of GA enters the domain of engineering design. It can be used in the optimization of one object shaping, circuit layout and many other applications. Because an engineering design is similar to an art form, the GA can be used to strengthen this design and can be seen as a futuristic way for the creation of an object.

A typical problem is the optimization of an airfoil shape design, where the target pressure distribution is acquired by the Navier-Stokes equations and then GA is applied to optimize this value so that a critical airfoil shape is obtained. Experimentation has shown that the genetic algorithm approach yields better results than the ones obtained via established heuristics that embody extensive domain knowledge.

After looking at the outline features of a genetic algorithm since the problem formulation, genetic functionality of operators and some practical applications is reasonable to assume it's capacity for solving complex and conflicting problems, which makes this algorithm a powerful and useful computational tool. Although GA has some difficulty to react properly to changing environments due to it's convergence population factor, this could be mitigated by monitoring the algorithm with some intelligent supervisory scheme.

The application of GA with other emerging technologies such as neural networks and fuzzy logic systems could provide great results, leading to successful practical application of these intelligent system design, overcoming the performance of each technology by it self. This could be a challenge to implement but could bring some interesting and fruitful outcomes to innumerous areas of science.

Chapter 3

Modeling

In this chapter the author intends to give the reader an insight of how the problem was approached, what were the methods applied, the formulation and concepts relative to this study problem and how they are going to be solved in this work. In this chapter can be seen a description of: the Navier-Stokes Equations, the Finite Volume Method, the Drag Force, Turbulence and the Software Used.

3.1 Navier-Stokes Equations

3.1.1 Introduction

Fluid dynamics is a hoary problem that human kind have been trying to solve since the great Greek philosophers and scientists Aristotle (384-322 BC) and Archimedes (287-212 BC). The first partial differential equation, this means the first differential equation that contains unknown multi-variable functions and their partial derivatives, for fluid dynamics was much later formulated by Euler in 1752. In 1822, Claude Louis Marie Henri Navier, after adopting the Newton's definition of friction due to the velocity gradient and fluid viscosity, was able to include viscous forces into the previous equation and got the correct equation to what would become the Navier-Stokes equation. This equation has the name of two scientists because, although the equation was right, it's derivation was extremely flawed, that's when George Gabriel Stokes, twenty three years later (1845) re-derived the equation in a more delicate way. However their names do not enter the equation name, there were other three main scholars that had made great contribution in the years between Navier and Stokes work to establish the fluid dynamic equation, their names were Augustin-Louis Cauchy, Siméon Denis Poisson and Adhémar Jean Claude Barré de Saint-Venant [52].

The Navier-Stokes equation is the fundamental equation for governing fluid motion and dynamics, and until today numerous real world problems such as vehicle aerodynamics, meteorology, ocean and polution modeling and so on, have shown it's correctness and applicability. It is well known that seeking an analytical solution is too difficult and can only be obtained for some simple laminar flows, therefore, some simplifications or assumptions must be done, such asthe use of numerical methods, or solve simplified versions of these equations at defined points in space and time and simulate unresolved features of the fluid flow, such as turbulence, with parametrized approximations like averaging in a method called Reynolds averaging. This subject will be approached latter on this study. This difficulty is attributable to turbulence, a phenomenon frequently referred as one of the major unsolved problems of classical physics [15]. In 2000 the Navier-Stokes equation was selected to be one of the seven Millennium Problems by the Clay Mathematics Institute of Cambridge, U.S. and an astonishing award of one million dollars is given for the answer of each of the 7 millennium questions [28]. The answer to this problem must settle that three properties must be satisfied: a solution must exist, the solution must be always the same (unique solution) and must be a smooth solution, this means that a small change in one of the input parameters of the problem must result in a small change in the output solution. If these three properties are satisfied the equations would be fully discretized mathematically, this is already possible for a two-dimensional problem but not for a three dimensional problem [18]. The most recent development is this area was done by Terence Tao in the year of 2016, who demonstrated that for the averaged Navier-Stokes equations we get a infinite solution in finite time. The way he approached can be a way to show that is impossible to always obtain solutions in tri-dimensional in finite time [53].

3.1.2 Development

All hydrodynamic models are based upon the Newtonian law that acceleration is the result of imposed forces, also known as conservation of momentum, and on the conservation of mass, both fundamental laws of physics. If the flow is compressible the first law of thermodynamics must also be taken into account, that means that the energy is conserved [15]. Although the water is slightly compressible, it is often treated as incompressible when doing fluid flow calculations because the pressure changes involved are too small to make an appreciable change to it's density. The water at the bottom of the ocean is denser than at the surface due to the very large pressure there, it is also a little denser because it is colder at the bottom. Is also important to mention the existance of a thermocline, that simply consists in the transition layer between warmer mixed water at the ocean's surface and cooler deep water below. So that being said is easy to conclude that ocean water is not isothermal nor imcompressible, but in order to simplify the problem solution in this study water will be considered imcompressible and isothermic fluid.

In contrast with a solid body in a translational motion, when a fluid moves the velocity of each particle may be different at each location of the fluid. Due to that fact, in order to apply the fundamental physical principles, one of the following two discretization methods must be applied: finite control volume or infinitesimal fluid element. The fluid flow equations obtained directly from the finite control volume are in the integral form but they can be manipulated to be obtained in the partial differential form and vice-versa. In both scenarios the equations obtained are in the so called conservation form of the governing equations if the finite control volume or infinitesimal fluid element is fixed, however if they are moving with the flow, the equations obtained are in the non-conservation form [29]. In this study the first method was used, the finite control volume method, but in order to get a better comprehension of this equations both formulations were described.

Historically the momentum equations for a viscous flow were identified as the Navier-Stokes equations, however in modern Computer Fluid Dynamics (CFD) literature, this terminology has been expanded to include the entire system of equations, that means the equation for conservation of mass, momentum and energy. As mentioned before, the fluid is considered isothermic so the conservation of energy can be neglected.

Conservation of mass

Considering a finite control volume fixed in space as shown in Figure 3.1, the flow velocity at a point on the control surface is $\vec{V}$ and the vector surface area is $d\vec{S}$. An elemental volume inside the finite volume control is represented as $d\vartheta$. Applying the fundamental physical principal that mass is conserved is the same thing that saying that the net mass flow out of control volume through surface S is equal to the time rate of decrease of mass inside the control volume.

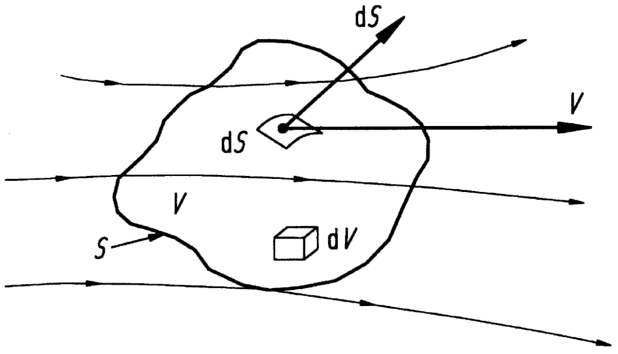


Figure 3.1: Finite control volume fixed in space

The mass flow of a moving fluid across the fixed surface can be obtained by calculating the product of density times the area of the surface times the component of velocity perpendicular to the surface, so

$$\rho V_n dS = \rho\vec{V}\cdot\vec{dS} \tag{3.1}$$

The net mass flow of the entire control volume through the control surface S is given by the summation over S of each of the infinetesimal elemental mass flows represented in the previous Figure 3.1. In the limite this becomes a integral and can be represented as

$$\oiint_S \rho\vec{V}\cdot\vec{dS} \tag{3.2}$$

this is equivalent to the left side of the equation and represents the net mass flow out of control volume through surface S. Now considering the time rate of decrease of mass inside the control volume, the right side of the equation is given by

$$-\frac{\partial}{\partial t}\oiiint_\vartheta \rho d\vartheta \tag{3.3}$$

then substituting both terms in the previous statement that net massflow out of control volume through surface S is equal to the time rate of decrease of mass inside the control volume, results in

$$\oiint_S \rho\vec{V}\cdot\vec{dS} = -\frac{\partial}{\partial t}\oiiint_\vartheta \rho d\vartheta \tag{3.4}$$

which is equivalent to

$$\frac{\partial}{\partial t}\oiiint_\vartheta \rho d\vartheta + \oiint_S \rho\vec{V}\cdot\vec{dS} = 0 \tag{3.5}$$

The expression obtained is the integral and conservative form of the continuity equation. After obtaining this equation is possible to manipulate it in order to get the differential form. Since the control volume is fixed in space the limits of integration for the integrals are constant, that means that the time derivative $\frac{\partial}{\partial t}$ can be placed inside the integral, resulting in

$$\oiiint_{\vartheta} \frac{\partial \rho}{\partial t} d\vartheta + \oiint_{S} \rho \vec{V} \cdot \vec{dS} = 0 \tag{3.6}$$

By applying the divergence theorem from vector calculus, the surface integral can be expressed as a volume integral

$$\oiint_{S} (\rho \vec{V}) \cdot \vec{dS} = \oiiint_{\vartheta} \nabla \cdot (\rho \vec{V}) d\vartheta \tag{3.7}$$

Substituting Equation (3.7) in Equation (3.6)

$$\oiiint_{\vartheta} \frac{\partial \rho}{\partial t} d\vartheta + \oiiint_{\vartheta} \nabla \cdot (\rho \vec{V}) d\vartheta = 0 \tag{3.8}$$

or,

$$\oiiint_{\vartheta} \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) \right] d\vartheta = 0 \tag{3.9}$$

Since the finite control volume can be arbitrarily placed the only way for the integral to be equal to zero is for the integrand to be zero at every single point within the control volume, that results in the following equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) = 0 \tag{3.10}$$

which represents the continuity equation in the differential and conservation form.

It's important to refer that both equations are statements of the mass conservation and also that both are written in the conservation form because initially was assumed that the finite control volume was fixed in space. If the control volume was moving along with the fluid the equations obtained directly would be written in the non-conservation form. Both are equally valid and one can be obtained easily from the other [29]. Consider the vector identity involving the divergence of the product of a scalar times a vector such as

$$\nabla \cdot (\rho \vec{V}) \equiv \rho \nabla \cdot \vec{V} + \vec{V} \cdot \nabla \rho \tag{3.11}$$

substituting Equation (3.11) in the Equation (3.10)

$$\frac{\partial \rho}{\partial t} + \vec{V} \cdot \nabla \rho + \rho \nabla \cdot \vec{V} = 0 \tag{3.12}$$

the first two terms of the equation are just the substantial derivative of density, so

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \vec{V} = 0 \tag{3.13}$$

which represents the non-conservation form of the continuity equation. It's important to notice that the difference between the two different forms of expressing the equation is very small in most of the theoretical aerodynamics, however it can make a considerable difference in some CFD applications and that's the main reason why both formulations are represented in this work [29].

Conservation of momentum

In this section another fundamental physical principle is applied to a model of the flow, the so called Newton's second law that says that force actuating on a body is equal to the product of it's mass times the acceleration

$$\vec{F} = m\vec{a} \tag{3.14}$$

This time let's start with the non-conservation form of the equation, to directly obtain it, it's necessary to assume that the finite control volume or the infinitesimal fluid element are moving with the flow. Since on the previous deduction of the equation of conservation of mass we started by using the finite control volume, this time the infinitesimal fluid element was used in order to prove that one can be obtained after another. The infinitesimal fluid element is an infinitesimally small fluid element in the flow with a differential volume, $\mathrm{d}V$, however it is large enough to contain a huge number of molecules so that it can be viewed as a continuous medium [29].

When we apply Newton's second law to a infinitesimal fluid element moving with the flow, it's reasonable to say that the net force in on the fluid element equals it's mass times the acceleration of the element. This relation can be split apart into three components x, y and z - axes. In this study only two dimensions were considered, so only the x and y-axes are relevant, however the equations are described for the three axes. If considering the forces actuating only on the x-component, the equation becomes

$$F_x = ma_x \tag{3.15}$$

where F_x and a_x are the scalar x-components of the force and acceleration, respectively.

The main sources of the force actuating on the fluid element are body forces and surface forces. The body forces are the ones which act directly on the volumetric mass of the fluid element such as gravitational, electric and magnetic forces. Surface forces are the ones that act directly on the surface of the fluid element, they are caused by the pressure distribution acting on the surface imposed by the fluid surrounding the fluid element and to the shear and normal stress distribution acting on the surface, also imposed by the outside fluid by means of friction.

The shear and normal stresses in a fluid are related with the time-rate-of-change of the deformation of the fluid element. As can be seen in Figure 3.2 the shear stress is denoted by τ_{xy} and is related with the time-rate-of-change of the shearing deformation of the fluid element, whereas the normal stress, represented in Figure 3.3 is related to the time-rate-of-change of volume of the fluid element and is denoted by τ_{xx}. Both of these stresses are dependent of the velocity gradients of the flow, however for most viscous flows the normal stresses are much smaller than shear stresses and are normally dilapidated, except in flows where the normal velocity gradients are very large, such as inside a shock wave [29].

The surface forces in the x-direction applied in the fluid element are sketched in Figure 3.4. The formulation used means that τ_{ij} denotes a stress in the j-direction and exerted on the perpendicular plane of the i-axis.

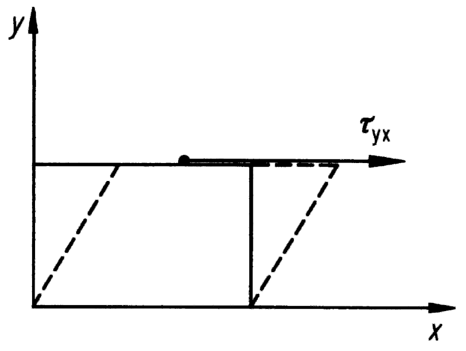


Figure 3.2: Shear stress on an infinitesimal fluid element

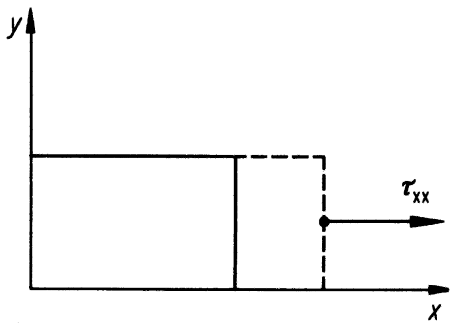


Figure 3.3: Normal stress on an infinitesimal fluid element

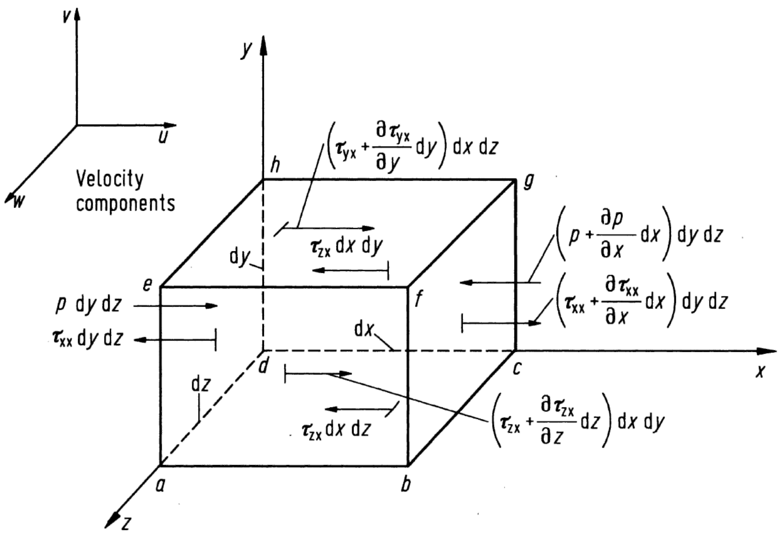

v
u
w
Velocity components
y
$\left(\tau_{yx} + \frac{\partial \tau_{yx}}{\partial y} dy\right) dx\, dz$
h
g
$\left(p + \frac{\partial p}{\partial x} dx\right) dy\, dz$
dy
$\tau_{zx}\, dx\, dy$
e
f
$\left(\tau_{xx} + \frac{\partial \tau_{xx}}{\partial x} dx\right) dy\, dz$
$p\, dy\, dz$
$\tau_{xx}\, dy\, dz$
dx
dz
d
c
x
$\left(\tau_{zx} + \frac{\partial \tau_{zx}}{\partial z} dz\right) dx\, dy$
$\tau_{zx}\, dx\, dz$
a
b
z

Figure 3.4: x-direction forces actuating on a moving infinitesimal fluid element

After analysing the Figure 3.4 above it's possible to comprehend that the net surface force in the x-direction is given by

$$\begin{aligned}&\left[p - \left(p + \frac{\partial p}{\partial x} dx\right)\right] dydz + \left[\left(\tau_{xx} + \frac{\partial \tau_{xx}}{\partial x} dx\right) - \tau_{xx}\right] dydz \\ +&\left[\left(\tau_{yx} + \frac{\partial \tau_{yx}}{\partial x} dy\right) - \tau_{yx}\right] dxdz + \left[\left(\tau_{zx} + \frac{\partial \tau_{zx}}{\partial z} dz\right) - \tau_{zx}\right] dxdy\end{aligned} \tag{3.16}$$

and also that the body force per unit of mass acting on the fluid element by $\vec{F}$ is given by f_x on the x-component and the volume of the infinitesimal fluid element is given by $(d_x d_y d_z)$, which results in the body force on the infinitesimal fluid element acting in the x-direction be equal to

$$\rho f_x (d_x d_y d_z) \tag{3.17}$$

combining Equations (3.16) and (3.17), after adding and cancelling terms, it's possible to obtain the Equation (3.18) that represents the total force in the x-direction, F_x

$$F_x = \left(-\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} \right) dxdydz + \rho f_x dxdydz \tag{3.18}$$

which represents the left side of Equation (3.14). Recalling that the mass of the infinitesimal fluid element is given by

$$m = \rho dxdydz \tag{3.19}$$

and also that the acceleration on the x-direction (a_x) of the infinitesimal fluid element is the time-rate-of-change of it's velocity in the x-direction (u), that can be obtained by applying the substantial derivative, thus:

$$a_x = \frac{Du}{Dt} \tag{3.20}$$

Combining Equations (3.14), (3.18), (3.19) and (3.20) it's possible to obtain the x-component of the momentum equation for a viscous flow

$$\rho \frac{Du}{Dt} = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x \tag{3.21}$$

In a similar way it's possible to obtain the y and z-components

$$\rho \frac{Dv}{Dt} = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y \tag{3.22}$$

$$\rho \frac{Dw}{Dt} = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z \tag{3.23}$$

These equations are written in the partial differential form and were obtained directly by the application of the fundamental physical principle, Newton's second law to a infinitesimal fluid element which results in the acquisition of the non-conservation form of these equations. Note that is possible to obtain them in the conservation form after some manipulation.

Rewriting the left side of Equation (3.21) in terms of the definition of the substantial derivative

$$\rho \frac{Du}{Dt} = \rho \frac{\partial u}{\partial t} + \rho \vec{V} \cdot \nabla u \tag{3.24}$$

and expanding the following derivative, it results in

$$\rho \frac{\partial u}{\partial t} = \frac{\partial (\rho u)}{\partial t} - \frac{\partial \rho}{\partial t} \tag{3.25}$$

after the application of the vector identity for the divergence of the product of a scalar times a vector, the result is

$$\rho \vec{V} \cdot \nabla u = \nabla \cdot (\rho u \vec{V}) - u \nabla \cdot (\rho \vec{V}) \tag{3.26}$$

substituting Equations (3.25) and (3.26) on the Equation (3.24), it's possible to obtain

$$\rho \frac{Du}{Dt} = \frac{\partial (\rho u)}{\partial t} - u \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{V}) \right] + \nabla \cdot (\rho u \vec{V}) \tag{3.27}$$

On the Equation (3.27) above, the term in brackets is no more than the left side of the continuity equation (Equation (3.10)), which means that it's value is equal to zero

$$\rho\frac{Du}{Dt} = \frac{\partial(\rho u)}{\partial t} + \nabla\cdot(\rho u\vec{V}) \tag{3.28}$$

Substituting Equation (3.28) into Equation (3.21)

$$\frac{\partial(\rho u)}{\partial t} + \nabla\cdot(\rho u\vec{V}) = -\frac{\partial p}{\partial x} + \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{yx}}{\partial y} + \frac{\partial \tau_{zx}}{\partial z} + \rho f_x \tag{3.29}$$

which represents the conservation form of the momentum equation on the x-component. In a similar way the equations for the y and z-components can be obtained

$$\frac{\partial(\rho v)}{\partial t} + \nabla\cdot(\rho v\vec{V}) = -\frac{\partial p}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{zy}}{\partial z} + \rho f_y \tag{3.30}$$

$$\frac{\partial(\rho w)}{\partial t} + \nabla\cdot(\rho w\vec{V}) = -\frac{\partial p}{\partial z} + \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} + \rho f_z \tag{3.31}$$

At the end of the seventeenth century Isaac Newton stated that shear stress in a fluid is proportional to the velocity gradients. This statement would give a group of fluids the name of Newtonian fluids, examples are air and water. Non-Newtonian fluids are the ones which the shear stress is not proportional to the velocity gradients such as blood. Theoretically, in all practical aerodynamic problems, the fluid can be treated as a Newtonian fluid. For such fluids, in the year of 1845, George Gabriel Stokes stated that

$$\tau_{xx} = \lambda\nabla\cdot\vec{V} + 2\mu\frac{\partial u}{\partial x} \tag{3.32}$$

$$\tau_{yy} = \lambda\nabla\cdot\vec{V} + 2\mu\frac{\partial v}{\partial y} \tag{3.33}$$

$$\tau_{zz} = \lambda\nabla\cdot\vec{V} + 2\mu\frac{\partial w}{\partial z} \tag{3.34}$$

$$\tau_{xy} = \tau_{yx} = \mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \tag{3.35}$$

$$\tau_{xz} = \tau_{zx} = \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \tag{3.36}$$

$$\tau_{yz} = \tau_{zy} = \mu\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right) \tag{3.37}$$

where μ is the molecular viscosity and λ is the bulk viscosity coefficient. By applying the Stokes statement to the previous conservation form of the momentum Equation (3.29), (3.30) and (3.31) for the x, y and z-components, respectively, it's possible to obtain

$$\begin{aligned} &\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} + \frac{(\rho uw)}{\partial z} \\ &= -\frac{\partial p}{\partial x} + \frac{\partial}{\partial x}\left(\lambda + 2\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)\right] + \rho f_x \end{aligned} \tag{3.38}$$

$$\begin{aligned}&\frac{\partial(\rho v)}{\partial t}+\frac{\partial(\rho v^2)}{\partial x}+\frac{\partial(\rho uv)}{\partial x}+\frac{(\rho uw)}{\partial z}\\=-\frac{\partial p}{\partial y}+\frac{\partial}{\partial y}\left(\lambda+2\mu\frac{\partial v}{\partial y}\right)+\frac{\partial}{\partial x}\left[\mu\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)\right]&+\frac{\partial}{\partial z}\left[\mu\left(\frac{\partial w}{\partial y}+\frac{\partial v}{\partial z}\right)\right]+\rho f_y\end{aligned}\tag{3.39}$$

$$\begin{aligned}&\frac{\partial(\rho w)}{\partial t}+\frac{\partial(\rho w^2)}{\partial z}+\frac{\partial(\rho uw)}{\partial x}+\frac{(\rho vw)}{\partial y}\\=-\frac{\partial p}{\partial z}+\frac{\partial}{\partial z}\left(\lambda+2\mu\frac{\partial w}{\partial z}\right)+\frac{\partial}{\partial x}\left[\mu\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial x}\right)\right]&+\frac{\partial}{\partial y}\left[\mu\left(\frac{\partial w}{\partial y}+\frac{\partial v}{\partial z}\right)\right]+\rho f_z\end{aligned}\tag{3.40}$$

In other words the Navier-Stokes equations are a coupled system of non-linear partial differential equations that until today there isn't a general closed-form solution, however they are widely used in a long range of practical applications. Their output results are largely influenced by the initial conditions and boundary conditions stipulated for the problem specially when working with CFD solutions. In a CFD simulation the results tend to be generally smooth and stable when the conservation form of this equations is utilized, however when the non-conservation form is used for a shock-capturing solution, the computed flow-field usually exhibit unsatisfactory a spatial oscillations upstream and downstream of the shock wave which can provide unstable solutions [29]. The concept of turbulence is really important to understand the problem with Navier-Stokes equations problem because fluids have a turbulent motion and the existing computational power always average the solutions by eliminating this phenomenon from the problem.

3.2 Finite Volume Method

3.2.1 Introduction

The start point of any numerical method is the mathematical model, such as in this study case, a group of partial differential equations and boundary conditions that define the problem. Selecting an appropriate model is very important, because some simplifications and relaxation of certain restrictions could lead to crucial mistakes. Commonly a numerical method is created to find an approximate solution to a particular group of equations, since that is impossible to create a general numerical method capable of solving all problems. After selecting a fitting mathematical method, it's necessary to choose a proper discretization method. A discretization method is a way of approximating the differential equations by a system of algebraic equations for the problem variables, that are going to be obtained in discrete locations in time and space. There are several discretization methods, however the most widely used are: Finite Difference Method (FDM), Finite Volume Method (FVM) and Finite Element Method (FEM) [20].

The finite volume method, in similarity with other numerical methods developed for the simulation of fluid flow, transforms a set of partial differential equations into a system of linear algebraic equations. However, the discretization procedure is distinctive from the other methods and involves two basic steps: first, the partial differential equations are integrated and transformed into balance equations over an element. The result is a set of semi-discretized equations. Second, interpolation profiles are chosen to approximate the variation of the variables within the element and relate the surface values of the variables to their cell values and thus transform the algebraic relations into algebraic equations [43].

The popularity of the FVM in computer fluid dynamics is due to the high flexibility it offers as a discretization method, its prominent role in the simulation of fluid flow problems urges as a result of the work done by the CFD group at Imperial College in the early seventies under the direction of Professor Spalding, with contributors such as Patankar, Gosman and Runchal [43]. The FVM flexibility and popularity is based on the fact that discretization is carried out directly in the physical space, thus there's no need for any transformation between the physical and the computational coordinate system. Another crucial characteristic of the FVM is that its numerics mirror the physics and the conservation principles it models, such as the integral property of the governing equations and the attributes of the terms it discretizes. Later its adoption of a collocated or non-staggered variable arrangement, where all dependent variables share the same control volumes, turned it capable of solving flows in complex geometries [30]. These improvements have provided the capacity to use the FVM in a wide range of applications while retaining the simplicity of its mathematical formulation.

3.2.2 Development

Considering a general flow field represented by streamlines, it's possible to insert a closed volume within a finite region of the flow, this defines a control volume V, and a control surface S, defined by the closed surface that bounds the volume. The control volume can be fixed in space, this means that the fluid moves through it (Figure 3.5) or the control volume can be moving with the fluid, which means that the volume particles inside the control volume are always the same (Figure 3.6). In both scenarios the control volume is a sufficiently large finite region of the flow and the fundamental physical principles are applied to the fluid inside the control volume in the second scenario and to the fluid crossing through the control surface in the first scenario. The main reason of applying this method is that by implementing this control volume our attention is redirected to a finite region instead of the whole fluid field, which simplifies the process [29].

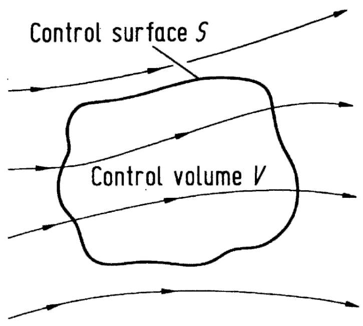


Figure 3.5: Fixed finite control volume

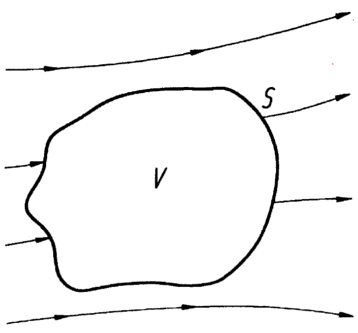


Figure 3.6: Moving finite control volume

As mentioned previously in this chapter the finite volume method utilizes directly the integral form of the conservation equations, therefore the basic quantities such as mass and momentum will be conserved at the discrete level. In the center of each control volume there's a computational node, as can be seen in Figure 3.7, where the variable values are calculated, the variable values in the control volume surface are obtained by interpolation in relation to the node value. The surface and volume integrals are approximated utilizing appropriate quadrature formulation, as a result, an algebraic equation is obtained for each control volume such as the values for the node and respective neighbor nodes.

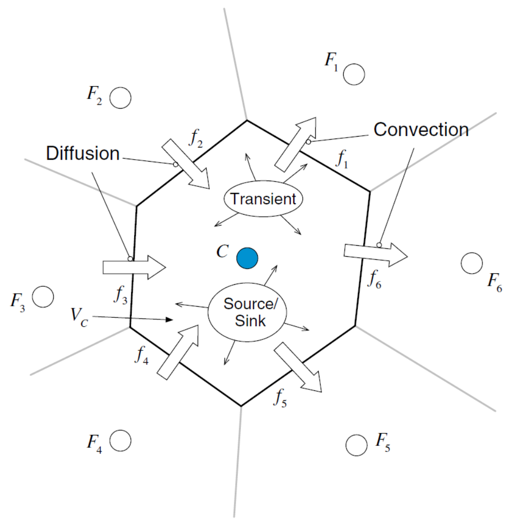


Figure 3.7: Representation of the finite control volume and interaction with the respective neighbors

This method can be applied to any kind of computational mesh because of it's capacity of adapting to complex geometries, the mesh only defines the control volume frontiers and it does not need to be relationed to the system coordinates. This method is inherent conservative since the surface integrals, that represent convective and diffuse fluxes, are the same in shared faces of control volumes.

In the first step of the discretization process the governing equations are integrated over the finite volumes into which the domain as been divided, then the Gauss theorem is applied to the volume integrals of the convection and diffusion terms into surface integrals, after this procedure the surface and volume integrals are transformed into discrete ones and integrated numerically through the use of integration points. The objective of the second phase of the discretization process is to transform the equations into algebraic equations by expressing the face and volume fluxes in terms of the values of the variable at the neighboring cell centers. In short, the first step consists in the discretization of the solution domain and the second in the discretization of the problem equations and also as a linearization of the fluxes [30].

After achieving the semi-discretized equation, it's important to set the boundary conditions. In a way to evaluate the fluxes at the faces of a domain boundary, generally it's not required to have a profile assumption, so usually a direct substitution is performed.

Notwithstanding the wide range of boundary conditions, the two most broadly used are the Dirichlet and the Neumann boundary conditions, which, in mathematical terms, represent a value specified and a flux specified boundary conditions, respectively. The order of accuracy of the discretization procedure is based on the fact that fluxes at the faces and sources over the element are evaluated following the mean value approach and also that the value of the variables are stored in the center of the cells and it's assumed that their variation is linear across the cells (center to surface) [43].

In an effort to ensure a meaningful solution field, the discretized equations must possess a set of properties such as: conservation, accuracy, convergence, consistency, stability, economy, transportiveness and boundedness of the interpolation profile. The reason to this matter is that when the size of the mesh element tends to zero, the numerical solution is expected to be the exact solution of the equation utilized, independently of the interpolation profile used to evaluate the element variable values, however since the finite volume method is used the following properties become crucial.

- Conservation

From a physical point of view, is fundamental that the transported variables are conserved in the discretized solution domain, such as in the initial domain, since that generally, these are conservative quantities such as mass, energy, momentum, etc. Otherwise the results obtained may be unrealistic. In the FVM this property is ensured because the fluxes integrated at an element face are based on the values of the elements sharing the face, so for any surface common to two elements, the flux leaving the face of one element will be equal to the flux entering the other element through that same face, this means that these fluxes have equal magnitude but opposite signs. Thus, the FVM can be considered a conservative method since it possesses this property [43].

- Accuracy

Accuracy can be defined as how close a numerical solution is to the exact solution, however in some cases the exact solution for the problem is unknown, consequently a direct comparison to check accuracy is not possible. An alternative way is to consider the truncation error as a measure of accuracy. In the FVM discretization process the error associated with the first step has a second order accuracy, this means that if the number of grid points is doubled then the discretization error will be reduced by a factor of four. The truncation error of a discretization scheme is the largest truncation error of each of the individual terms in the discretized equation. The discretization error does not give the value of the error on a certain grid, however, it shows how fast the error will decrease with grid refinement, which means the higher the order of the error the faster it will decrease with mesh refinement.

- Convergence

Due to the fact of dealing with nonlinear conservation equations an iterative process is required. This approach consists in repeatedly applying a solution algorithm, after starting with an initial guess, with the solution at the end of an iteration as the initial guess of the next iteration. Theoretically, a solution has converged when it's value does not change from the previous iteration to the current iteration, however this may be very hard to get and may consume a lot of time, so in practice is said that the solution has converged when the change of value is smaller than a specified quantity. Generally, the therm convergence is used to indicate the obtainment of a solution with any method, but it can also be used to indicate the achievement of a grid independent solution in a CFD case, this means a solution that does not change despite any further grid refinement.

- Consistency

A solution to an algebraic equation approximating a partial differential equation is said to be consistent if, at each point in the solution domain, the numerical solution approaches the exact solution of the partial differential equation as the time step and grid spacing tend to zero, so, the discretization error also tends to zero. If this properties are not satisfied, consistency can not be assured.

- Stability

The concept of stability is allusive to the discretized equations behavior when solved by an iterative solver, it indicates if the set of algebraic equations can be solved under a variety of initial and boundary conditions. In this context the stability it's not regarding the process but respective to the resulting system of equations. A sufficient condition for a system of linear equations to be stable and converge to a solution is for it to satisfy the Scarborough criterion, which means that its matrix of coefficients is diagonally dominant. For transient problems the use of explicit or implicit transient schemes has direct impact on the stability of the numerical method.

- Economy

When developing and applying a CFD code it's important to have in mind the costs involved in the computational time and requirements needed to solve the problem, this process could be time and monetary expensive but this properties must not be prohibitive of its appliance.

- Transportiveness

The directional properties exhibited by fluid transport are well known and are signaled by the change in the type of the transport scalar equation. When applying the FVM there are some implications that can be seen in Figure 3.8.

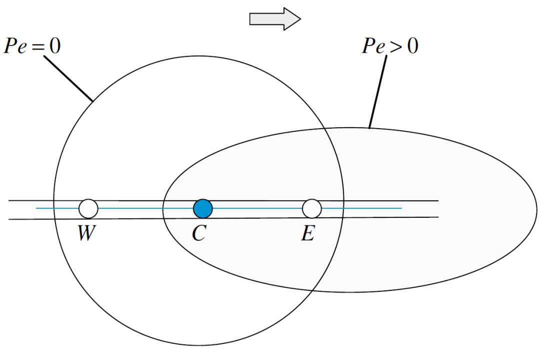


Figure 3.8: Representation of the fluid flow transportability property

If a constant source of the scalar variable exists within the element C in a flow field with uniform diffusivity and velocity, then the contours of the constant scalar variable will be influenced by the ratio between the convection and diffusion phenomenon. The Péclet number (Pe) shows this relation and can be written as

$$Pe = \frac{Convection\ strength}{Diffusion\ strength} = \frac{\rho u}{\Gamma/\Delta x} \tag{3.41}$$

When Equation (3.41) value is equal to zero, that means that the transport is ruled by diffusion and the isolines of the scalar variable are circular and the value of the scalar variable at point C is influenced by the surrounding nodes W and E. Conversely, when the convection effects increase, the circular contours become elliptical and the region influencing the value of the scalar variable shifts in the direction of the flow, accordingly for high value of Pe flows, the element C will have weak or nonexistent influence on upstream nodes (W), while downstream nodes (E) will be strongly affected. When this property is not verified it could lead to unstable and nonphysical solutions in the selected discretization schemes.

- Boundedness of the interpolation profile

Ensuring conservation does not guarantee that other important properties of the original partial differential equation will be maintained after the realization of the discretization method. This can be accomplished by controlling the discretization schemes an their linearization.

Another feature to have in mind is the variable arrangement, that defines the location where the variables and their respective values are stored. There are two possible arrangements, the cell-centered arrangement (Figure 3.9 (a)) and the vertex-centered arrangement (Figure 3.9 (b)).

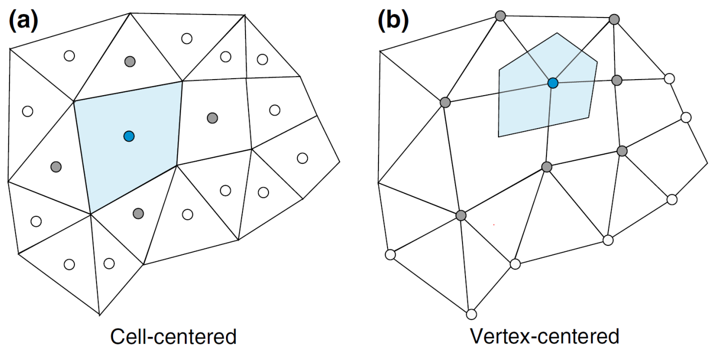


Figure 3.9: Representation of the cell-centered (a) and vertex-centered (b) arrangements

In the vertex-centered arrangement the flow variables are stored at the vertices of the elements while elements are integrated around the variable location. When applying this concept a cell can be created around a grid point in several ways. In a two dimensional approach, one way is to connect the centroids of the cells having the grid point in common or centroids can be joined of the surrounding elements to the centroids of their faces. This type of arrangement allows for an explicit profile to be defined over the elements in terms of the vertex variables and also permits an accurate resolution of face fluxes for all mesh topologies. However it has some shortcomings such as yield of a lower order accuracy of element-based integrations (since the vertex is not necessarily in the center of the centroid), it increases the storage requirements, due to the creation of larger matrix, handling of boundary conditions requires additional treatment, the mesh must be based on a set of element types for which a shape function can be defined, also, additional complications may arise at sharp edges and branch cuts.

On the other hand there's the cell-centered variable arrangement, that currently is the most popular type of variable arrangement used in FVM. By applying this practice the variables are stored at the centroids of grid cells or elements. Therefore, the elements are identical to the discretization elements and the method has second order accuracy, since all quantities are computed at element and face centroids. Another advantages of the cell-centered formulation are it's allowance to use general polygonal elements with no need for pre-defined shape functions, this enables a straightfoward implementation of a full multigrid strategy, variations within the cell can be re-constructed using Taylor series expansion. Withal, as any other method it has a some disadvantages, two of the most important are its treatment of non-conjunctional elements and the manner the diffusion term is discretized on non-orthogonal cells, these influence the method accuracy and robustness, respectively. Both are affected by the mesh quality and so depends the discretization error, so it's possible to conclude that this method as a strong dependence on the mesh construction. In accordance with this fact, for a sufficiently smooth grid, the cell-centered arrangement can attain accuracy of order two or higher. Another problem of this arrangement is the treatment of non-orthogonality in the discretization of the diffusion term.

To recap, the vertex-centered scheme and the cell-centered scheme are numerically very similar, in the interior of a domain, for steady state calculations. The only situation where the performance of the vertex-centered scheme is superior to the cell-centered scheme is over a distorted grid, however in all other situations, it is more advantageous to use the cell-centered arrangement, as it leads to a more straightforward implementation in a computer code [43].

As mentioned earlier on this chapter, there are explicit and implicit numerical methods. The first method is one in which the dependent variables are computed directly via already know values, thus any discretization operator can be directly evaluated based on the variable values. Contrastingly, in a implicit method the variables are treated as unknows and assembled to form a coupled set of equations which are treated as unknowns and assembled to form a coupled set of equations which are then solved via numerical tools using either a direct or an iterative solution algorithm. In CFD the conservation equations dealt are nonlinear, therefore the implicit approach is most often chosen for solving them.

This method is widely used by engineers because of it's simplicity and because every term that needs approximation has physical meaning [20]. But as all methods it has some disadvantages, when comparing the FVM with the FDM, it's possible to conclude that is harder to apply methods with order higher than second order to develop tri-dimensional non-structured meshes, this is a result of the FVM need for interpolation, differentiation and integration [24].

After going through this method description, it's possible to understand it's guiding principles for the discretization process, which guarantee that the resulting discretized equation will possess the desirable attributes, in such a way that the desiring results would be obtained and this tool could be used in several applications.

3.3 Drag Coefficient (C_D)

3.3.1 Introduction

Drag is the heart of hydrodynamic design. The accurate assessment of drag characteristics and values is essential for engineering design in order to get the best and more economic vehicle. Determination of vehicle drag coefficient as been one of the most interesting topics in the last fifty years, so today there are many different methods, from experimental to numerical methods. This phenomenon can be simulated and measured by recreating air flow over an object or can be simulated in appropriate simulation programs, those who are used in Computer Fluid Dynamics (CFD). However, the experimental methods offer the exact value of this coefficient that as an urge to be known in the early phase of vehicle design [39].

Having in mind the daily changes of fuel costs, greater vehicles speeds and necessary reduction of the fuel consumption due to CO_2 emissions, further research should be directed to get a better understanding of vehicle hydrodynamics and aerodynamics in such a way to achieve a more efficient fuel consumption. For this purpose car manufacturers use vehicle models, although this experimental method requires very expensive equipment, such as the wind tunnel, which dimensions make possible a research on real scale vehicles that would lead to real experimental values. With the aim to meet this need there are numerical methods that offer a cheaper but reliable way to determine air or water flows around the vehicle in study [49].

Notwithstanding, this subject must be approached with great respect and caution, because there is much more to drag calculation than meets the eye and it's very hard to predict accurately, being very easy to over or under estimate it's value. This does not mean that the calculations are difficult, but rather that the process of identifying the sources that contribute to this phenomenon is complex. In spite of the intuitive thought of reducing this parameter in a way of achieving less movement resistance, this is also important when turning the body comes in the picture, so more important than reducing it is controlling it [25].

3.3.2 Contextualization

Airfoils and hydrofoils operate in fluids (air and water, respectively) which differ principally in density and viscosity, properties that are readily treated by the concept of Reynolds number. Since that is true, the vast amount of aerodynamic data already accumulated becomes available for use in predicting hydrofoil characteristics. Aside from the effects of cavitation then, the principal difference between airfoil and hydrofoil applications are the boundaries. In restricted areas such as shallow harbors, canals and towing tanks, other boundaries are present besides the water surface, that is, the bottom and sides. Naturally these boundaries also influence the characteristics of a hydrofoil, and their effects must be evaluated in order to use aerodynamic data for the prediction of the characteristics of hydrofoils under such conditions. In addition to the reflective influence of the bottom and sides we have the finite depth of water limits and the speed of propagation of the transverse waves generated by the hydrofoil. This change in flow causes the lift and drag characteristics to be different at speeds below this critical speed [33].

Drag, D_g, can be defined as the component of the resultant force, which is parallel to the trajectory of motion (Figure 3.10). The drag force differs from the lift force in it's direction although, like lift, drag is a component of the total aerodynamic force that results from the pressure differential over a body. Besides pressure differential, there is another component in drag named friction. This force acts parallel to the airspeed which explains why it contributes only to the drag force and not to the lift force.

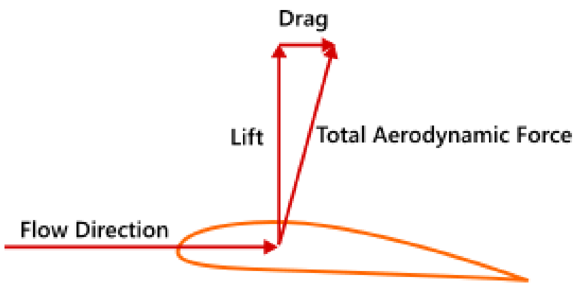


Figure 3.10: Representation of drag and lift on an airfoil

The purpose of performing a drag analysis is to estimate the magnitude of this force and understand how the geometry will behave in the practical situation that it was designed for. a classical approach to drag estimation is based on geometry and flow properties. A good estimation method must account for:

- Laminar boundary layer
- Turbulent boundary layer
- Location of laminar-to-turbulent transition
- Flow separation regions
- Compressibility

Non-dimensional coefficients are essential when working with hydrodynamics forces and moments. The coefficient form that represents the total drag is referred to as the drag model. A drag model is a mathematical expression that when multiplied by the dynamic pressure and a reference area will yield the drag force acting on the airfoil, also it expresses the Drag Coefficient (C_D) that describes how the drag of the body changes as a function of its orientation in the flow field. This model can be used in a series of important ways, ranging from plotting the drag polar (a graph in which the drag coefficient is plotted as a function of the lift coefficient), to evaluating important performance characteristics such as the lift coefficient to achieve the longest glide distance or navigate farthest [25].

Generally, the total drag is broken into two classes: flow separation (pressure drag) and skin friction (skin friction drag), thus, drag consists on the following contributions:

1. Basic pressure drag - caused by the pressure differential formed by the airfoil that acts parallel to the tangent to the glide path.
2. Skin friction drag - caused by the "rubbing" of molecules along the surface of the airfoil.
3. Lift-induced drag - caused by the circulation around the airfoil, which tilts the lift vector backwards, creating a force component that adds to the total drag.
4. Wave drag - caused by the rise in pressure around a body due to the formation of a normal shock wave on the airfoil.
5. Miscellaneous drag is caused by a number of "small" contributions that are often easily overlooked, such as small inlets and outlets, access panels, caps and other construction features that are necessary to ensure the maintenance and good function of the vehicle [25].

Therefore, drag, D_g, can be written as:

$$D_g = f(geometry, \alpha, \beta, \rho, U_\infty, Re, M) \tag{3.42}$$

where *geometry* refers to reference and wetted area, α to the angle-of attack, β to the angle-of-yaw, ρ to the water density, U_∞ to the far-field flow speed, Re is the Reynolds number and M is the Mach number. The standard way to estimate the drag value is to represent the dependency of flow speed and density through the dynamic pressure, $\frac{1}{2}\rho V_\infty^2$, the geometry using a reference area, S_{ref}, and the remaining dependencies are lumped into the Drag Coefficient, denoted by C_D. Hence, drag can be computed using the following expression:

$$D_g = \frac{1}{2}\rho U_\infty^2 S_{ref} C_D \tag{3.43}$$

The Equation (3.43) can be written in order to the Drag Coefficient (C_D), so it results in Equation (3.44)

$$C_D = \frac{D_g}{\frac{1}{2}\rho U_\infty^2 S_{ref}} \tag{3.44}$$

Basic drag modeling is the mathematical combination of all sources of drag for a vehicle, such that the effect of changing its orientation with respect to its path of motion and fluid velocity is realistically replicated. This modeling culminates in the determination of the total drag coefficient, CD. As stated before, the total drag coefficient comprises the effect of basic pressure drag, skin friction drag, lift-induced drag, wave drag and contributions from other sources, commonly referred to as miscellaneous drag.

$$C_D = C_{D0} + C_{Df} + C_{Di} + C_{Dw} + C_{Dmisc} \tag{3.45}$$

where C_D is the total drag coefficient, C_{D0} is the basic drag coefficient (pressure drag), C_{Df} defines the skin friction drag coefficient, C_{Di} is the induced drag coefficient (pressure drag), C_{Dw} defines wave drag and C_{Dmisc} represents the miscellaneous or additive drag. Each of the previous components depend on the geometry as well as its orientation and flowspeed. For low subsonic speeds, the wave drag component can be neglected because of it's insignificant value [25]. Also, having in mind that miscellaneous drag cannot be defined by an equation because of it's subjectivity, the Drag Coefficient equation results in:

$$C_D = C_{D0} + C_{Df} + C_{Di} \tag{3.46}$$

Basic drag can be seen as a pressure drag force caused by resultant pressure distribution over the surface of the body. It can be treated as the component of the pressure force parallel to the tangent to the glide path. The force is the product of the pressure acting on a cross-sectional area of the body, normal to the flight path.

$$D_0 = \int_S (Pxn) dA \tag{3.47}$$

where n is the normal to the glide path and S is the surface of the body. After obtaining the value of the basic drag force, the basic drag coefficient can be defined as:

$$C_{D0} = \frac{2D_0}{\rho U_\infty^2 S_{ref}} \tag{3.48}$$

Ultimately, the basic drag force can be thought of as the increase in the skin friction forces due to the applied form factor [25]. Skin friction is caused by a fluid's viscosity as it flows over a surface. Its magnitude depends on the viscosity of the fluid and the wetted (or total) surface area in contact with it, as well as the surface roughness. The skin friction drag coefficient is defined as follows:

$$C_{Df} = \frac{2D_f}{\rho U_\infty^2 S_{ref}} \tag{3.49}$$

Lift induced drag is caused by the flow circulation around the surface of the airfoil, which will reveal through the formation of vortices. The lift-induced drag is a pressure drag force and can be defined as:

$$C_{Di} = \frac{2D_i}{\rho U_\infty^2 S_{ref}} \tag{3.50}$$

which can be reformulated by combining Equations (3.46), (3.48), (3.49) and (3.50) and the result would be:

$$C_D = \frac{2D_0}{\rho U_\infty^2 S_{ref}} + \frac{2D_f}{\rho U_\infty^2 S_{ref}} + \frac{2D_i}{\rho U_\infty^2 S_{ref}} \tag{3.51}$$

3.4 Turbulent Kinetic Energy (k)

3.4.1 Introduction

Turbulence is one of the most prevailing mysteries of physics. After more than a century studying turbulence we've only come up with a few answers for how it works and affects the world around us. And yet, turbulence is ubiquitous, springing up in virtually any system that has moving fluids. Understanding precisely how this phenomenon works would have a bearing effect on many aspects of our lives.

Usually, liquids and gases have two types of motion: a laminar flow, which is stable and smooth and a turbulent flow, which is composed by seemingly unorganized swirls. The laminar flow is steady and easy to predict, on the other hand a turbulent flow is unstable and the pattern of movement is chaotic. That phenomenon is turbulence in action and turbulent flows have certain characteristics in common. Firstly, turbulence is always chaotic, this definition is different from being random. Rather, this means that turbulence is very sensitive to disruptions, a little change can eventually turn into completely different results. That fact makes it nearly impossible to predict what will happen, even with a lot of information about the current state of a system . Another important characteristic of turbulence is the different scales of motion that these flows display. Turbulent flows have many different sized whirls called eddies, which are like vortexes of different sizes and shapes. All those different sized eddies interact with each other, breaking up to become smaller and smaller, until all that movement is transformed into heat, in a process called the "energy cascade" [14].

In every flowing liquid or gas there are two opposing forces: inertia and viscosity. Inertia can be defined as the tendency of fluids to keep moving which causes instability and posteriorly turbulence, while viscosity works against disruption, turning the flow laminar. In thick fluids such as honey viscosity is prominent, but on less viscous substances like water or air, are more prone to inertia, which creates instability that develops into turbulence. It's possible to measure where a flow falls on that spectrum with the Reynolds number, which is the ratio between a flow's inertia and its viscosity [11]. The higher it's value the more likely is is that turbulence will occur.

$$Re = \frac{Inertial\, forces}{Viscous\, forces} = \frac{\rho \cdot U \cdot L}{\mu} = \frac{U \cdot L}{\nu} \tag{3.52}$$

In this equation ρ is the fluid density, U is the flow velocity, L is the characteristic length and μ represents the bulk viscosity. Knowing that $\frac{\rho}{\mu}$ is equal to ν, which is the kinematic viscosity of the fluid, the Reynolds number can be expressed as shown in Equation (3.52) [11].

Turbulence affects the mean (non-turbulent) part of the flow through a specific mechanism called eddy flux. The most unique measure for turbulence is the kinetic energy of the turbulent part of the flow. This concept is called Turbulent Kinetic Energy, k, which measures the kinetic energy per unit mass of the turbulent fluctuations in a turbulent flow. The SI unit of k is $J/kg = m^2/s^2$ and, as is noticeable from the name of this quantity, the value of k directly represents the 'strength' of the turbulence in the flow, thus it can be seen as a measure of the intensity of turbulence [50].

In computational of turbulent flows the range of length scales and complexity of phenomena involved in turbulence make most modeling approaches prohibitively expensive. The resolution required to solve all scales involved in turbulence is beyond what is computationally possible, thus, the primary approach in such cases is to create numerical models to approximate unresolved phenomena. Turbulence models can be classified based on computational expense, which corresponds to the range of scales that are modeled versus resolved. The more turbulent scales that are resolved, the finer the resolution of the simulation, and therefore the higher the computational cost. If a majority or all of the turbulent scales are not modeled, the computational cost is very low, but the trade-off comes in the form of decreased accuracy [32].

In order to achieve the best results possible, by balancing computational cost with accuracy the Shear-Stress Transport (SST) k - ω Model was used. The following section is a extensive description of this method.

3.4.2 Shear-Stress Transport (SST) k - ω Model

Overview

The shear-stress transport (SST) k - ω model was developed by Menter in the year of 1993 to effectively blend the robust and accurate formulation of the k - ω model in the near-wall region with the free-stream independence of the k - ϵ model in the far field. To achieve this, the k - ϵ model is converted into a k - ω formulation. The SST k - ω model is similar to the standard k - ω model, but includes the following refinements:

- The standard k - ω model and the transformed k - ϵ model are both multiplied by a blending function and both models are added together. The blending function is designed to be one in the near-wall region, which activates the standard k - ω model, and zero away from the surface, which activates the transformed k - ϵ model.
- The SST model incorporates a damped cross-diffusion derivative term in the ω equation.
- The definition of the turbulent viscosity is modified to account for the transport of the turbulent shear stress.
- The modeling constants are different.

These features make the SST k - ω model more accurate and reliable for a wider class of flows (e.g. adverse pressure gradient flows, airfoils, transonic shock waves) than the standard k - ω model. Other modifications include the addition of a cross-diffusion term in the ω equation and a blending function to ensure that the model equations behave appropriately in both the near-wall and far-field zones [2].

The SST k - ω turbulence model is a two-equation eddy-viscosity model which has become very popular. The shear stress transport (SST) formulation combines the best of two worlds. The use of a k - ω formulation in the inner parts of the boundary layer makes the model directly usable all the way down to the wall through the viscous sub-layer, hence the SST k - ω model can be used as a Low-Re turbulence model without any extra damping functions. The SST formulation also switches to a k - ϵ behaviour in the free-stream and thereby avoids the common k - ω problem that the model is too sensitive to the inlet free-stream turbulence properties. Authors who use the SST k - ω model often merit it for its good behaviour in adverse pressure gradients and separating flow. The SST k - ω model does produce a bit too large turbulence levels in regions with large normal strain, like stagnation regions and regions with strong acceleration. This tendency is much less pronounced than with a normal k - ϵ model [47].

Transport Equations for the SST k - ω Model

The SST k - ω model has a similar form to the standard k - ω model. The Turbulent Kinetic Energy, k, and the specific dissipation rate, ω, are obtained from the following transport equations:

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_i}(\rho k u_i) = \frac{\partial}{\partial x_j}\left(\Gamma_k \frac{\partial k}{\partial x_j}\right) + \widetilde{G}_k - Y_k + S_k \tag{3.53}$$

and

$$\frac{\partial}{\partial t}(\rho \omega) + \frac{\partial}{\partial x_i}(\rho \omega u_i) = \frac{\partial}{\partial x_j}\left(\Gamma_\omega \frac{\partial \omega}{\partial x_j}\right) + G_\omega - Y_\omega + D_\omega + S_\omega \tag{3.54}$$

In these equations, Γ_k and Γ_ω represent the effective diffusivity of k and ω, respectively. $\widetilde{G_k}$ represents the generation of Turbulent Kinetic Energy due to mean velocity gradients. G_ω represents the generation of ω. Y_k and Y_ω represent the dissipation of k and ω due to turbulence. All of the above terms are calculated as described below. S_k and S_ω are user-defined source terms. Also u_i represents velocity, $\frac{\partial}{\partial t}$ is the partial derivative in order of time, ρ is the specific density, $\frac{\partial}{\partial x_i}$ and $\frac{\partial}{\partial x_j}$ are partial derivative in order of space.

Modeling the Effective Diffusivity

The effective diffusivities for the SST k - ω model are given by

$$\Gamma_k = \mu + \frac{\mu_t}{\sigma_k} \tag{3.55}$$

$$\Gamma_\omega = \mu + \frac{\mu_t}{\sigma_\omega} \tag{3.56}$$

where μ is the dynamic viscosity, σ_k and σ_ω are the turbulent Prandtl numbers for k and ω, respectively.

$$\sigma_k = \frac{1}{F_1/\sigma_{k,1} + (1 - F_1)/\sigma_{k,2}} \tag{3.57}$$

$$\sigma_\omega = \frac{1}{F_1/\sigma_{\omega,1} + (1 - F_1)/\sigma_{\omega,2}} \tag{3.58}$$

where $\sigma_{k,1} = 1.176$, $\sigma_{k,2} = 1.0$, $\sigma_{\omega,1} = 2.0$, $\sigma_{\omega,2} = 1.168$. The blending function F_1 is given by

$$F_1 = \tanh\left(\Phi_1^4\right) \tag{3.59}$$

$$\Phi_1 = \min\left[\max\left(\frac{\sqrt{k}}{0.09\omega y}, \frac{500\omega}{\rho y^2 \omega}\right), \frac{4\rho k}{\sigma_{\omega,2} D_\omega^+ y^2}\right] \tag{3.60}$$

$$D_\omega^+ = \max\left[2\rho \frac{1}{\sigma_{\omega,2}} \frac{1}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j}, 10^{-10}\right] \tag{3.61}$$

where y is the distance to the next surface and D_ω^+ is the positive portion of the cross-diffusion term, see Equation (3.86).

The turbulent viscosity, μ_t, is computed as follows:

$$\mu_t = \frac{\rho k}{\omega} \frac{1}{\max\left[\frac{1}{\alpha^*}, \frac{SF_2}{a_1 \omega}\right]} \tag{3.62}$$

where $a_1 = 0.31$. S is the modulus of the mean rate-of-strain tensor, defined in the same way as for the k - ϵ model.

$$S \equiv \sqrt{2S_{ij}S_{ij}} \tag{3.63}$$

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) \tag{3.64}$$

The blending function F_2 is given by

$$F_2 = \tanh\left(\Phi_2^2\right) \tag{3.65}$$

$$\Phi_2 = \max\left[2\frac{\sqrt{k}}{0.09\omega y}, \frac{500\mu}{\rho y^2 \omega}\right] \tag{3.66}$$

The coefficient α^* damps the turbulent viscosity causing a low-Reynolds-number correction. It is given by

$$\alpha^* = \alpha^*_\infty \left(\frac{\alpha^*_0 + \mathrm{Re}_t/R_k}{1 + \mathrm{Re}_t/R_k}\right) \tag{3.67}$$

$$\mathrm{Re}_t = \frac{\rho k}{\mu \omega} \tag{3.68}$$

$$\alpha^*_0 = \frac{\beta_i}{3} \tag{3.69}$$

$$\beta_i = F_1 \beta_{i,1} + (1 - F_1)\beta_{i,2} \tag{3.70}$$

where $\beta_{i,1} = 0.075$, $\beta_{i,2} = 0.0828$, $R_k = 6$, $\alpha^*_\infty = 1$. Note that, in the high-Reynolds-number form of the k - w model, $\alpha^* = \alpha^*_\infty = 1$.

Modeling the Turbulence Production

The term $\widetilde{G}_k$ represents the production of turbulence kinetic energy, and is defined as:

$$\widetilde{G}_k = \min(G_k, 10\rho\beta^* k\omega) \tag{3.71}$$

where G_k is defined in the same manner as in the standard k - ω model. From the exact equation for the transport of k, this term may be defined as

$$G_k = -\rho \overline{u_i' u_j'} \frac{\partial u_j}{\partial x_i} \tag{3.72}$$

To evaluate G_k in a manner consistent with the boussinesq hypothesis,

$$G_k = \mu_t S^2 \tag{3.73}$$

and

$$\beta^* = \beta_i^* \left[1 + \zeta^* F(\mathrm{M}_t)\right] \tag{3.74}$$

$$\beta_i^* = \beta_\infty^* \left(\frac{4/15 + (\mathrm{Re}_t/R_\beta)^4}{1 + (\mathrm{Re}_t/R_\beta)^4} \right) \tag{3.75}$$

$$F(\mathrm{M}_t) = \begin{cases} 0 & \mathrm{M}_t \le \mathrm{M}_{t_0}; \\ \mathrm{M}_t^2 - \mathrm{M}_{t_0}^2 & \mathrm{M}_t > \mathrm{M}_{t_0} \end{cases} \tag{3.76}$$

$$\mathrm{M}_t^2 \equiv \frac{2k}{a^2} \tag{3.77}$$

$$a = \sqrt{\gamma R T} \tag{3.78}$$

where $\zeta^* = 1.5$, $R_\beta = 8$, $\beta_\infty^* = 0.09$, $\mathrm{M}_{t_0} = 0.25$, γ is the ration of specific heat, R is the gas constant and T is the temperature. Note that, in the high-Reynolds-number form of the k - ω model, $\beta_i^* = \beta_\infty^*$, and also that in the incompressible form, $\beta^* = \beta_i^*$.

The term G_ω represents the production of ω and is given by

$$G_\omega = \frac{\alpha}{\nu_t} \widetilde{G}_k \tag{3.79}$$

$$\alpha = \frac{\alpha_\infty}{\alpha^*} \left(\frac{\alpha_0 + \mathrm{Re}_t/R_\omega}{1 + \mathrm{Re}_t/R_\omega} \right) \tag{3.80}$$

where $\alpha_0 = 1/9$, $R_\omega = 2.95$, $\alpha^* = 0.024$ and Re_t is given by Equation (3.68). Note that the formulation of G_ω differs from the standard k - ω model. The difference between the two models also exists in the way the term α_∞ is evaluated. In the standard k - ω model, α_∞ is defined as a constant, for the SST k - ω model, α_∞ is given by

$$\alpha_\infty = F_1 \alpha_{\infty,1} + (1 - F_1)\alpha_{\infty,2} \tag{3.81}$$

where

$$\alpha_{\infty,1} = \frac{\beta_{i,1}}{\beta_\infty^*} - \frac{\kappa^2}{\sigma_{w,1}\sqrt{\beta_\infty^*}} \tag{3.82}$$

$$\alpha_{\infty,2} = \frac{\beta_{i,2}}{\beta_\infty^*} - \frac{\kappa^2}{\sigma_{w,2}\sqrt{\beta_\infty^*}} \tag{3.83}$$

where $\beta_\infty^* = 0.09$, $\kappa = 0.41$,

Modeling the Turbulence Dissipation

The term Y_k represents the dissipation of turbulence kinetic energy, and is defined in a similar manner as in the standard k - ω model. The difference is in the way the term f_{β^*} is evaluated. In the standard k - ω model, f_{β^*} is defined as a piecewise function. For the SST k - ω model, f_{β^*} is a constant equal to 1. Thus,

$$Y_k = \rho \beta^* k \omega \tag{3.84}$$

The term Y_ω represents the dissipation of ω, and is defined in a similar manner as in the standard k - ω model. The difference is in the way the terms β_i and f_β are evaluated. In the standard k - ω model, β_i is defined as a constant and f_β is defined by a function. For the SST k - ω model, f_β is a constant equal to 1. Thus,

$$Y_\omega = \rho \beta_i \omega^2 \tag{3.85}$$

Instead of having a constant value, β_i is given by Equation (3.70) and F_1 is obtained from Equation (3.59).

Cross-Diffusion Modification

The SST k - ω model is based on both the standard k - ω model and the standard k - ϵ model. To blend these two models together, the standard k - ϵ model has been transformed into equations based on k and ω, which leads to the introduction of a cross-diffusion term (D_ω in Equation (3.54)). D_ω is defined as

$$D_\omega = 2\left(1 - F_1\right) \rho \sigma_{\omega,2} \frac{1}{\omega} \frac{\partial k}{\partial x_j} \frac{\partial \omega}{\partial x_j} \tag{3.86}$$

3.5 Software Used

3.5.1 Ansys

Design

The first aspect of pre-processing is the geometry definition, this is the core of the work and it delineates how the study is done and what aspects and parameters were examined. Besides geometry being an essential component of engineering simulation, it also links engineering simulation with design and manufacturing, and therefore plays an important role in simulation driven product development. In Ansys Workbench, the geometry can either be imported from a file or drawn using Ansys Design Modeler or Ansys SpaceClaim. For this study case Ansys Design Modeler was used.

With direct interface with all major computer-aided design (CAD) systems, support from reader and translators and other characteristics Ansys Design Modeler offers a comprehensive geometry handling solution for engineering simulation in an integrated environment. This tool provides unique modeling functions for simulation that include parametric geometry creation, concept model creation, CAD geometry modification, automated cleanup and repair, and several custom tools designed for fluid flow, structural and other types of analyses [8].

Mesh

The most important part of pre-processing is the mesh generation. It consists in the subdivision of the domain into smaller non-overlapping subdomains (elements or cells) where the governing equations are solved numerically determining the discrete values of pressure, velocity and other variables of interest. The accuracy of the CFD solution is not only influenced by the size, in other words, by the number of cells in the mesh, as stated above, but also by the type of mesh, the order of accuracy of the numerical method, and the adequacy of the numerical methods chosen to describe the physics of the problem [31]. This increase in accuracy comes with the cost of additional computational power and computational time requirements, thus a better and more automated meshing tool is crucial in an attempt to get a fast and accurate solution.

Ansys provides a meshing software that produces accurate and efficient results with a wide range of applicability from general purpose to high-performance solutions. The methods available cover the meshing spectrum of high-order to linear elements and fast tetrahedral and polyhedral to high-quality hexahedral and Mosaic. Smart defaults are built into the software to make meshing a painless and intuitive task delivering the required resolution to capture solution gradients properly for dependable results.

The cells can be of several shapes. Bi-dimensional meshes usually use triangle or quadrilateral cells, while elements of tri-dimensional meshes are generally tetrahedral or hexahedral. In this study case, a bi-dimensional mesh was used because the problem solution only required two dimensions. The mesh can be classified into structured or non-structured based on the cells connectivity. In a structured mesh each cell has the same number of neighboring cells, they follow an uniform pattern and are usually quadrilaterals (2D) or hexahedron (3D), while in an non-structured mesh there is not a regular pattern and its cells are usually triangles (2D) or tetrahedrons (3D) [31]. The advantage of structured meshes lies on its connectivity which makes it faster to solve, while non-structured meshes are better suited for complex geometries as the skewness is not as intense as in structured grids, which can lead to nonphysical solutions [31]. Since the geometry used in this study is bi-dimensional, a non-structured triangular and quadrilateral mesh was used.

Fluent

Ansys Fluent software contains the broad physical modeling capabilities needed to model flow turbulence, heat transfer and reactions for industrial applications. With a highly scalable, high-performance computing (HPC) to help solve complex and large-model CFD simulations quickly and cost-effectively, Ansys Fluent became a well known and exploited solution in this area. These properties allowed it's implementation in a wide range of applications such as : study of an air flow over an aircraft wing to combustion in a furnace, from bubble columns to oil platforms, from blood flow to semiconductor manufacturing and from clean room design to wastewater treatment plants. It also possesses a fault-tolerant workflow, that speeds meshing for non-watertight ("dirty") geometries with use of a "wrapper"- a layer of mesh that covers surface imperfections in the geometry, so that complex models, that previously took days or even weeks can now be meshed and solved in hours with minimal sacrifice to simulation accuracy.

Ansys Fluent software places special emphasis on providing a wide range of turbulence models to capture the effects of turbulence accurately and efficiently, several innovative models such as the Menter–Langtry laminar–turbulent transition model are available only in Fluent. The main goal of this work is to evaluate the kinematic turbulence along with the coefficient of drag of a water flow around an AUV body, which is very similar to an airfoil, so this computational solution becomes very attractive for this matter [7].

3.5.2 MATLAB

MATLAB is a high-performance language for technical computing. It integrates computation, visualization and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation [46]. Typical applications include:

- Math and computation
- Algorithm development
- Modeling simulation and prototyping
- Data analysis, exploration and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

The name MATLAB stands for "MATrix LABoratory". MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation [46].

3.5.3 Fortran

Fortran was originally named after the contraction of "Formula Translation", highlighting Fortran's origins as a language designed specifically for mathematical calculations. Fortran was developed in the early 1950s and the first ever Fortran program ran in 1954. Fortran has outlived several nation states since its conception, and still is in wide use today in a number of specialised scientific communities [54].

Fortran is a computer programming language that is extensively used in numerical, scientific computing. While outwith the scientific community it still has a strong user base with scientific programmers and is also used in organizations such as weather forecasters, financial trading, and in engineering simulations. Fortran programs can be highly optimised to run on high performance computers, and in general the language is suited to producing code where performance is important [54].

(This page was intentionally left blank)

Chapter 4

Case Study

In this chapter a description of the methods and procedures utilized in this work is realized. The author pretends to give the reader an insight about how the study was conducted.

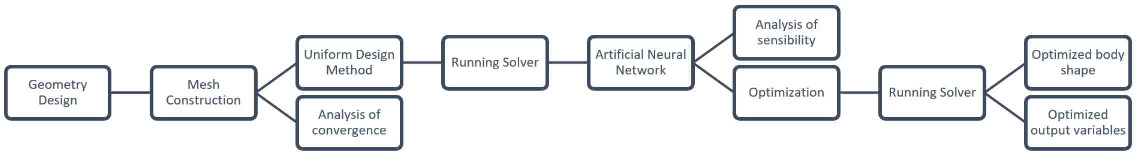


Figure 4.1: Schema of the procedure used in this case study

4.1 Computer Fluid Dynamics (Ansys)

For performing a hydrodynamic simulation in Ansys software three different stages are necessary: Geometry, Mesh and Fluent (see Figure 4.2). This stages are the base of the software manipulation. First there is the Geometry stage where, as the name says, the geometry is defined, for the domain and the body, together with the geometrical variables. Next there is the Mesh stage, were the software takes the Geometry as an input and constructs the mesh around all the domain. This is where the mesh characteristics are defined such as refinement, growth and inflation. Lastly, there's the Fluent stage which is the solver of this software. This is where all the numerical solution requirements are established and also where the output variables are defined.

4.1.1 Geometry

First a rectangle was drawn to represent the geometrical domain of the problem, this is a large body/surface in comparison with the body in study. The rectangle as the following dimensions : 40 m x 15 m, see Figure A.1. The distance from the inlet boundary to the body is 10 m, and the distance of the rear part of the body and the outlet boundary is more than 20 m to assure that the flow around the body is not affected by the boundary conditions and can be measured in the stipulated domain (Figure A.2). One frequent CFD recommendation is to leave at least three times the body length to the outlet boundary, as so, this recommendation is fully satisfied.

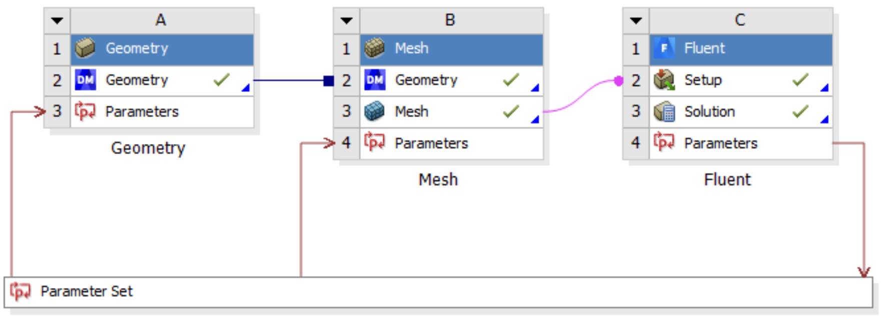


Figure 4.2: Three stages prevalent in Ansys software: Geometry, Mesh and Fluent

The body itself is composed by 7 variables: Length (L), Diameter (D), Front Length (L_f), Rear Length (L_r), Front Radius (R_f), Rear Radius (R_r) and Middle Radius (R_m). The body started as a simple rectangle where L and D were defined (Figure A.3). Next, a triangle was drawn in the front and rear of the body, where L_f and L_r were defined (Figures A.4). Using the triangle vertice and both corners a middle point was defined. Using that middle point a perpendicular line was created (Figure A.5 and A.6). Using this line and the respective corners of the triangle, a radius was defined, which value depends on the line length (Figures A.7, A.8 and A.9). This process was done four times, so four radius were defined, see Figure A.10. As shown in Figure A.9, the two radius mentioned before will only represent one radius, with the vertice as it's connection point, this is the way R_f was defined. The same process was done for the R_r. Last, the R_m was defined using the most distanced corners of the rectangle, a middle point between them and a perpendicular line, in a process similar to that previously described (see Figures A.11 and A.12). The final result can be seen in Figure 4.3.

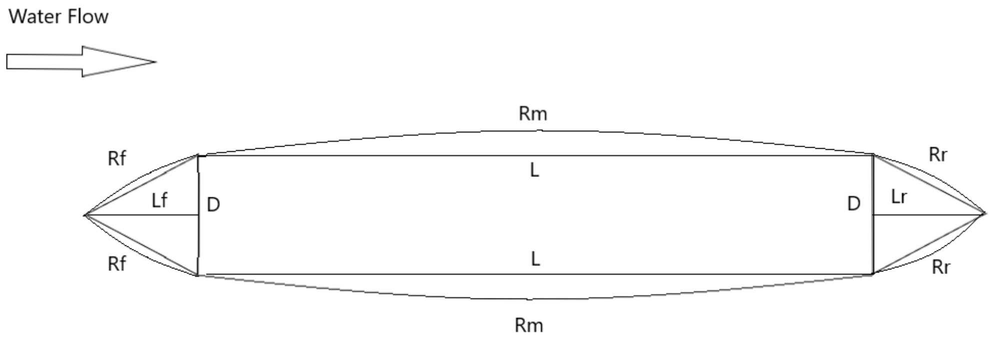


Figure 4.3: Schema of the geometrical variables that compose the body

Lastly, a partial elliptic/circular body was created around the body to be used later in the mesh generation, this can be called a body of control and does not add material to the domain. Two kinds of body's of control were created, one type for the "INITIAL" and other "FINAL" meshes, these can be seen in Figure A.13 and A.14, respectively. Also, to help the program processing, two symmetry lines were drawn: one intersecting the body in its horizontal middle plan, O_x (also used for mesh generation) and other intersecting the body in its vertical middle plan, O_y, see Figure A.15 and Figure A.16, respectively.

4.1.2 Mesh

Procedure

In order to evaluate the results obtained and to assess the convergence six types of meshes were constructed. These meshes are divided into two groups: ”INITIAL” (Figure 4.4) and ”FINAL” (Figure 4.5). The difference between them consists in the body of control dimensions and the refinement of the ”tail” of the body along the domain. To do this, the line that crosses the horizontal symmetry of the body was used. Inside this two groups exist three kinds of nets named ”1” , ”2” and ”3”, being ”1” the less refined and ”3” the most refined. This means that mesh ”3” as more elements in it and the elements size is smaller, so it is more precise (see Table 4.1). It should be remarked that we used a STUDENT VERSION of Ansys, which means that the maximum number of elements is restricted to 512000 elements.

Table 4.1: Size values for each mesh element

Units [m]		Domain Mesh	Body Sizing	Edge Sizing (Body)	Edge Sizing (Tail)
INITIAL	MESH1	1.50E-01	1.50E-02	1.50E-03	-
	MESH2	1.00E-01	1.00E-02	1.00E-03	-
	MESH3	9.00E-02	9.00E-03	9.00E-04	-
FINAL	MESH1	1.50E-01	1.50E-02	1.50E-03	1.50E-02
	MESH2	1.00E-01	1.00E-02	1.00E-03	1.00E-02
	MESH3	9.00E-02	9.00E-03	9.00E-04	9.00E-03

First of all, the mesh was created in all domain with a constant value for the mesh size, this and the next values depend on which kind of mesh is in use (”1”,”2” or ”3”). Afterward a more refined mesh (more elements) is created in the body of control/influence (Figure B.1), and a even more refined mesh is generated on the edges of the body (Figure B.2). Also on the edges of the body is created a inflation with ten layers (Figure B.3). As mentioned before, on the meshes from the ”FINAL” group, a more refined mesh is created around the horizontal symmetry axis (Figure B.4) with the same mesh size as the body of control, the reason for this implementation can be explained this way: when the flow encounters the rigid body it suffers a change because of it’s interference with the body walls, and that change does not disappear immediately when the body ends, in fact the most perturbed part of the flow occurs at the end of the rigid body. However, as can easily be perceived, that disturbance is going to fade gradually in the flow domain after the body edge. This refinement implementation leads to a better approximation of the flow disturbance caused by the rigid body and therefore a better solution field. At last, the growth rate used for the ”INITIAL” and ”FINAL” meshes is the pre-definition value of 1.2, this ensures that the growth in the elements size is ”smooth” and the result will be a precise net.

The last thing to be done was the definition of the domain boundaries: ”*Inlet*”, ”*Outlet*”, ”*WallTop*”, ”*WallBottom*”, ”*WallBodyTop*”, ”*WallBodyBottom*”. This kind

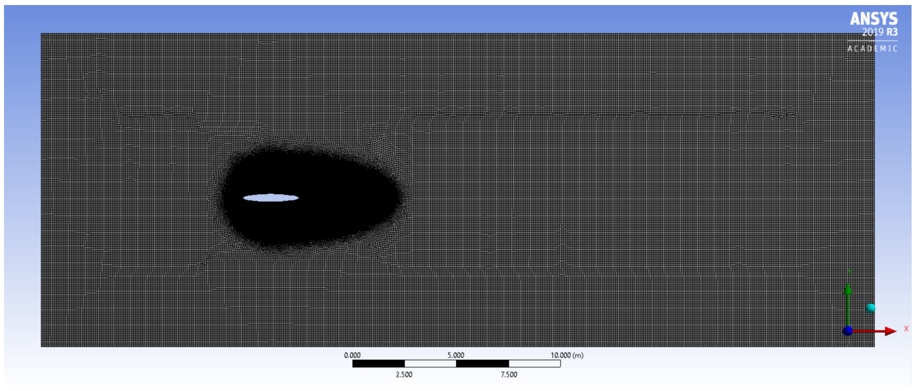


Figure 4.4: Image from Ansys Meshing procedure: "INITIAL" mesh

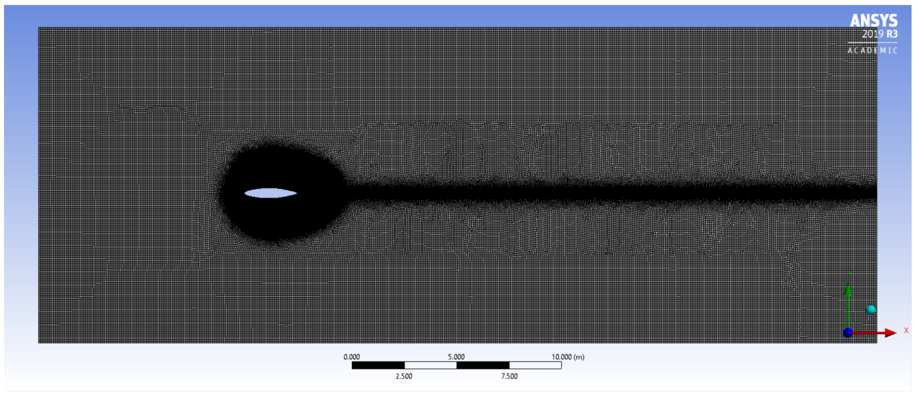


Figure 4.5: Image from Ansys Meshing procedure: "FINAL" mesh

of nomenclature is useful when introducing the mesh in Ansys Fluent, because this way the program identifies each kind of boundary and their respective function. These procedure is represented in Appendix C in Figures B.5, B.6, B.7, B.8, B.9, B.10. Also on Appendix C it's possible to take a closer look to the "FINAL" mesh in Figures B.11, B.12, B.13 and B.14.

Convergence

As mentioned before in the present work, a crucial requirement for CFD simulations is convergence. This means that results must converge to a unique solution or to a solution with a small predefined discrepancy. With the aim of obtaining this requirement a study was done involving the behavior of the Turbulent Kinetic Energy (k) for the six computational meshes: "M1_INITIAL", "M2_INITIAL", "M3_INITIAL", "M1_FINAL", "M2_FINAL", "M3_FINAL".

In Figure 4.6 the Turbulent Kinetic Energy (k) measured from the O_x symmetry is represented for all the domain extension. Starting by the beginning of the domain there is a small amount of turbulence caused by the initiation of the flow, this should be neglected because it has no meaning for the following study. It's possible to identify that the maximum values occur after the mark of twelve meters (see Figure 4.7), this is where the body ends, which will cause an increase in the flow turbulence. It's also possible to observe that the "k" values obtained from the "INITIAL" meshes are bigger than the ones obtained using the "FINAL" meshes.

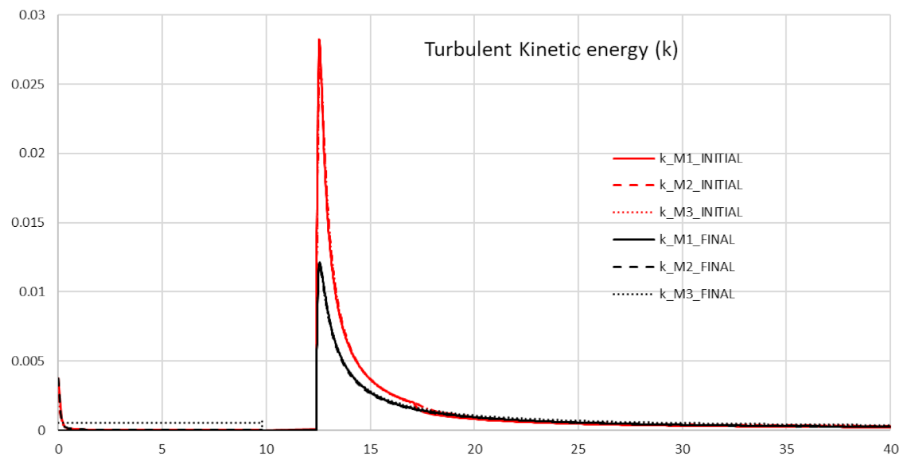


Figure 4.6: Turbulent kinetic energy (k) [J/kg]: full domain

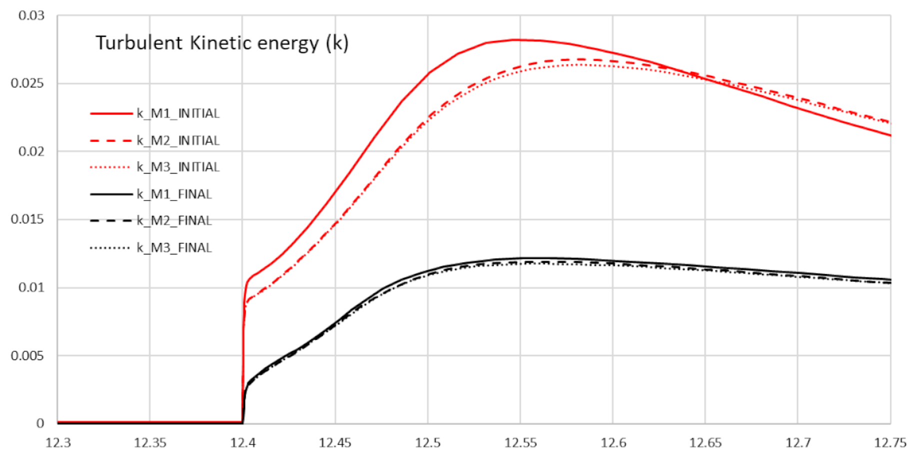


Figure 4.7: Turbulent kinetic energy (k) [J/kg]: exponential increase of the values

In a way to take a closer look to the maximum values obtained for for the "INITIAL" and "FINAL" meshes, in Figure 4.8 and Figure 4.9 there is a representation of each peak,"INITIAL" and "FINAL" meshes, respectively. Inside each figure, meshes "1", "2" and "3" are represented, as mentioned before "M1" is the least refined mesh and "M3" is the most refined one. By observing those figures the conclusion supports the previous observation, the values obtained from the less refined meshes are bigger than the ones obtained using the more refined meshes.

In Figures 4.7, 4.8 and 4.9 is also possible to recognize that the values obtained for the "FINAL" meshes ("1", "2" and "3") are more similar to each other than the "INITIAL" meshes, which also can be accounted as a positive factor for convergence. Another important aspect is the "smoothness" of the lines. Since the natural phenomenon is progressive the approximation values represented by the lines must have a linear behaviour.

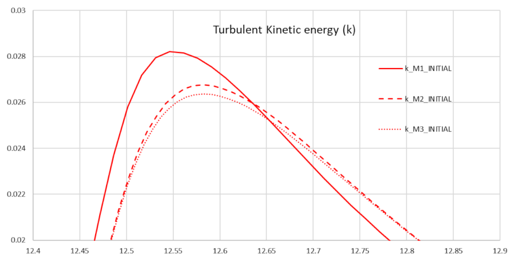


Figure 4.8: Turbulent kinetic energy (k) [J/kg]: peak values of the "INITIAL" meshes

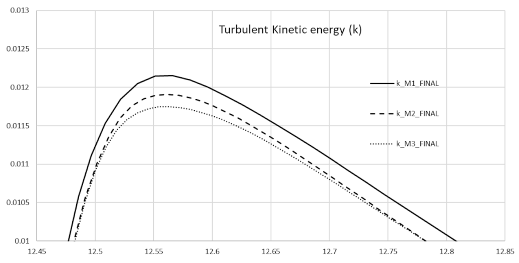


Figure 4.9: Turbulent kinetic energy (k) [J/kg]: peak values of the "FINAL" meshes

As can be seen in Figure 4.10, in the "INITIAL" meshes there is an abrupt decrease in the kinematic turbulence, specially on "M1", that doesn't happen in the "FINAL" meshes, which proves that the values are more similar to reality which means that the "FINAL" construction of the mesh is better suited to the problem.

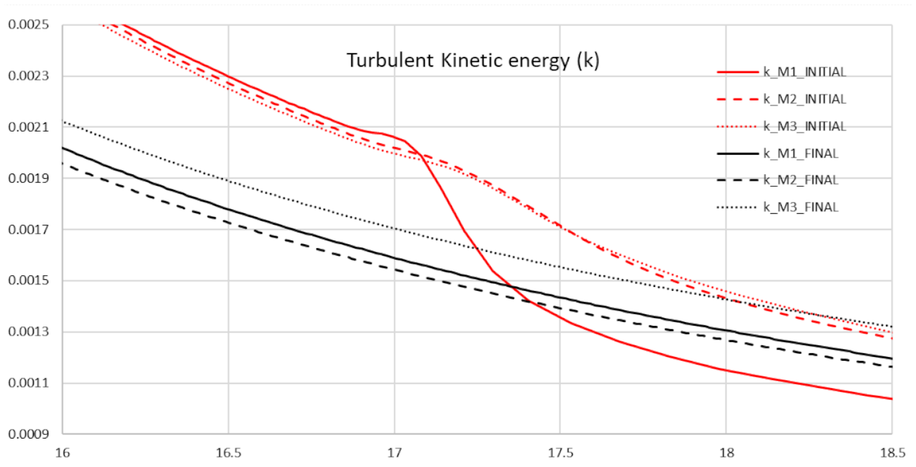


Figure 4.10: Turbulent kinetic energy (k) [J/kg]: linear decrease of the values

In a further study, with the same goal of verifying the mesh convergence, an analysis was done using the global values of the Turbulent kinetic energy (k) and Drag Coefficient (C_D) for all the domain, unlike before that the values corresponded to the o_x symmetry. Therefore there's only one value for each variable (k and C_D) in each mesh. These values can be seen in Table 4.2. For a more intuitive observation two plots were done, these are represented in Figure 4.11 which represents the Drag Coefficient (C_D) and in Figure 4.12 which represents the Turbulent Kinetic Energy (k) for all the meshes in this study case. As can be seen in these plots convergence of the values is not completely ensured since that, for both variables, when the mesh refinement increases the value obtained is substantially different from the others.

By looking at the Drag Coefficient (C_D) plot (Figure 4.11), two lines can be spotted. The red one is referent to the "INITIAL" meshes and the black one is referent to the "FINAL" meshes. Looking at the red line is possible to observe that the C_D values are decreasing with the increase of elements, this means that the value obtained with the mesh "M1" is

Table 4.2: Number of elements, Turbulent kinetic energy (k) [J/kg] and Drag Coefficient (C_D) values for all the meshes

	INITIAL MESH			FINAL MESH		
	M1	M2	M3	M1	M2	M3
C_D	0.037763	0.037579	0.03732	0.031044	0.03122	0.032538
k	0.088908	0.079005	0.077015	0.038711	0.07557	0.308629
Elements	194043	422669	505288	221215	404850	474240

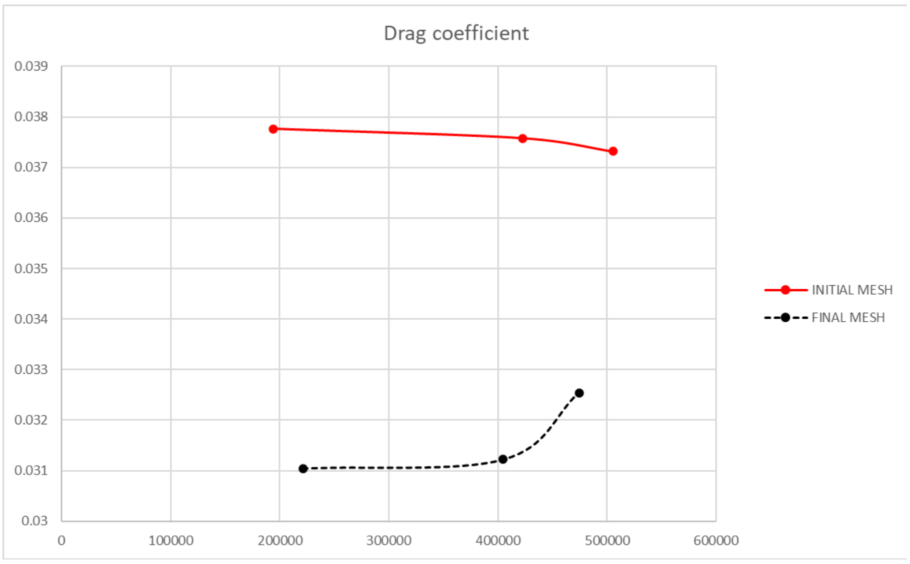


Figure 4.11: Drag Coefficient (C_D): linear decrease of the values

greater than the one obtained from the "M3". On the other hand, when looking at the black line is possible to observe an increase in the C_D value when the number of elements increases, which means that the values of C_D measured from the mesh "M3" are greater than the ones measured from "M1". Besides, the values obtained for the "INITIAL" meshes are very close to each other, while the ones obtained for the "FINAL" meshes, specially from the mesh "M3", is substantially different from the meshes "M1" and "M2". This could potentially mean that by increasing the number of elements better results could be achieved, although this might be true the computational tool utilized, "STUDENT VERSION of Ansys", has the maximum number of elements of 512000 elements, which restraints further refinent of the mesh.

In a similar way, by looking at the Turbulent Kinetic Energy (k) plot (Figure 4.12) two lines can be detected with the same color "nomenclature" from before, the red one is referent to the "INITIAL" mesh and the black one is referent to "FINAL" meshes. Like before the values from the red line decrease with the increase of the number of elements while the values from the black line increase with the increase of the of the number of elements. Unlike the previous plot, where the values of the C_D variable of the "INITIAL" meshes where always bigger than the "FINAL" ones, for the k variable the values from the "INITIAL" meshes are bigger in the first two meshes ("M1" and "M2") but the value from the mesh "M3" is considerably bigger on the "FINAL" mesh.

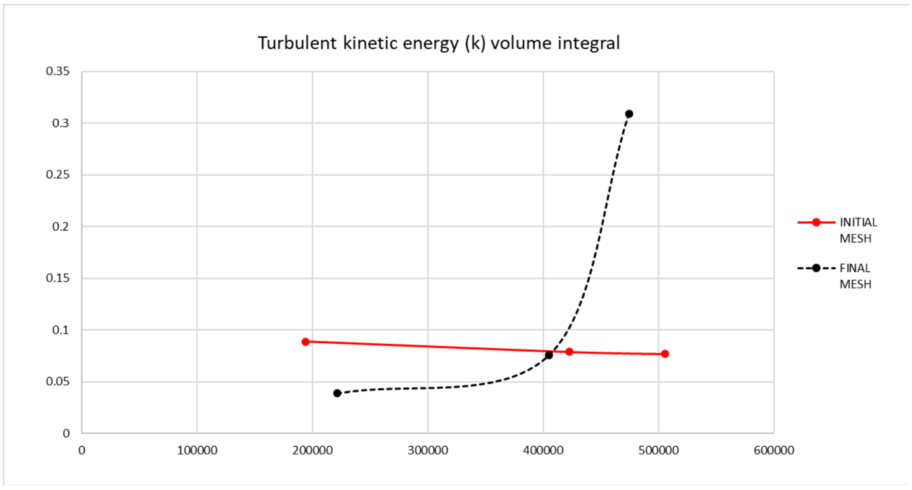


Figure 4.12: Turbulent kinetic energy (k) [J/kg]: linear decrease of the values

As can be seen in the Table 4.2 the number of elements of the "M2" and "M3" meshes from the "INITIAL" group is greater than the "M2" and "M3" meshes from the "FINAL" group. The values obtained from the "INITIAL" ones are closer to each other than the "FINAL" ones, but, as mentioned before the "FINAL" meshes are a better approach to this particular problem, which supports the previous conclusion that a mesh with the same construction as the "FINAL" mesh should be constructed and more elements should be utilized in order to verify results convergence.

4.1.3 Solver

When launching Ansys Fluent the first thing to pop up is the Fluent Launcher. In the options field a double precision procedure was chosen so that better results are ensured by the solver. After entering Fluent work station, the "Setup" must be done before solving the system of equations. First of all, the solver must be chosen (Figure C.1). In this work a "Pressure-Based Solver" was used, in this method the pressure field is extracted by solving a pressure (or pressure correction) equation obtained by manipulating the momentum and continuity equations in such a way that the velocity field, corrected by the pressure, satisfies the continuity. This is called the projection method [6].

The next step consists in defining the "Model" (Figure C.2), this is based on the type of study that is going to be done relatively to the body. The "Viscous" model was chosen in order to evaluate the water flow passing around the body. Since that water offers resistance to the movement of a rigid body, this creates a Drag Coefficient (C_D), which is one of the two output variables evaluated in this work. A turbulence model must be chosen too. With the objective of getting the best results a "$k-\omega$" Shear-Stress Transport (SST) model was chosen, this way the second variable evaluated in this work, Turbulent Kinetic Energy (k) could be achieved. This model was explained in detail on the previous Chapter.

It's also important to define the materials in use on our mesh (Figure C.3), because they will condition the interactions between the body and the flow because of their physical characteristics. The flow field is fluid and is defined as liquid water with a density of

998.2 kg/m^3 and a viscosity value of 0.001003 $kg/m{\cdot}s$, being both values constant through all the simulation.

Then it's time to set the boundaries conditions of the previous characterized boundaries. For the *Inlet* boundary (Figure C.4) it was stipulated a variable named U_{in}, which corresponds to the Flow Velocity. A turbulence specification method of "Intensity and Hydraulic Diameter" was utilized. For this method it was necessary to input two parameters values: turbulent intensity (%) and hydraulic diameter (m). For this study case, the hydraulic diameter (D_h) was stipulated as 30 m because

$$D_h = \frac{4V}{A} \tag{4.1}$$

where V is the volume and A is the area. The domain is a rectangle so it results in a D_h with twice the size the height of the rectangle, that as said before is equal to 15 m.

In order to calculate the percentage of the turbulent intensity, being this parameter defined as the ratio between the root-mean-square of the velocity fluctuations to the mean flow velocity [3], Ansys Fluent suggests that for fully developed pipe flow the turbulence intensity at the core can be estimated as:

$$I = 0.16(Re_{D_h})^{-1/8} \tag{4.2}$$

being Re_{D_h} the Reynolds number based on the pipe hydraulic diameter D_h. It's value can be calculated with the following expression:

$$Re_{D_h} = \frac{U_{in} \cdot D_h}{\nu} \tag{4.3}$$

where U_{in} is the inlet velocity and ν is the kinematic viscosity of the water.

The next boundaries to be discriminated were the "*WallTop*" and the "*WallBottom*" (Figure C.9 and C.8, respectively). Both of them were described as slippery surfaces and to do that in the "Shear Condition" was introduced a "Specific Shear" condition and given the value zero Pascal to both of the main axis (O_x and O_y). This way all boundaries conditions are fully defined in order to progress in the simulation work. It's possible to observe the remaining boundary conditions definition and Reference Values in Appendix C.

After introducing all this Setup conditions, it was necessary to do the same for the Solution segment of the solver. The first thing to be done was defining the solution methods (Figure C.11). For the "Pressure-Velocity Coupling" the "Scheme" was established as "Coupled" because this method uses pressure-based coupled algorithm instead of a segregated one. It obtains a more robust and efficient single-phase implementation for steady-state flows [4]. For the "Spacial Discretization" of "Momentum", "Turbulent Kinetic Energy" and "Specific Dissipation Rate" was used the "Second Order Upwind". It provides a second order precision and face values are computed using a multidimensional linear reconstruction approach. Having worse convergence than first-order upwind scheme, it yields more accurate results, especially on triangle and tetrahedral meshes [4]. For the "Gradient" was used the "Least Squares Cell Based", in this method is assumed that the solution varies linearly and it was chosen because on irregular (skewed and distorted) unstructured meshes, the accuracy of the least-squares gradient method is comparable to the node-based gradient (and both are much more superior compared to the cell-based gradient). However, it is less expensive to compute the least-squares gradient than the node-based gradient [5]. For the "Pressure" term was used a "Second Order" scheme that reconstructs the face pressure in the manner used for second-order accurate convection terms. This scheme may provide some improvement over the standard and linear schemes, but it may have some trouble if it is used at the start of a calculation and/or with a bad mesh adequacy [4].

Afterwards, with a focus on obtaining the Drag Coefficient (C_D) and the Turbulent Kinetic Energy (k), reports for both of these output parameters must be set (Figure C.13). For the C_D report (Figure C.15) was necessary to define the wall zones where "$WallBodyBottom$" and "$WallBodyTop$" where selected in such a way that the results obtained are referred to the body in study. The type of output report was set as "Drag Coefficient" and an output parameter was created, by checking the "Create Output Parameter" option, this allows the incorporation of this analysis on the "Parameter Set" table of Workbench. For the k report (Figure C.14) the "Cell Zones" selected was Surface Body, the report type was set as "Volume Integral", the field variable set was "Turbulence" and inside this category "Turbulent Kinetic Energy (k)" was chosen. This parameter was also stipulated as an output parameter as the previous one.

Next the "Solution Initialization", the "Hybrid Initialization" method was applied (Figure C.17). This method is a collection of recipes and boundary interpolation methods. It solves Laplace's equation to determine the velocity and pressure fields. All other variables, such as temperature, turbulence, species fractions, and volume fractions, will be automatically patched based on domain averaged values or a particular interpolation recipe. Last, the "Run Calculation" (Figure C.18) as been activated and 1200 iterations were set, this way we ensure that the error is minimized and the results are the most realistic possible.

It's also possible to see the mesh representation in Ansys Fluent by defining it's display domain as shown in Figure C.19. This can be useful when we want to see a visual representation of the mesh chosen and it's adequacy to the body behaviour. Also to do the convergence analysis presented before a XY Plot of the Turbulent Kinetic Energy (k) was created as shown in Figure C.21. Lastly, the contours of the body behaviour, also for Output variable k, can be obtained in a visual representation by defining this option in the Results section (Figure C.20).

4.2 Uniform Design Method (UDM)

The main goal of this approach is to shorten the solution domain design points in order to reduce the computational time and obtain realistic solutions. After the definition of the set of variables that define the problem : Length (L), Diameter (D), Front Length (L_f), Rear Length (L_r), Front Radius (R_f), Rear Radius (R_r) and the Middle Radius (R_m); it's time to limit their values. These dimensional restrictions are based on some existing models and prototypes found in bibliography such as [21], [22], [45] and [56]. Also they were chosen in order to meet the limitations of the Ansys STUDENT VERSION, because a bigger body require more elements in mesh creation, and that could not be possible to achieve due to the limited number of elements in this student version. Also, the variables domain is restricted by some Uniform Design Method rules of application as described is this section.

The UDM table must be transformed into a hyperrectangle region corresponding to the input variable domain by linear transformation. The idea is to obtain a relationship between the design variables on the interval $[\bar{\pi} - 2\sigma_p, \bar{\pi} + 2\sigma_p]$ and the output values, being $\bar{\pi}$ the Mean Value of the Domain and σ_p the Standard Deviation [17]. In this study case the Variation Coefficient (α), defined has a constant value for each variable, was set to a value of 10 %. As we know, the Variation Coefficient is equal to the Standard Deviation (σ_p) divided by the Mean Value of the Domain ($\bar{\pi}$) of each value as Equation (4.4) shows

$$\alpha = \frac{\sigma_p}{\bar{\pi}} \tag{4.4}$$

Table 4.3: Calculation of the Minimum and Maximum values for each of the Input variables for a Variation Coefficient (α) equal to 10 %

Variables	R$_f$	L$_f$	D	L	R$_r$	L$_r$	R$_m$
Mean Value	0.5625	0.35	0.25	1	0.5625	0.4	9
Standard Deviation	0.05625	0.035	0.025	0.1	0.05625	0.04	0.9
Minimum Value	0.45	0.28	0.2	0.8	0.45	0.32	7.2
Maximum Value	0.675	0.42	0.3	1.2	0.675	0.48	10.8

The dimensional domain for each variable obtained by this method is shown beneath:

$$0.8\,m < L < 1.2\,m$$

$$0.2\,m < D < 0.3\,m$$

$$0.28\,m < L_f < 0.42\,m$$

$$0.32\,m < L_r < 0.48\,m$$

$$0.45\,m < R_f < 0.675\,m$$

$$0.45\,m < R_r < 0.675\,m$$

$$7.2\,m < R_m < 10.8\,m$$

In order to apply the UDM there is a need to divide each input variable domain in 27 equally distributed points. This can be seen in table D.1, where the method to obtain each of the points was defined by the following simple equation:

$$Design\,Point_{n+1} = Design\,Point_n + ((Design\,Point_{27} - Design\,Point_1)/26) \tag{4.5}$$

As mentioned in Chapter 2, UDM tables denoted by $U_n(q^s)$, being U the uniform design, n the number of samples, q the number of levels of each input variable, and s the maximum number of columns of the table, can be used with the purpose of applying this methodology. There is an accessory table for each UDM table, including recommendations of columns with minimum discrepancy for a given number of input parameters [17]. By acessing [13] it was possible to extract a table suited to the problem requirements (Figure D.1).

The problem in this study case has seven input variables, so by consulting the accessory table represented in Figure D.2 it's possible to realize that the columns that are meant to be used in this case are "1,2,3,5,6,7,8". By analysing the accessory table is also possible to see that the Discrepancy, D_p, associated with this method is 0.2927. After making this selection the result is Table D.2 which contains the "sequence" of values that should be used for each variable. After this table is obtained its necessary to cross reference it with D.1, that means, organizing each of the twenty seven points of the domain in the order present in each column of Table D.2. By doing this, the result is Table D.3 which exemplifies the values of the input variables to insert in Ansys Fluent.

4.3 Modeling and Simulation (MATLAB/Fortran)

4.3.1 Artificial Neural Network (ANN)

Learning procedure

The proposed ANN is organized into three layers of nodes (neurons): input, hidden, and output layers. The linkages between input and hidden nodes and between hidden and output nodes are denoted by synapses. These are weighted connections that establish the relationship between input data and output data. Deviations on neurons belonging to hidden and output layers are also considered in the proposed ANN model. In the developed ANN, the input data vector D^{in} is defined by a set of values for input variables vector $\vec{I}$, which are the input data variables from the software Ansys. The input geometrical variables of the body are Length (L), Diameter (D), Front Length (L_f), Rear Length (L_r), Front Radius (R_f), Rear Radius (R_r) and the Middle Radius (R_m). The corresponding output data vector D^{out} is composed by the output variables from the software Ansys, wich are the Drag Coefficient (C_D) and the Turbulent Kinetic Energy (k).

The data used to build the ANN needs to be normalized with the aim to avoid numerical error propagation during the learning process. Thus, the data normalization is done as follows:

$$\overline{D}_k = (D_k - D_{min}) - \frac{D_N^{max} - D_N^{min}}{D_{max} + D_{min}} D_N^{min} \tag{4.6}$$

where D_k is the real value of the variable before normalization, D_{min} and D_{max} are the minimum and maximum values of D_k, respectively, in the input/output data set to be normalized [9]. According to Equation (4.7), the data set is normalized to values, $\overline{D}_k$, that need to verify the condition:

$$D_N^{min} \leq \overline{D}_k \leq D_N^{max} \tag{4.7}$$

The sum of the modified signals (total activation) is performed through the activation function. A sigmoid function is applied on each node of the hidden layer while a linear function is considered for the output layer [17]. The activation of the kth node of the hidden layer (p=1) or output layer (p=2) is obtained through sigmoid functions as follows:

$$A_k^{(1)} = \frac{1}{1 + e^{-\eta C_k^{(1)}}} \tag{4.8}$$

$$A_k^{(2)} = C_k^{(2)} \tag{4.9}$$

where $A_k^{(1)}$ and $A_k^{(2)}$ represent the activation functions of the signal of the nodes of the hidden and output layers, respectively. The signal in each node ($C_k^{(p)}$) is defined as the components of the vector $C^{(p)}$ given by

$$C^{(p)} = M^{(p)} D^{(p)} + r^{(p)} \tag{4.10}$$

where $M^{(p)}$ is the matrix of the weights of synapses associated with the connections between input and hidden layer (p=1) or between hidden and output layers (p=2), $r^{(p)}$ represents the biases vector considered for the nodes of the hidden (p=1) or output (p=2) layers and $D^{(p)}$ is the input data vector for the hidden (p=1) or output (p=2) layer [48].

The scaling parameters, η, must be controlled because they influence the sensitivity of the sigmoid activation function. The weights of the synapses, $w_{ij}^{(p)}$, and biases in the neurons at the hidden and output layers, $b_k^{(p)}$, are controlled during the learning process. The ANN supervised learning is an optimization process based on the minimization of the error between predefined (or experimental) output data and ANN simulated results [9]. In this optimization process, the weights of the synapses and the biases in neurons are used as design variables. For each set of input data and any configuration of the weight matrix, $M^{(p)}$, and biases matrix, $r^{(p)}$, a set of output simulated results φ_i^{sim} is obtained. These simulated output results are compared with the experimental output values $\varphi_i^{exp(Fluent)}$, obtained for the same input data to evaluate the difference (or error), which must be minimized during the learning procedure aiming to obtain the optimal ANN configuration [17].

In the proposed ANN approach, several measures of the error are considered with the objective to accelerate and stabilize the learning process. The first measure is the root-meansquare error defined as

$$RMSE^{(1)} = \frac{1}{N_{exp}} \sqrt{\sum_{i=1}^{N_{exp}} \left(\varphi_i^{sim} - \varphi_i^{exp(Fluent)} \right)^2} \tag{4.11}$$

being N_{exp} the number of experiments, here associated with the set of design points of UDM application. The superscripts *sim* and *exp* denote the simulated and experimental data, respectively. The second measure of the error is the mean relative error component:

$$RE = \frac{1}{N_{exp}} \sum_{i=1}^{N_{exp}} \left[\left(\frac{\varphi_i^{sim} - \varphi_i^{exp(Fluent)}}{\varphi_i^{exp(Fluent)}} \right)^2 \right]_i \tag{4.12}$$

The biases calculated at neurons of the hidden and output layers is included aiming to stabilize the learning process:

$$\Gamma = \sqrt{\frac{1}{N_{exp}} \sum_{i=1}^{N_{exp}} \left[\frac{1}{N_{hid}} \sum_{k=1}^{N_{hid}} \left(r_k^{(1)} \right)^2 \right]_i} + \sqrt{\frac{1}{N_{exp}} \sum_{i=1}^{N_{exp}} \frac{1}{N_{out}} \left[\sum_{k=1}^{N_{out}} \left(r_k^{(2)} \right)^2 \right]_i} \tag{4.13}$$

where N_{hid} and N_{out} are the number of nodes of the hidden layer and of the output layer, respectively. The three error measures presented in Equations (4.11), (4.12) and (4.13) are aggregated throughout the Equation (4.14):

$$\Gamma_1\big(M^{(1)}, r^{(1)}, M^{(2)}, r^{(2)}\big) = c_1 RMSE^{(1)} + c_2 RE + c_3 \Gamma \tag{4.14}$$

where the constants c_k are used to stabilize the numerical differences between the three error terms. The weights of the synapses and biases are changed during the ANN learning procedure until the value of Γ_1 falls within a prescribed value. The ANN supervised learning procedure is based on the minimization of the Equation (4.14) using the weights of synapses $M^{(p)}$, and biases of neural nodes at the hidden and output layers $r^{(p)}$ as design variables. The search for the ANN optimal configuration is performed by a Genetic Algorithm $GA^{(1)}$. A binary code format is used for these variables with different digits depending on the connection between the input-hidden layers or hidden-output layers. The domain of the design variables $M^{(p)}$ and $r^{(p)}$ (p = 1 and p = 2) and scaling parameter η can be tuned together [48]. The optimization problem formulation associated with the ANN learning procedure is based on the minimization of the Equation (4.14) as follows:

$$Maximize\, FIT^{(1)} = K - \Gamma_1\big(M^{(1)}, r^{(1)}, M^{(2)}, r^{(2)}\big) \tag{4.15}$$

subject to $M^{(p)}$, $r^{(p)} \in \Omega_1$ (p = 1 and p = 2), being Ω_1 the domain of design variables in learning procedure, and $FIT^{(1)}$ is the Fitness Function. The solution of the problem defined in Equation (4.15) corresponds to the optimal ANN topology, P^{opt}_{ANN}. The constant $K^{(1)}$ must be large enough to obtain always positive fitness values [17].

Sensitivity index

After the construction and training of the Artificial Neural Network, the next step is to evaluate the importance of each of the seven input variables. That is, an attempt to achieve the best results possible, it's necessary to define which variables are more relevant to the problem, because it would be a waist of time and money investing on a geometrical parameter that has small or even no relevance in the output. With the intent to accomplish this sensitivity measure a Global variance-based method was utilized.

Among the Global Sensitivity Analysis (GSA) techniques the variance-based methods are the most appropriated [9]. In this work the variance-based method is applied to a group of input variables, namely the geometrical properties of the body and then compared with local importance measures. Assuming that $X = (X_1, \dots, X_n)$ are n independent input parameters and Ψ_m is the model function previously defined, an indicator of the importance of an input parameter X_i could be based on the outcomes of the variance of Ψ_m if X_i is fixed at its true value x_i^*: $var(\Psi_m|X_i = x_i^*)$[17]. This is the conditional variance of Ψ_m given $X_i = x_i^*$. However, in most cases the true value x_i^* of X_i is not known and then to overcome this difficulty the average of the conditional variance under all possible values for X_i denoted by $E[var(\Psi_m|X_i)]$, is calculated [48]. Considering the following algebraic property represented in Equation (4.16):

$$var(\Psi_m) = var[E(\Psi_m|X_i)] + E[var(\Psi_m|X_i)] \tag{4.16}$$

the variance of the conditional expectation $var[E(\Psi_m|X_i)]$ can be used as an indicator of the importance of X_i on the variance of Ψ_m. This indicator is directly proportional to the importance of X_i. A normalised index can be established using the conditional expectation as

$$S_i = \frac{var[E(\Psi_m|X_i)]}{var(\Psi_m)} \tag{4.17}$$

named first-order sensitivity index by Sobol [9]. Furthermore, Sobol proposed a complete variance decomposition of the uncertainty associated with Ψ_m into components depending on individual parameters and interactions between individual parameters. This procedure explains the variance $var(\Psi_m)$ as a contribution of the partial variance associated to each individual parameters or each parameter groups as

$$var(\Psi_m) = \sum_i V_i + \sum_{i<j} V_{ij} + \sum_{i<j<k} V_{ijk} + ... + V_{1,2...n} \tag{4.18}$$

where

$$V_i = var[E(\Psi_m|X_i)] \tag{4.19}$$

$$V_{ij} = var[E(\Psi_m|X_i, X_j)] - V_i - V_j \tag{4.20}$$

$$V_{ijk} = var[E(\Psi_m|X_i, X_j, X_k)] - V_{ij} - V_{jk} - V_{ik} - V_i - V_j - V_k \tag{4.21}$$

and assuming that all input parameters are independent in this approach. From this decomposition higher-order sensitivity indices can be established in particular the second-order sensitivity index as:

$$S_{ij} = \frac{V_{ij}}{var(\Psi_m)} \tag{4.22}$$

The second-order index S_{ij} defines the sensitivity of the physical response Ψ_m to the interaction between X_i and X_j, i.e. the portion of the variance of Ψ_m that is not included in the individual effects of X_i and X_j. The sum of all order indices is equal to one in case all input parameters are independent [9].

Optimization

The optimal design procedure is completely different from ANN learning process. Here, the design variables are the components of the vector $\vec{I} = [L, D, L_f, L_r, R_f, R_r, R_m]$. The process starts with a initial population $X^{(o)}$ of solutions for $\vec{I}$. The population of solutions $X^{(t)}$ is updated for each t-th generation of the evolutionary search driven by the genetic algorithm, $GA^{(2)}$. Each solution in X(t) is ranked according its fitness value, which is related with the objective function. The fitness value of each solution results from the objective value [17].

The universal form of a multi-objective design optimization problem can be mathematically defined as:

$$Minimize\, F(x) = \left[f_i\, \mathfrak{R}^n \rightarrow \mathfrak{R};\ i = 1, ..., m;\ m > 1\right],\ over\, x \tag{4.23}$$

In this particular study case, the problem can be defined as:

$$MinimizeF(x) = \left[f_1(x),\ f_2(x)\right],\ over\, x \tag{4.24}$$

where $f_1(x) = C_D(x)$ and $f_2(x) = k(x)$. This is subjected to

$$x_{inf} < x < x_{sup} \tag{4.25}$$

For each of the seven geometrical input variables. Their domain values are going to be defined later on Chapter 5.

There is not an unique solution for a multi-objective problem with more than one conflicting objectives. The obtained solutions are denoted by Pareto-optimal solutions, this concept depends on the theory of dominance according to the definitions shown bellow [48]:

- Definition 1 (Dominance): Being $Q \subset \mathfrak{R}^n$ the subset in the minimization problem formulated in Equation (4.24), a solution $x_1 \in Q$ dominates a solution $x_2 \in Q$ if the objective value for x_1 is smaller than the objective value for x_2 in at least one objective.

- Definition 2 (Pareto optimal design): Since $Q \subset \mathfrak{R}^n$ the subset in the minimization problem formulated in Equation (4.24). A solution x^Λ can be defined as a Pareto optimal design only in the situation where this solution is not dominated by any other solution $\in Q$. The group of all the Pareto solutions is called the Pareto front.

These definitions are essential to Pareto evolutionary search for multi-objective optimization of the geometry design.

The proposed approach follows the problem definition established in the previous paragraphs. The multi-objective optimization is based on the minimization of the objectives functions represented in Equation (4.24), so the performance is going to be measured by the minimization of the Drag Coefficient (C_D) and the Turbulent Kinetic Energy (k)

4.4 Exploratory Analysis

Since the beginning of the present work a extended amount of attempts were done to get the dimensional domain appropriated to the study case. This had multiple conditions to be satisfied. First, this domain should be physically possible and should be based on existing models, this required background study of existing models and prototypes for this kind of vehicle.

Second, the domain is restricted by the limit number of elements given by Ansys Student Software, this was one of the main problems encountered during this work, because the software simulations are very time consuming and it was very hard to find an acceptable domain for each of the seven variables that would not exceed the maximum number of elements given by the student version of the software. As so, this required a lot of experimentation and several hours of simulation in Ansys Fluent, because each simulation contained twenty-seven different simulations in it, in which the body geometrical variables changed, so the mesh also had to be adjusted so finally a solution could be achieved. This being said, the different combinations could easily surpass the maximum number of elements in just one of the twenty-seven points and so all the simulation needed to be repeated because the software could not achieve values to this point. This also required several adjusts to the meshing process because it was not possible to oversimplify the meshing procedure or else the results obtained would not be precise, this created the need to make changes in the mesh refinement, in all the domain, in the body of control and in the body surface. Also, the body control shape was changed in order to satisfy the software needs.

Third, the Uniform Design Method also required a parameterized domain, this meant that the values should have a standard deviation from it's mean value as explained in the previous Chapter 4, which also complicated the search for an acceptable domain. In the beginning of this work this was not taken in account, so the domain chosen for each variable was selected based in the two previous parameters mentioned. When the results from the Ansys Fluent were inserted in the ANN the sensibility obtained for each variable was not viable because the standard deviation from the mean value of each variable was not the same. This required a new search for a indicated domain, which in sum culminate in a serious amount of attempts to find an acceptable hydrodynamic simulation in a total account of thirty-four Ansys simulation runs, which represents an approximated number of five-hundred and ten hours.

The process to obtain the the expected results was also changed during the process, which also took some considerable time to execute. First approach to solve this problem was set by utilizing seven geometrical variables and one fixed velocity of 1 m/s. This process required the application of the UDM, mesh creation and search for a indicated domain for each of the seven geometrical variables Length (L), Diameter (D), Front Length (L_f), Rear Length (L_r), Front Radius (R_f), Rear Radius (R_r) and the Middle Radius (R_m).

After obtaining the Ansys Fluent simulation results for the Turbulent Kinetic Energy (k) and the Drag Coefficient (C_D) it was discussed that it would be crucial to obtain the same output variables values for different velocities. To do so, the strategy adopted was to replace one of the geometrical variables, in this case the Middle Radius (R_m), because it was assumed that this variable would not have as great impact on the output variables k and C_D as the other input variables, and defined it's value as a constant value. After defining this variable value as constant in the Design Modeler of Ansys, the Flow Velocity (U_{in}) was introduced as one of the input variables, this was done with an eye on saving computational time. This required a new application of the UDM and search for a indicated domain for each of the seven variables, because since one of the input variables had changed, the domain chosen for the other six variables was no longer valid because it surpassed the limit number of elements supported by the software.

After this simulation was done, the results for each of the output variables were obtained and could now be inserted in the Artificial Neural Network (ANN). This was done and the results have shown an over-fit of the ANN due to the major relevance of the Flow velocity (U_{in}). This means that, since U_{in} is a state variable it's importance was overwhelming when compared to the body geometrical variables, which made the ANN converge in order of this variable only. Since this happened the optimization results that could be obtained using this network would not accomplish the main goals of this work, obtaining a optimal body for each flow velocity, because the only variable that would be optimized by the network would be the flow velocity since it's relevance was so much higher than the others.

After these results were obtained it was time to take a step back and bring back the geometrical variable that was removed to insert the Flow velocity as one of the input variables. So the Middle Radius (R_m) was reinserted in the input variables group. This did not required a new search for the variables domain, nor renew the application of the UDM but it required the simulation in Ansys Fluent for different flow velocities. Since these simulations are very time consuming, the Flow Velocity (U_{in}) domain was set from one to ten meters per second. This took several hours of computer simulation in order to obtain the output variables values for the ten different velocity values. After these simulations were finished the data collected could be inserted in the Artificial Neural Network in order to obtain the results needed and that are represented in the next Chapter 5.

Chapter 5

Results

In this chapter the author intends to give the reader an insight of which were the results obtained, analyse them and take conclusions about these results. There is also a description of the path taken in order to get these results.

5.1 Computer Fluid Dynamics (Ansys)

After running Ansys Fluent the results obtained are demonstrative of the variation of the values of the Turbulent Kinetic Energy (k) and the Drag coefficient (C_D) relating to the input parameters: Length (L), Diameter (D), Front Length (L_f), Rear Length (L_r), Front Radius (R_f), Rear Radius (R_r) and the Middle Radius (R_m); and dependent on the Flow Velocity (U_{in}) that varies between 1 m/s and 10 m/s. As mentioned before some input variables will have a greater impact on the output parameters, this subject will be approached carefully and in detail later on this chapter.

After performing the simulations, it's possible to construct Tables F.1, F.2, F.3, F.4, F.5, F.6, F.7, F.8, F.9, F.10 that represent the 27 values obtained for each of the 2 output variables as a function of the 7 input variables and the 10 flow velocities.

These tables provide the information needed about the variation of the output variables C_D and k due to the input variables L, D, L_f, L_r, R_f, R_r and R_m. After these values are attained, the next area of work can begin, the development of the Artificial Neural Network. Besides this tables, is also possible to obtain contours which demonstrate visually how one of the output variables of the simulation, Turbulent Kinetic Energy (k), varies along the dimensional domain chosen. This plot is shown later in this chapter, but it's data was already used in the Mesh evaluation procedure when comparing the "INITIAL" and "FINAL" meshes (Figures 4.6 to 4.7).

5.2 Modeling and Simulation (MATLAB/Fortran)

5.2.1 Sensitivity Analysis

After the Artificial Neural Network (ANN) is up and running, the results for the sensitivity indexes can be extracted as represented in Table G.1. As explained before, this indexes are called Sobol indexes and they measure the importance of each variable in order to get the best results, i.e. the relevance of each variable to minimize the Drag Coefficient (C_D) and the Turbulent Kinetic Energy (k).

The last column of this table represents the sum of the sobol indexes for each velocity and output variable, and as can be seen, it's value always is different of one, this is due to some uncertainty involved in the process of obtaining this values, so the need to uniform the values in order to do a more precise analysis as taken the author to apply a simple method represented in Equation (5.1), which transforms Table G.1 in Table G.2

$$Sobol\,Index\,(\%) = \frac{Geometrical\,Variable\,Sobol\,Index\,*\,100}{Sum\,of\,Sobol\,Index} \tag{5.1}$$

As can clearly be seen in the Figure 5.1, that represents a graphical distribution of the global sensitivity index obtained for the Drag Coefficient (C_D) as a function of the seven different geometrical variables and the ten different flow velocities, the most relevant variables are the body Diameter (D) and the Rear Radius (R_r). This indicates that in a attempt to minimize the Drag Coefficient (C_D) these variables are the most important ones and need to have more weight than the other variables because they are going to have the greatest impact on this variable.

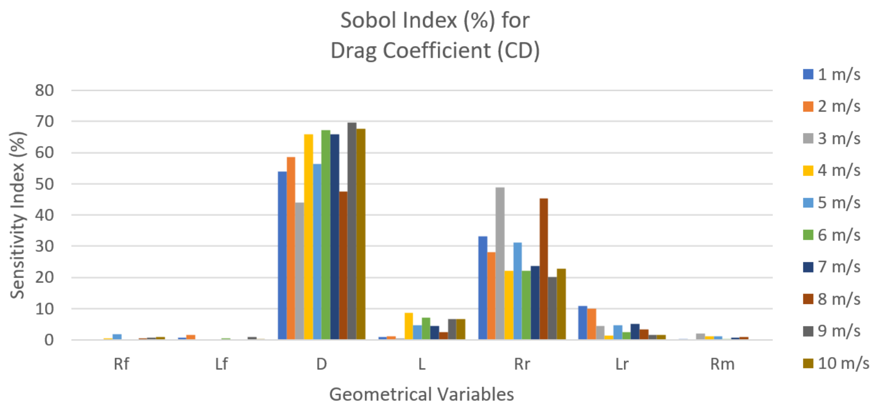


Figure 5.1: Global sensitivity index percentage value obtained for the Drag Coefficient (C_D) as a function of the the seven different geometrical variables and the ten different flow velocities

This being said, let's take a closer look to variable Diameter (D) and its relevance in the output results of the Drag Coefficient (C_D) for all the velocities. As can be seen in Figure 5.2, the variable D has its most relevant impact when the velocity is equal to 9 m/s, achieving a total value of 69.60 % (Figure 5.17). Its minor impact is when the velocity is equal to 3 m/s, achieving a total value of 43.96 % (Figure 5.11), which is still one of the two most relevant variables, besides R_r with a total value of 48.90 % (Figure 5.11).

Due to it's great relevance let's also take a look to variable Rear Radius (R_r). In Figure 5.3, this variable has its greatest values on the output variable C_D when the velocity is equal to 3 m/s (Figure 5.11), achieving a total percentage value of 48.90 %, as mentioned before. On the other hand its minor impact is when the velocity is equal to 9 m/s with a total value of 20.16 % (Figure 5.17). The maximum value for the relevance of the input geometrical to variable C_D corresponds to the minimum value for the relevance of the input geometrical variable k and vice-versa. This observation led to the construction of the plot represented in Figure 5.4. The goal was to find out if this phenomenon also happens for other velocities.

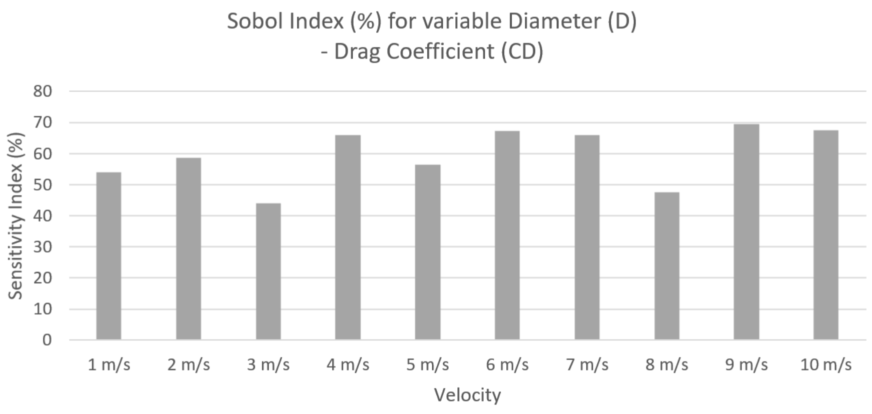


Figure 5.2: Sensitivity index percentage value obtained for the Drag Coefficient (C_D) as a function of the Diameter (D) geometrical variable and the ten different flow velocities

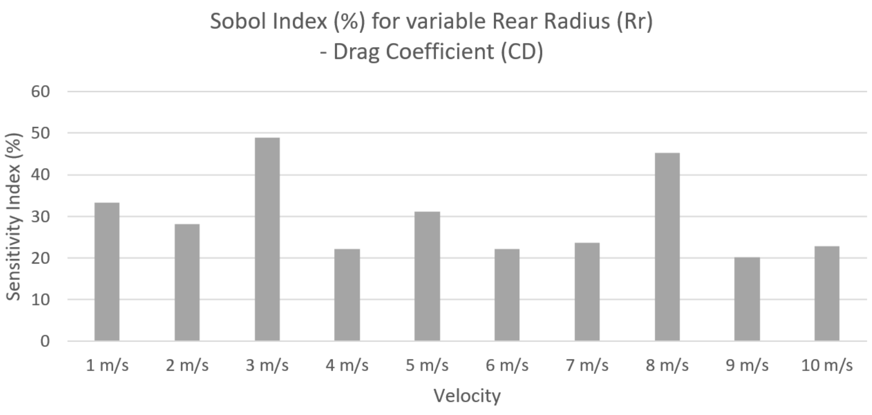


Figure 5.3: Sensitivity index percentage value obtained for the Drag Coefficient (C_D) as a function of the Rear Radius (R_r) geometrical variable and the ten different flow velocities

As can be seen in Figure 5.4 the perceptual Sobol values for the output variable C_D of the geometrical variables Diameter (D) and Rear Radius (R_r) are almost a mirror of each other. This means that when the relevance of one of them increases it's value for one specific velocity the other one decreases. This happens for every and each velocity, which results in the "mirror" representation of each other. Besides this phenomenon, it's also possible to see that these variables are always more relevant than all of the others geometrical variables.

By analysing Figure 5.5, that represents a graphical distribution of the global sensitivity index obtained for the Turbulent Kinetic Energy (k) as a function of the seven different geometrical variables and the ten different flow velocities, the most relevant variables are also the body Diameter (D) and the Rear Radius (R_r) as shown before for the output

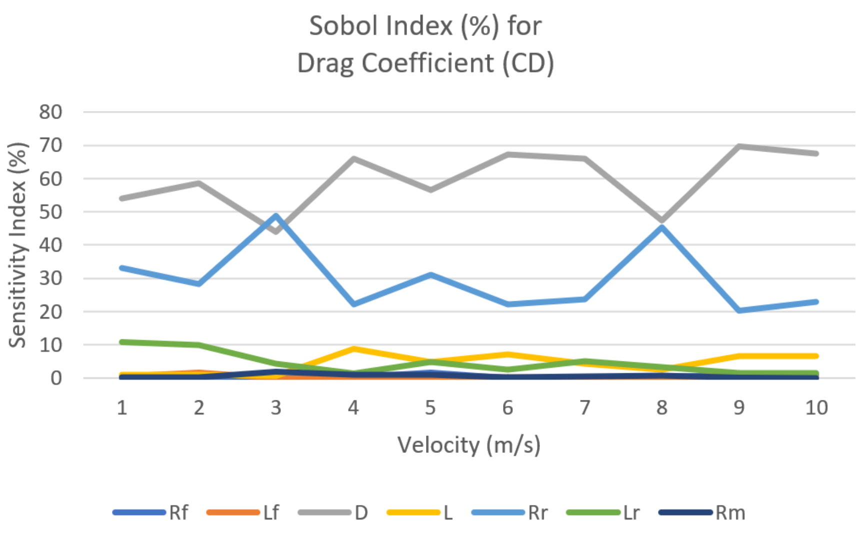


Figure 5.4: Global sensitivity index percentage value obtained for the Drag Coefficient (C_D) as a function of the seven different geometrical variables and the ten different flow velocities: Lines

variable C_D. This indicates that in a attempt to minimize the Turbulent Kinetic Energy (k) these variables are the most important ones and need to have more weight than the other variables because they dominate all the others in terms of relevance.

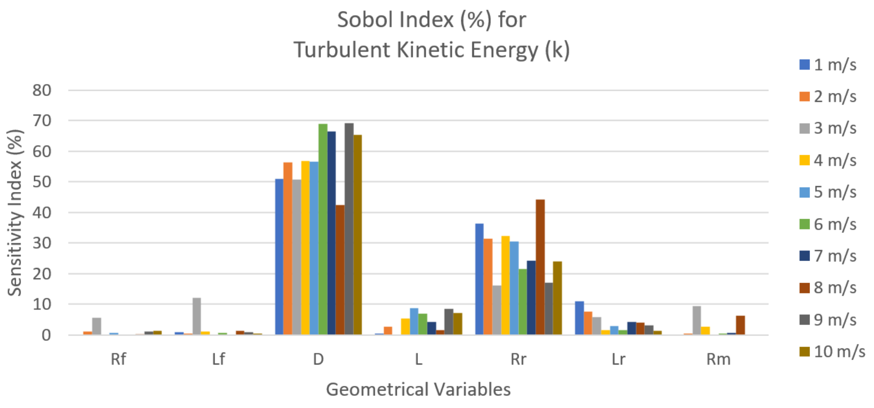


Figure 5.5: Global sensitivity index percentage value obtained for the Turbulent Kinetic Energy (k) as a function of the seven different geometrical variables and the ten different flow velocities

Doing the same analysis as done before, but now for the output variable k (Figure 5.6), the input variable D has its most important impact when the velocity is equal to 9 m/s having a total value of 69.24 % (Figure 5.17) and it's less important impact when the velocity is equal to 8 m/s with a total value of 42.41 % (Figure 5.16). This point corresponds to the highest value of the R_r variable with a total value of 44.16 %.

Due to it's great relevance let's also take a look to variable Rear Radius (R_r). As can be seen in Figure 5.7, this variable has its greatest values on the output variable k when the velocity is equal to 8 m/s (Figure 5.16), achieving a total percentage value of 44.16 % and it's minor impact is when the velocity is equal to 3 m/s with a total value of 16.20 % (Figure 5.11).

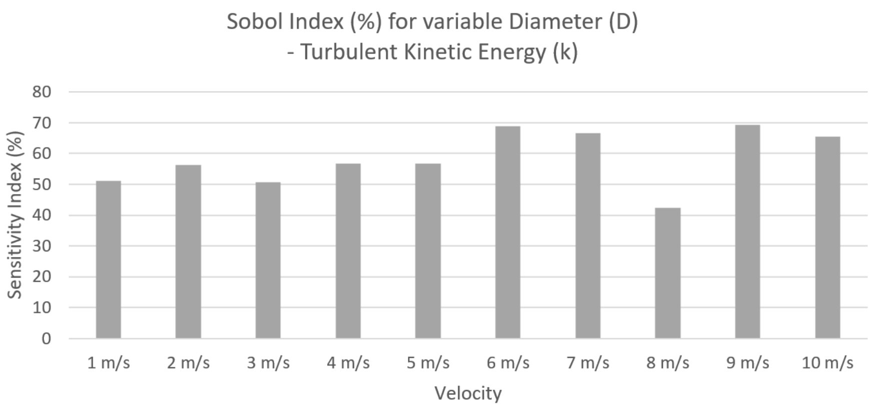


Figure 5.6: Sensitivity index percentage value obtained for the Turbulent Kinetic Energy (k) as a function of the Diameter (D) geometrical variable and the ten different flow velocities

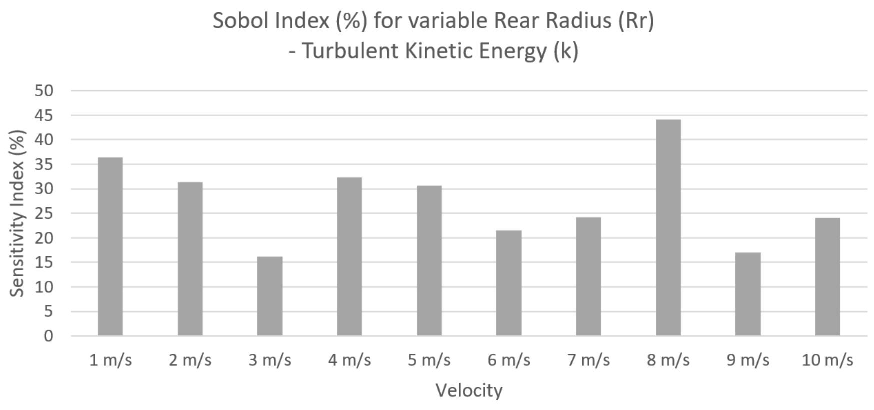


Figure 5.7: Sensitivity index percentage value obtained for the Turbulent Kinetic Energy (k) as a function of the Rear Radius (R_r) geometrical variable and the ten different flow velocities

Unlike the scenario of the output variable C_D, when we look at Figure 5.8, it's not possible to find that almost perfect "mirror effect" between the input variables D and R_r as seen before in Figure 5.4. For sure it's possible to identify that pattern for higher velocity values but not for velocity values under 5 m/s. This may be due to the fact that other input variables such as Front Length L_f and Middle Radius R_m have a certain amount of importance for the output variable k that did not have for the output variable C_D. In order to better analyse this phenomenon, the next step was to perform an analysis velocity by velocity and finding out what others variables might be relevant to the optimization procedure.

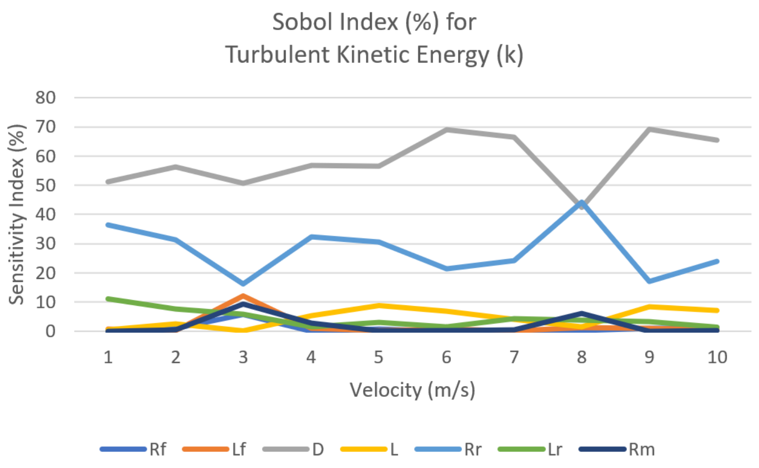


Figure 5.8: Global sensitivity index percentage value obtained for the Turbulent Kinetic Energy (k) as a function of the seven different geometrical variables and the ten different flow velocities

By analysing the values attained for Flow Velocity (U_{in}) equal to 1 m/s is possible to note in Figure 5.9 that besides variables D and R_r, variable L_r also as some relevance, achieving values equal to 10.81 % for the Drag Coefficient, C_D, and 11.10 % for the Turbulent Kinetic Energy, k. This also happens when the velocity is equal to 2 m/s (Figure 5.10), where L_r achieves a total value of 10,03 % for C_D and 7.73 % for the k variable.

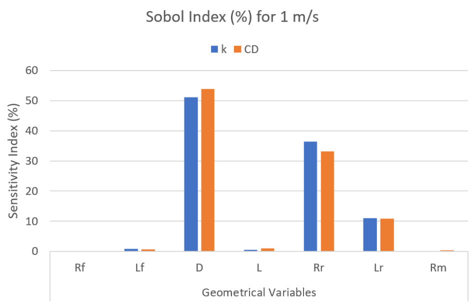


Figure 5.9: Sensitivity index percentage value obtained for velocity equal to 1 m/s for both output varibles: C_D and k

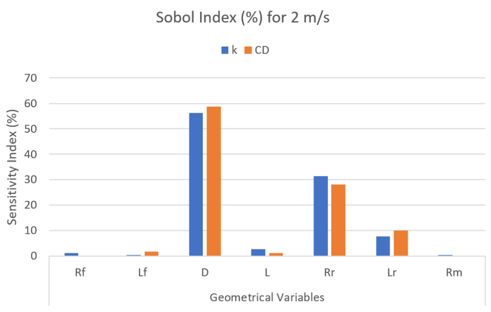


Figure 5.10: Sensitivity index percentage value obtained for velocity equal to 2 m/s for both output varibles: C_D and k

When the velocity is equal to 3 m/s (Figure 5.11), the third most important input variable for the output variable C_D is still L_r achieving a total value of 4.43 % but when it comes to the output variables k the third most important variable is Front Length (L_f) achieving a total value of 12.13%.

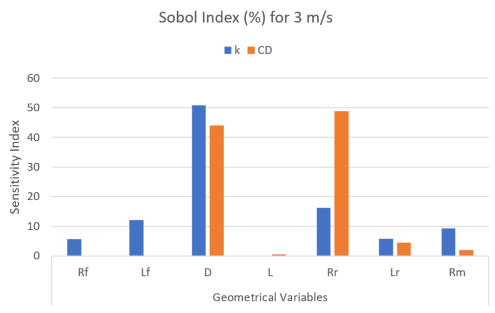


Figure 5.11: Sensitivity index percentage value obtained for velocity equal to 3 m/s for both output varibles: C_D and k

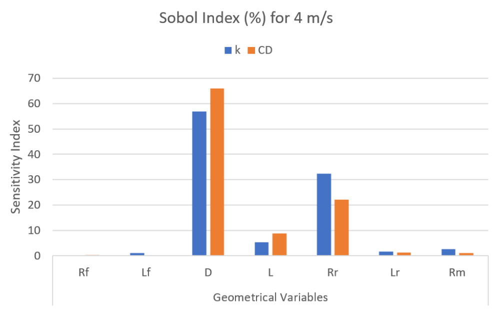


Figure 5.12: Sensitivity index percentage value obtained for velocity equal to 4 m/s for both output varibles: C_D and k

For the U_{in} value of 4 m/s (Figure 5.12) it's possible to note another change in the system, the input variable Length (L) becomes more relevant than before, turning into the new third more relevant input variable for both input variables C_D and k, achieving values for these values of 8.76 % and 5.37 %, respectively. This event repeats itself for velocities equal to 5 m/s and 6 m/s (Figures 5.13 and 5.14), although a subtle change occurs for variable C_D when the velocity is equal to 5 m/s. The L variable remains the third most relevant for this velocity, but it's sensitivity index value of 4.76 % is really near the value obtained for variable L_r of 4.75 %, which means that they have almost the same relevance and should be taken in account in a similar matter. Regarding the output variable k the scenario does not change, the input variable L is always more important.

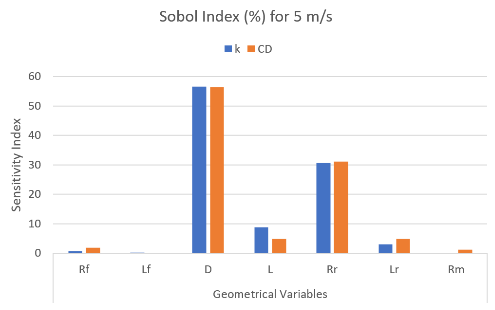


Figure 5.13: Sensitivity index percentage value obtained for velocity equal to 5 m/s for both output varibles: C_D and k

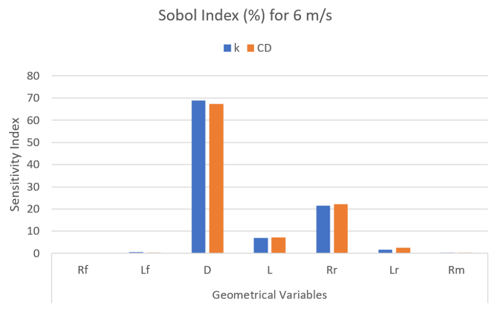


Figure 5.14: Sensitivity index percentage value obtained for velocity equal to 6 m/s for both output varibles: C_D and k

For a Flow Velocity (U_{in}) value equal to 7 m/s (Figure 5.15) both input variables L and L_r acquire similar values of relevance. Variable L attains a total value of 4.37 % for C_D and 4.16 % for k. On the other hand, variable L_r achieves a total value of 5.15 % for C_D and 4.27 % for k, surpassing again the importance of variable L, however this difference is very small, so they should be treated with similarity.

When the flow velocity is equal to 8 m/s (Figure 5.16) a new change occurs. Variable L_r becomes again the third most relevant geometrical variable when we look for a combined impact both output variables C_D and k (3.31 % and 3.93 %, respectively) but more importantly variable Middle Radius (R_m) enhanced its relevance to a considerable value of 6.19 % regarding the output variable C_D. On the other hand, its impact on k is smaller than L and L_r, achiving merely a total value of 0.88 %.

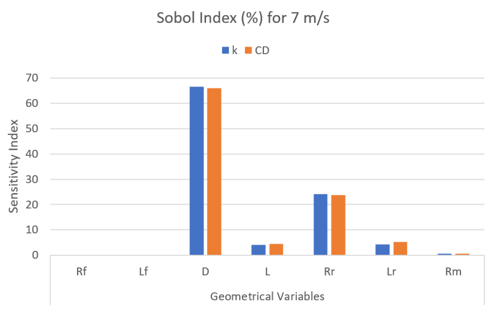


Figure 5.15: Sensitivity index percentage value obtained for velocity equal to 7 m/s for both output varibles: C_D and k

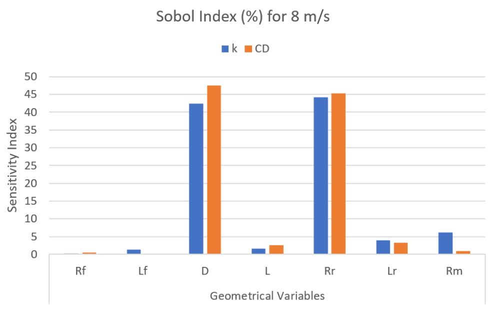


Figure 5.16: Sensitivity index percentage value obtained for velocity equal to 8 m/s for both output varibles: C_D and k

In a similar way to when the Flow Velocity (U_{in}) is equal to 6 m/s, when it achieves a value of 9 m/s (Figure 5.17) the third most important geometrical variable is L, achieving a total sensitivity index value of 6.67 % and 8.54 % for the output variables C_D and k, respectively. On the other hand when we compare the relevance of R_m to the velocity before (8 m/s), it basically disappears, becoming a non relevant variable.

Finally for a U_{in} equal to 10 m/s it is clear, that besides variables D and R_r (like on every other scenario), the geometrical variable Length (L) it's the most relevant of the remaining variables, having a total impact of 6.67 % regarding the output variable C_D and 7.12 % regarding the output variable k, as can be seen Figure 5.18.

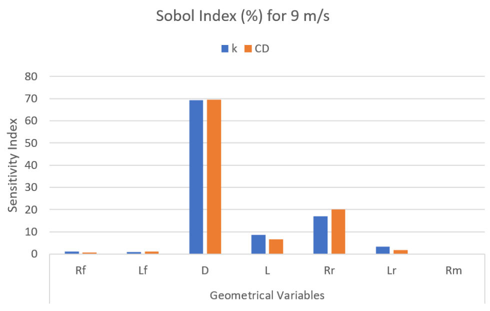


Figure 5.17: Sensitivity index percentage value obtained for velocity equal to 9 m/s for both output varibles: C_D and k

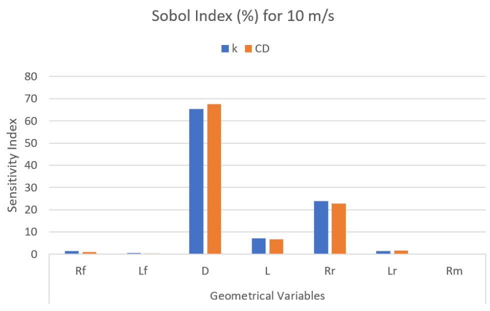


Figure 5.18: Sensitivity index percentage value obtained for velocity equal to 10 m/s for both output varibles: C_D and k

In conclusion it's possible to comprehend that the geometrical variable Rear Length (L_r) is more important when the Flow Velocity (U_{in}) has a smaller value, so its relevance decreases when the value of U_{in} increases. On the other hand, the geometrical variable Length (L) has a minor impact on the output variables C_D and k when the Flow Velocity is lower but its influence grows while the velocity increases. These two geometrical variables have opposite behaviours when regarding the flow velocity, and that is an interesting aspect to consider in the optimization process. The geometrical variables Middle Radius (R_m) and Front Length (L_f) have bursts of relevance for specific velocities but besides those velocities their importance is near 0 % for all the other situations, so it's viable to conclude that these variables are not important for optimizing the body performance and should be neglected when comparing to other geometrical variables. The same should happen to variable Front Radius (R_f), this variable should also have a smaller weight when doing the optimization process because its sensitivity index values are always low. It also as some higher value points like R_m and L_f but for this variable those highest values are always smaller than all the others variables values. This means that this variable is the least important of all the seven geometrical variables.

The previous observations are grounded by Tables G.3 and G.4, that represent the perceptual sensitivity index values for all the geometrical variables concerning all flow velocities values (1 m/s to 10 m/s).

Table G.5 represents the maximum and minimum of the Sobol index values for all the geometrical variables regarding the output variables Drag Coefficient (C_D) and Turbulent Kinetic Energy (k). It is clear that variable D is the most important one. This variable range fluctuates between 69.60 % and 43.96 % for C_D and between 69,23 % and 42.41 % for k. Next, there's variable R_r, this is the second more important variable for the body performance with sensitivity values ranging between 48.90 % and 20.16 % for C_D and among 44.16 % and 16.20 % for k. Next should be variable L_r, this variable according to Table G.5, is the third most relevant variable, achieving importance values between 10.81 % and 1.36 % for C_D and between 11.10 % and 1.38 %. Although this observation is only based on the maximum and minimum values, it does not have in consideration the variability regarding the velocity. The fourth most important value is L, this variable accomplishes values between 8.75 % and 0,55 % for C_D and 8.85 % and 0,16 % for k. The remaining geometrical variables R_f, L_f and R_m have not a considerable or substantial impact on the body performance. They have some peak values but overall their importance ranges between 0 % and 2 %, which makes them negligible when comparing to the other four geometrical variables (Figure 5.19).

In Appendix G can be found similar plots represented in this subsection, but instead of the percent values the real values are presented. There is data for the Global sensitivity index value for the Drag Coefficient (C_D) in Figure G.11 and for the Turbulent Kinetic Energy (k) in Figure G.19. There is also the distribution of the sensitivity index values for each and everyone of the geometrical variables in order of the ten different flow velocities, regarding output variable C_D (Figures G.12 to G.18) and respecting output variable k (Figures G.20 to G.26). There is also the representation of the distribution of the sensitivity index by velocity as can be seen in Figures G.27 to G.36. Besides this non-percentage values plots, there are also the representing plots of the percentage sensitivity values obtained for the C_D and k variables in order of the ten different flow velocities for each of the

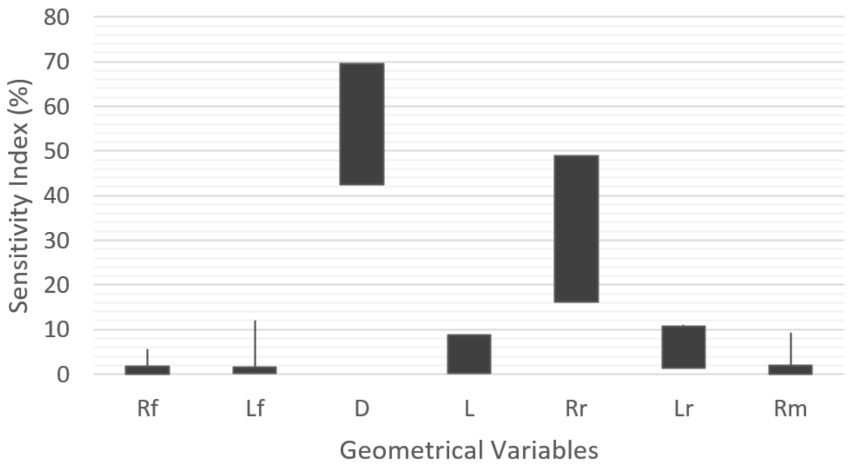


Figure 5.19: Sensitivity Index domain for each of the geometrical variables

remaining five variables that were not shown in this chapter, those can be seen in Figures G.1 to G.10.

5.2.2 Optimization

The optimization procedure has begun with the introduction of two new conditions that were implemented in the problem. First, in such a way that the body can carry a certain volume inside it, a new restriction to the input geometrical variables was introduced: $L \cdot D = 0.25\ \mathrm{m}^2$. In other words, the body Length (L) times the body Diameter (D) must always be equal to 0.25 m^2. Second, the geometrical variables domain was reduced to 90 % of it's original domain, because this way ensures that the results obtained are between the domain extreme values utilized to train the network. The mathematical formulation of the bi-objective optimization problem is defined as C_D and k minimization subject to the resulting domain (y) for each geometrical variable is represented bellow.

$$Maximize\, FIT^{(2)} = K^{(2)} - \Xi_1\, f_1(y) - \Xi_2\, f_2(y),\ over\, y \tag{5.2}$$

where Ξ_1 is equal to 0.5 Ξ_2 is also equal to 0.5, this way is established a balanced optimization. Regarding the new domain, $y_{min} < y < y_{max}$ is defined by as follows:

$$0.495\, m < R_f < 0.6075\, m$$

$$0.308\, m < L_f < 0.378\, m$$

$$0.22\, m < D < 0.27\, m$$

$$0.88\, m < L < 1.08\, m$$

$$0.495\, m < R_r < 0.6075\, m$$

$$0.352\, m < L_r < 0.432\, m$$

$$7.92\, m < R_m < 9.72\, m$$

Having this changes in consideration the Artificial Neural Network (ANN) gives a set a values for each of the geometrical variables (R_f, L_f, D, L, R_r, L_r, R_m) for the minimization of both the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) values at the same time in a double objective optimization procedure. With this concept in mind, is easy to understand that some of the found solutions might favor one of the output parameters and might neglect the other when compared to other optimal solution, i.e. one optimized solution might have the lowest value for the C_D output variable, but the same could not happen for k, it's lowest value might be on other optimized solution. This introduces the concept of Pareto efficiency or Pareto optimality which consists in a situation where no individual can upgrade it's performance without making at least one individual minimize it's performance.

In this case study the number of optimal solutions obtained vary along with the velocity domain. When the Flow Velocity (U_{in}) is equal to 1 m/s, 5 m/s, 6 m/s, 7 m/s, 9 m/s and 10 m/s the ANN has reached to one optimal solution for each of these U_{in} values. In this situation the solutions obtained are the solutions chosen as the best solution. When U_{in} is equal to 2 m/s and 8 m/s the number of optimal solutions is equal to two and as can be seen in Table H.1 the situation described before happens, one of the solutions has a smaller C_D value and the other has a smaller k value. For a Flow Velocity equal to 3 m/s was possible to find eight optimized solutions and for a U_{in} equal to 4 m/s were found seven optimized solutions. These situations require a Pareto optimality assessment, so a solution can be chosen in such a way that a balanced performance for both output variables (C_D and k) could be achieved.

But first, it's important to evaluate the ANN reliability. To do so, the optimal values for the geometrical variables were inserted in Ansys so both output variables (C_D and k) values could be calculated with an eye on comparing these values from the ones obtained from the ANN. The results obtained in the Ansys Fluent are represented in Table H.2. Table H.3 shows the percentage difference between all the solutions obtained for both output variables in both methods: ANN and CFD. As can be seen the results are very similar. The mean difference value for the Drag Coefficient is equal to 3.17 % and for the Turbulent Kinetic Energy is 0.26 %. This small discrepancy assigns credibility to the Artificial Neural Network results obtained for both variables.

As mentioned before, when the Flow Velocity (U_{in}) is equal to 1 m/s, 5 m/s, 6 m/s, 7 m/s, 9 m/s and 10 m/s the ANN has reached to one optimal solution for each of this U_{in} values, so there isn't a need to pick a solution between the domain of solutions obtained because only one is in ranking "1" of the dominance category. This can be seen in Figure 5.20, where the point closest to the origin of the plot, with the tag "1", is the best solution obtained for this flow velocity. The Drag Coefficient value for this solution is equal to 0.028379 and the Turbulent Kinetic Energy is equal to 0.30327 J/kg. On the other hand the values for the next point, with the dominance category of "2" are 0.028528 and 0.30336 for the C_D and k, respectively.

The same situation happens for the previous mentioned velocities of 5 m/s, 6 m/s, 7 m/s, 9 m/s and 10 m/s. Their plots can be seen in Figures H.1, H.2, H.3, H.4 and H.5, respectively.

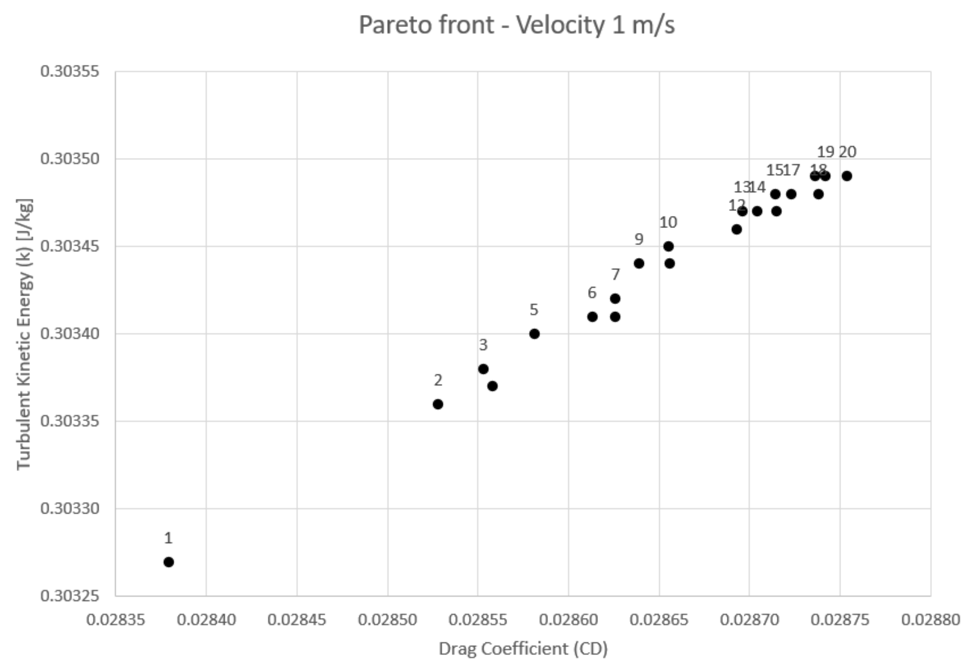


Figure 5.20: Pareto efficiency - Optimal solution distribution plot for U_{in} equal to 1 m/s

Another completely different scenario is the one obtained for U_{in} equal to 3 m/s and 4 m/s. This is due to the fact that in these cases there are several ranking "1" solutions, so the Pareto front is build upon these solutions only. The question is which one of these solutions will have a greater balanced performance for both variables (C_D and k) between the ranking "1" solutions.

First let's take a closer look to the behavior of the output variables when the Flow Velocity is equal to 3 m/s. As can be seen in Figure 5.21, the variables (C_D and k) have an opposing behaviour for this velocity. This means that when the value from one of the variables gets higher the other one gets lower and so on. The goal is to find a solution where the performance obtained for the C_D and k are as low as possible while being balanced.

Aiming to obtain that optimal solution, the Pareto front was constructed and can be seen in Figure 5.22. By analysing this plot is easy to comprehend that the best solution is the one where C_D as a value of 0.20927 and k is equal to 2.088 J/kg. This values correspond to solution eight in Table H.1 and to the closest point to the origin in Figure 5.22. The distance to the origin is given by the Equation (5.3)

$$Dist_0 = (C_D^2 + k^2)^{-1/2} \tag{5.3}$$

Now analysing the scenario for a Flow Velocity (U_{in}) equal to 4 m/s, Figure 5.23 shows that the behaviour of C_D and k has not changed. For this U_{in} value the output variables still have an opposing behaviour like the situation analysed before. So, in order to achieve

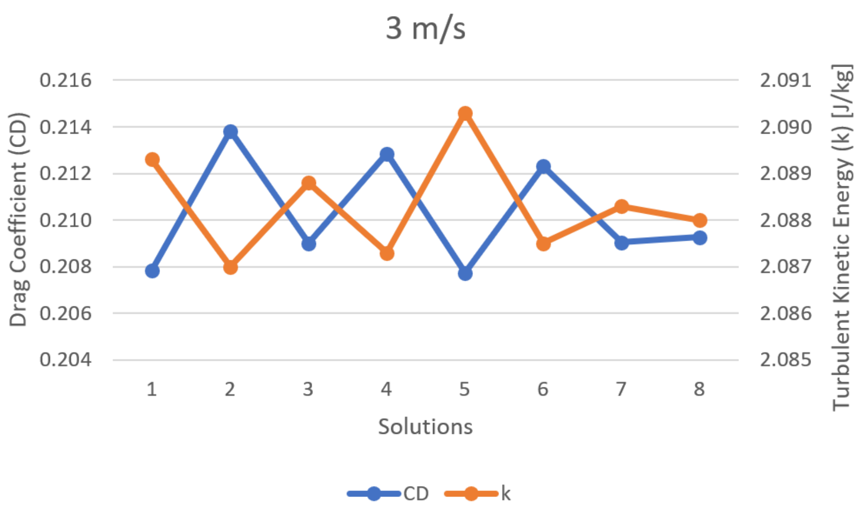


Figure 5.21: Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) values obtained for the ranking of one solution when the Flow Velocity U_{in} is equal to 3 m/s

a balanced performance point, i.e. the solution that minimizes both variables equally, without prioritizing one of them, a Pareto optimality study was done.

The plot represented in Figure 5.24 shows the Pareto front for this Flow Velocity value. The best solution is the nearest to the origin, aplying the same method as before, represented in Equation (5.3). The optimal solution point has a C_D value of 0.37252 and a k value of 3.4681 J/kg, which performs a total distance value of 3.4880. This solution corresponds to solution five in Table H.1.

The situation that occurs for a Flow Velocity equal to 2 m/s and 8 m/s is peculiar in a certain way . The ANN as achieved two solutions of ranking "1" for each of the U_{in} values so this solutions are very similar to each other in a qualitative way but one of them achieves better results for the C_D variable while the other achieves better results for the k variable. So the distance to the origin criterion will tell which one of this solutions is more balanced, i.e. the one that achieves a better performance ratio between the two output variables.

Regarding the Flow Velocity equal to 2 m/s (Figure 5.25) the optimal solution is solution one in Table H.1 with a C_D value of 0.101740 and a k value of 1.024300 J/kg, which together perform a total $Dist_0$ value of 1.029340 while the other solution has a total $Dist_0$ value of 1.029528. The other solutions represented in the top right corner in Figure 5.25 are not ranking one solutions, so their performance will always be worst then the two solution described above.

The scenario for a velocity equal to 8 m/s is similar to the situation described before. There are two ranking one solutions which are very identical to each other, although one of them achieves a better performance result for the C_D and the other for the k. The selection criterion is the same as before, the overall distance from the point to the origin (Equation 5.3). By this means the best solution acquired is solution two in Table H.1,

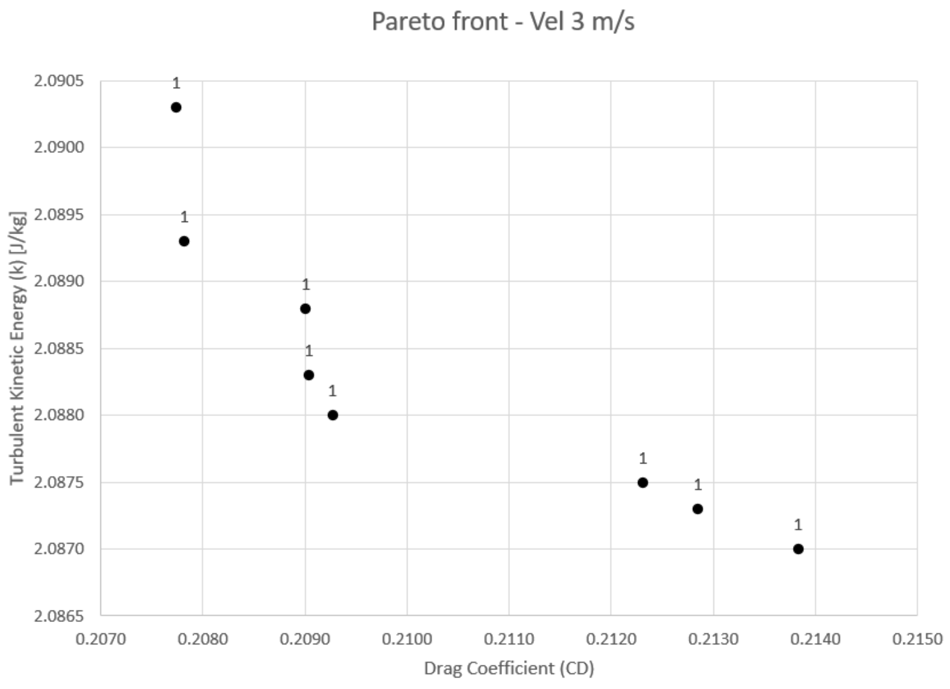


Figure 5.22: Pareto efficiency - Optimal solution distribution plot for U_{in} equal to 3 m/s

which has a C_D value of 1.2447 and a k value of 11.6540 J/kg. The total distance $Dist_0$ is equal to 11.7202 while the other solution point as a total distance of 11.7210.

After performing this analysis it's possible to build Table H.4 which represents the optimal solutions chosen as well as the geometrical variables values corresponding to these solutions. Utilizing these solutions three more tables were built.

The first Table H.5, represents the C_D and k values obtained by the Artificial Neural Network, the second (Table H.6) represents also the C_D and k values, but obtained from the CFD simulation in Ansys Fluent and the third one (Table H.7) represents the differences between the values obtained in the ANN and Ansys (like Table H.3) just for the optimal solutions chosen. Which led to the construction of the plots represented in Figures 5.27 and 5.28. The first plot (Figure 5.27) represents the difference between ANN and Ansys C_D values and as can be seen, the maximum difference is equal to 5,896 % and the minimum difference goes down to 0.0152%, which clarifies the good approximation between the optimal values confirming their veracity. The same goes to the k variable, represented in Figure 5.28, where the maximum difference is equal to 0.686 % and the minimum 0.0801 %. Is this case the difference between the results obtained using the two methods (ANN and CFD) is even smaller than the Drag Coefficient case, which supports the accuracy of the ANN once again.

In Figure 5.29 are represented the optimal geometrical bodies for each of the flow velocities values. The left column represents the optimal bodies for U_{in} equal to 1 m/s to 5 m/s and the right column for U_{in} equal to 6 m/s to 10 m/s. The reason to do this representation is to get a visual demonstration of how the body would change while changing

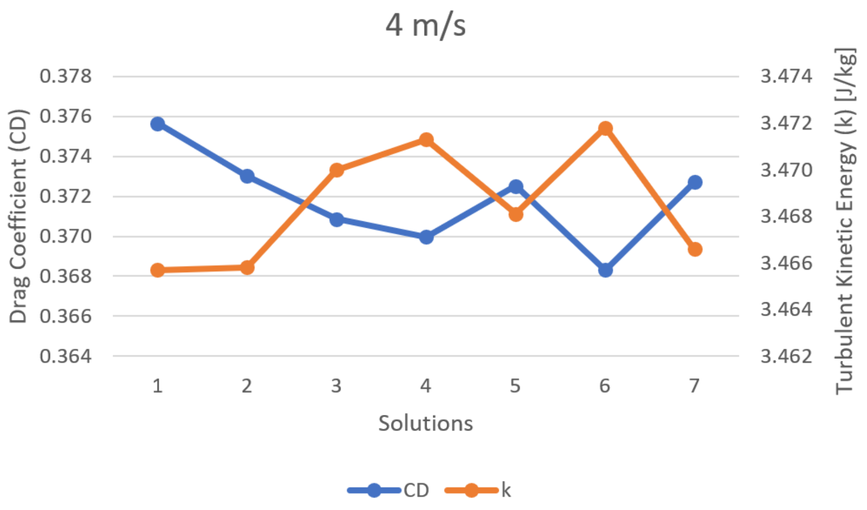


Figure 5.23: Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) values obtained for the ranking one solution when the Flow Velocity U_{in} is equal to 4 m/s

it's velocity.

In Ansys Fluent was also possible to obtain a visual representation of the distribution of the Turbulent Kinetic Energy (k) along the domain. For example, when the Flow Velocity (U_{in}) is equal to 5 m/s the representation can be seen in Figure 5.30. In this figure is possible to visualize that the highest value of this variable is accomplished after the body ends close to the rear part of the body and it soon starts to vanish. Figure 5.31 represents a close up vision of the front part of the body. On the "tip" of the body the k value is at it's lowest value, then it starts to rise slowly along the body surface until it reaches the rear part of the body (Figure 5.32) where the value of k decreases in the body surface but starts to increase exponentially (Figure 5.33) around the body rear part until it reaches it highest value. Then it starts to decrease gradually when the distance to the body rear "tip" starts to increase, until the point it reaches it's lowest value again. These plots can be seen for U_{in} equal to 1 m/s and 10 m/s in Figures H.6 to H.11.

With an eye on evaluating this optimization procedure, the next step consisted in comparing the optimized bodies for each velocity with a non-optimized body. To do so, the Design Point (DP) 25 was randomly chosen between the initial Design Points utilized to train the ANN. The method utilized to compare the performance of the optimized body with the DP25 consisted in comparing the output variables (C_D and k) values from each geometry. Since the DP25 is a non-optimal body it's C_D and k values are expected to be bigger than the ones obtained from the optimized body's, so the procedure was to quantify the difference between this values and reach the percent increase in the body performance for each velocity. For example, for a U_{in} equal to 5 m/s, the process to obtain the C_D percent increase is shown in Equation (5.4):

$$C_{D_{improv(\%)}[5\,m/s]} = \frac{\left(DP25_{C_{D[5\,m/s]}} - OPT_{C_{D[5\,m/s]}}\right)\cdot 100}{DP25_{C_{D[5\,m/s]}}} \tag{5.4}$$

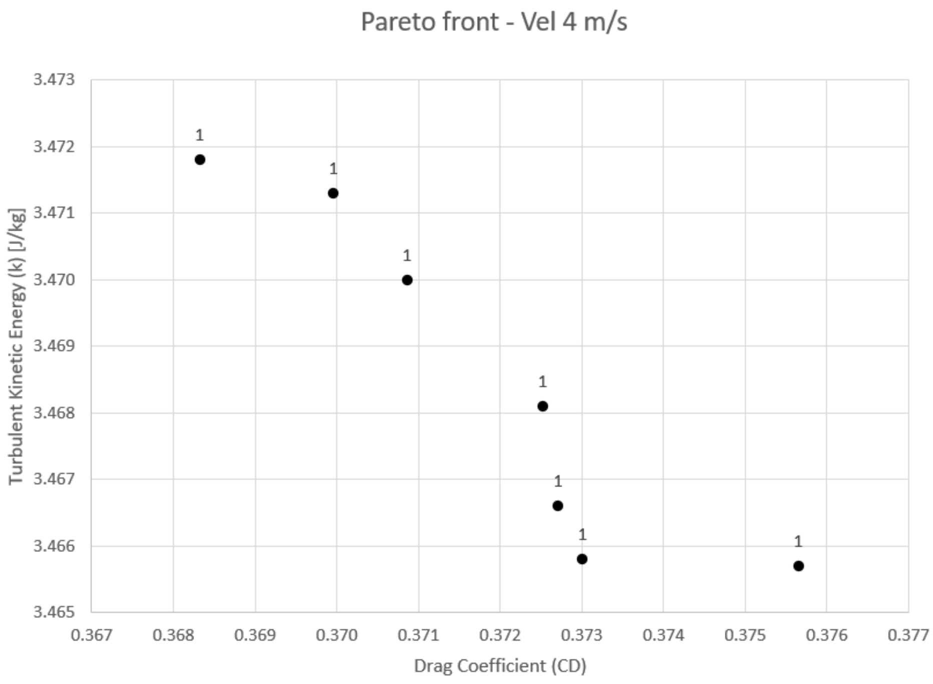


Figure 5.24: Pareto efficiency - Optimal solution distribution plot for U_{in} equal to 4 m/s

being the $C_{D_{improv(\%)_{[5\,m/s]}}}$ the Drag Coefficient performance percent increase for U_{in} equal to 5 m/s, the $DP25_{C_{D[5\,m/s]}}$ is the C_D value of Design Point 25 geometry obtained in Ansys for U_{in} equal to 5 m/s and $OPT_{C_{D[5\,m/s]}}$ the optimal geometry C_D value also obtained in Ansys for a U_{in} equal to 5 m/s.

Utilizing the same procedure, represented in Equation (5.4), for all the flow velocities for C_D and k variables it's possible to construct the plots represented in Figures 5.34 and 5.35. The plot represented in Figure 5.34 shows the percentage improvements achieved on variable C_D utilizing this optimization procedure. The mean value attained for this improvement was 15,149 %. On the other hand, for the output variable k (Figure 5.35) the mean value of improvement was 0.829 %.

Regarding the evolution of the input geometrical variables values with the variation of the Flow Velocity (U_{in}) from 1 m/s to 10 m/s the results obtained allow a first impression of what that evolution would look like, i.e. if, for example, the Diameter D values are smaller when the flow velocities are lower and it increase when the flow velocities reach greater values. This analysis was done for each one of the geometrical variables and the plots obtained can be seen in Figures H.12 to H.18. To do this investigation the optimal geometries for each Flow Velocity chosen were utilized. By doing this a picture of the way the body is going to respond to the velocity variation could be ploted.

When analysing Figure H.12, it's possible to observe a gradual, even though osciliting, increase of the Front Radius (R_f) values when the Flow Velocity (U_{in}) increases. On the other hand, the variable Front Length (L_f) (Figure H.13) values decrease when the velocity increases. Both these variables are related to the front part of the geometry and

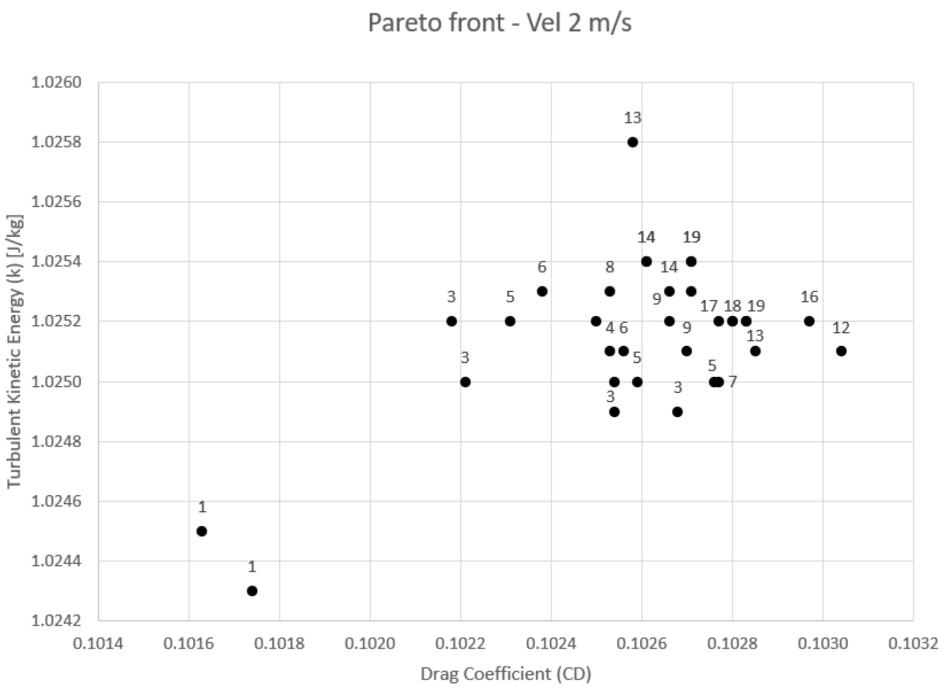


Figure 5.25: Pareto efficiency - Optimal solution distribution plot for U_{in} equal to 2 m/s

this behaviour shows the trend to a shorter and curved geometry in the front part of the vehicle.

The Diameter (D) and Length (L) of the body (Figures H.14 and H.15) tend to have a more stable behaviour, this means that they do not vary their values a lot through the process, although it can be seen that when U_{in} reaches greater value the body Diameter (D) tends to increase and the body Length (L) tends to decrease. This behaviour shows that for cruising lower velocities the body assumes a longer and thinner shape, while on higher speeds it tends to get smaller and wider. This is also supported by the behaviour of the evolution of the Middle Radius (R_m) variable (Figure H.18), which tends to increase when the velocity increases, which creates a flatter body in the top and bottom sides while for lower velocities it creates a more concave body, counterbalancing the Diameter increase and decrease.

Regarding the rear part of the geometry, it's possible to detect through the evolution of variables Rear Radius (R_r) and Rear Length (L_r) (Figures H.16 and H.18, respectively) that the body tends to get shorter and concave, approximating it's rear part to a cone. This observation is supported by the fact that the Rear Radius (R_r) values tend to decrease when the Flow Velocity (U_{in}) attains greater values, transforming the tail of the body in a more concave shape than when the velocity values were lower. The rear part also tends to get smaller, this is based upon the fact that the value of the Rear Length (L_r) tends to get smaller when the Flow Velocity increases.

After all this observations is possible to conclude that for a cruising way of navigating, the optimal form of the body should be longer, flatter and thinner. Contrastingly, when

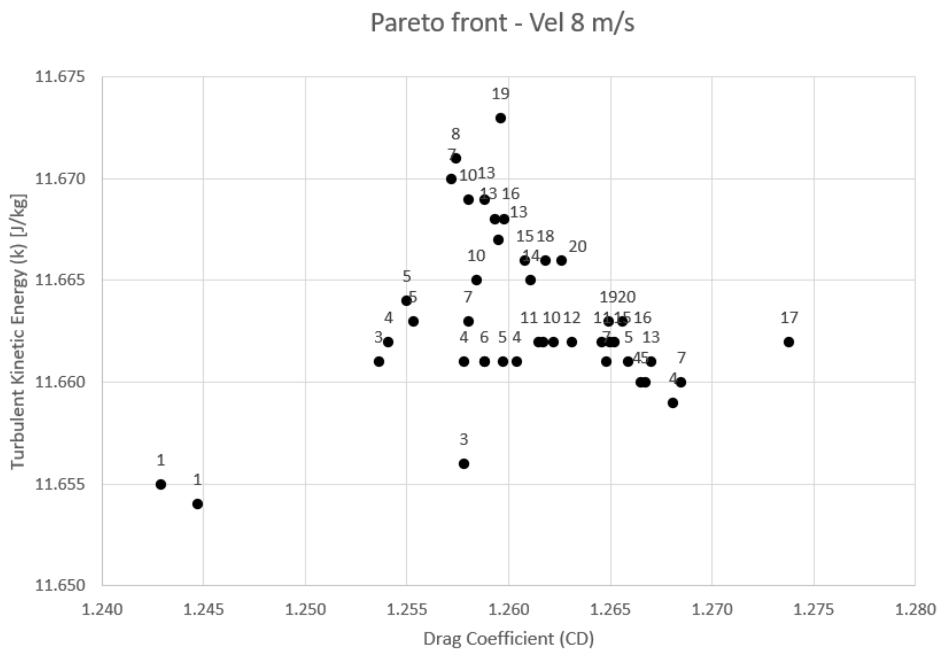


Figure 5.26: Pareto efficiency - Optimal solution distribution plot for U_{in} equal to 8 m/s

the goal is to achieve a higher locomotion velocity the body tends to get smaller, wider and more curved or concave. This transformation allows the body to excel its performance when changing velocities, through a continuous and gradual change.

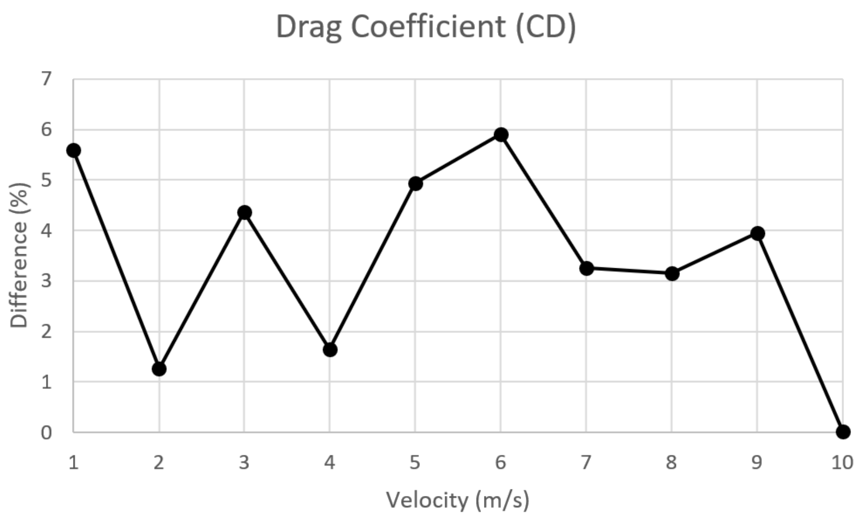


Figure 5.27: Percentage difference between the values obtained for the C_D using the ANN and CFD methods

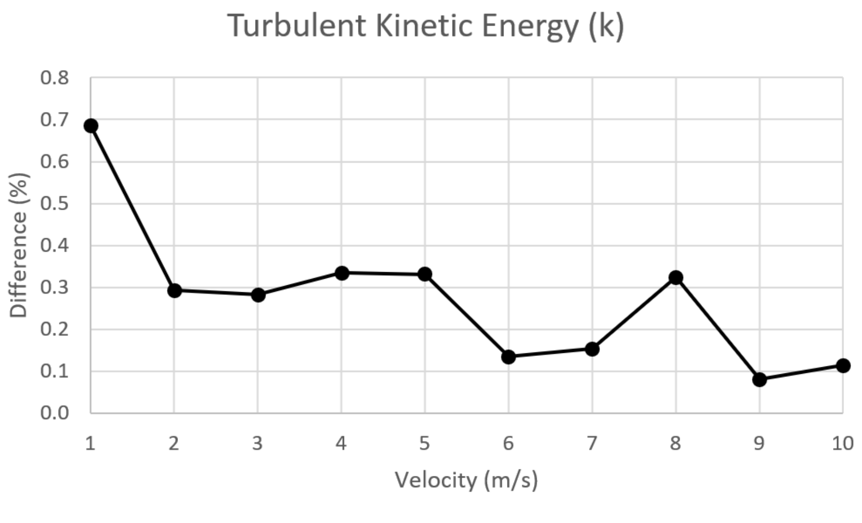


Figure 5.28: Percentage difference between the values obtained for the k using the ANN and CFD methods

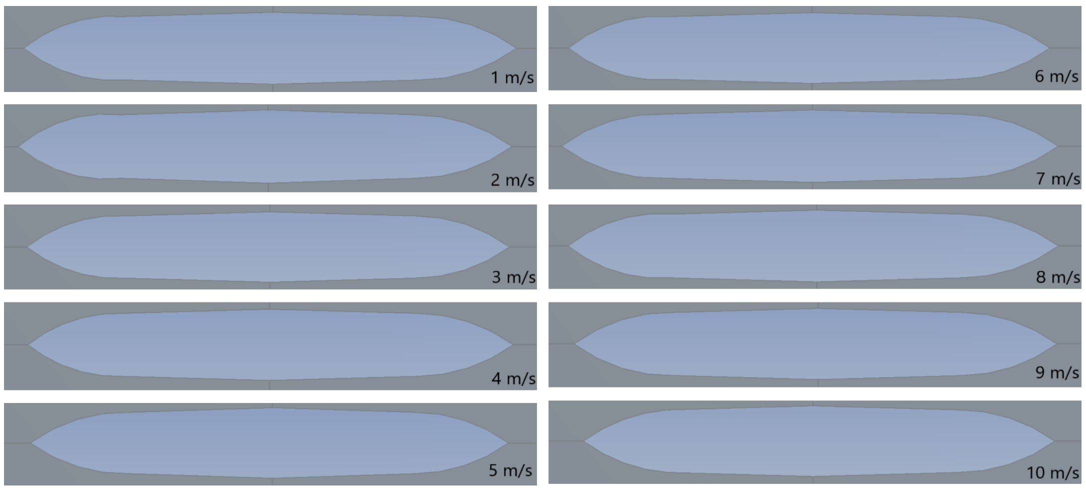


Figure 5.29: Optimized geometries for each of the Flow Velocity U_{in} values

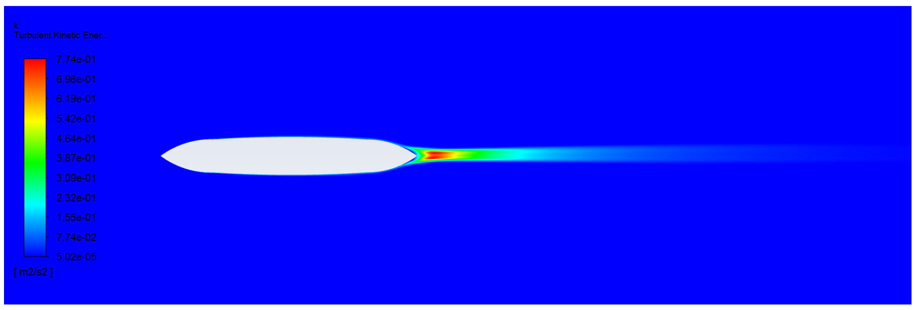


Figure 5.30: Turbulent Kinetic Energy (k) contour - Full body (5 m/s)

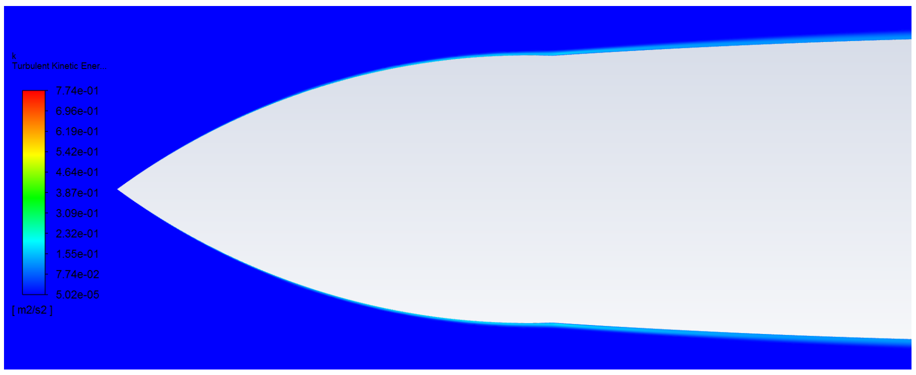


Figure 5.31: Turbulent Kinetic Energy (k) contour - Front part of the body (5 m/s)

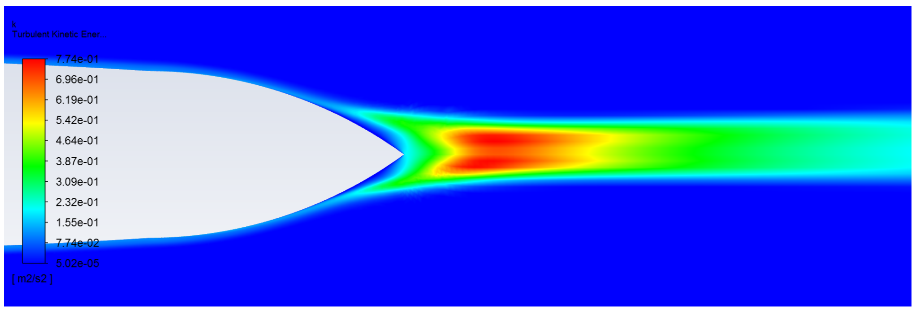


Figure 5.32: Turbulent Kinetic Energy (k) contour - Rear part of the body (5 m/s)

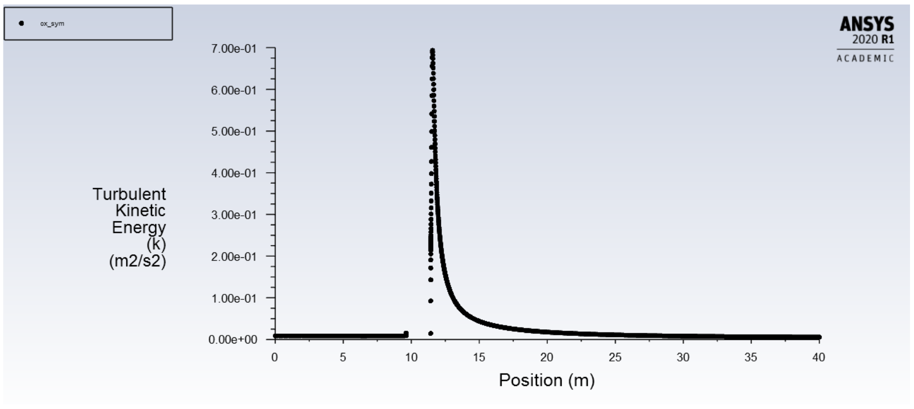


Figure 5.33: Turbulent Kinetic Energy (k) value along the domain - 5 m/s

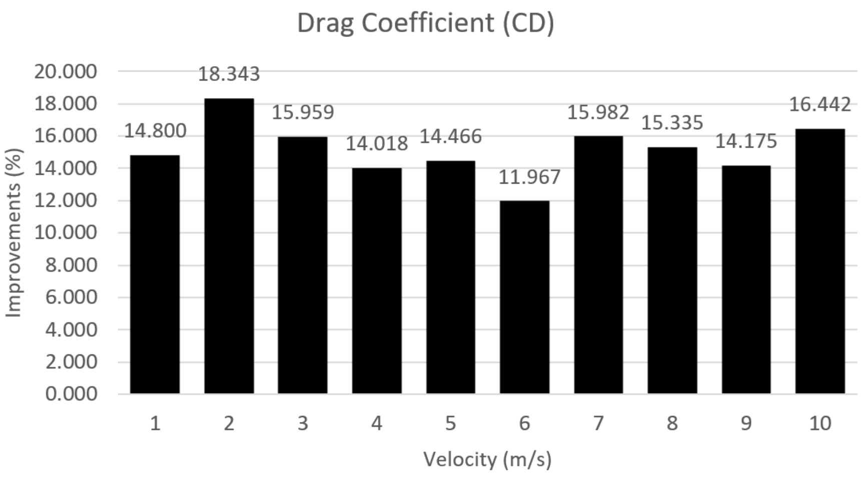


Figure 5.34: Percentage improvements achieved for the C_D - Optimized body versus non-optimized body (DP25)

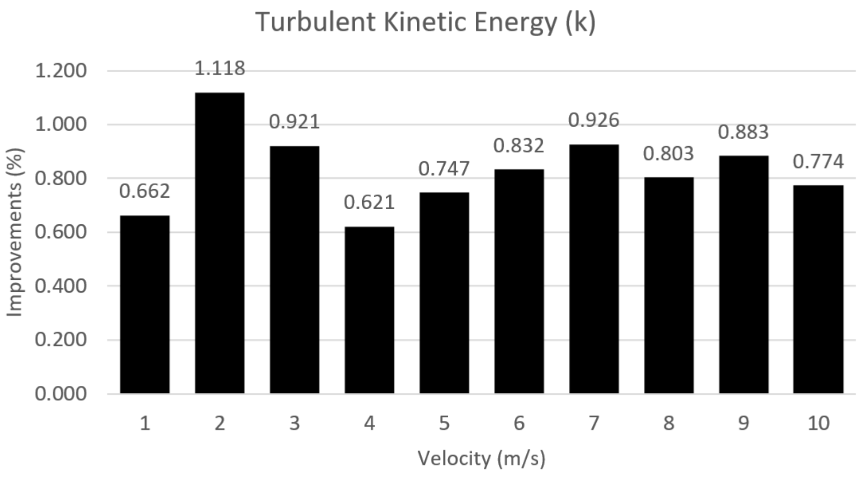


Figure 5.35: Percentage improvements achieved for the k - Optimized body versus non-optimized body (DP25)

(This page was intentionally left blank)

Chapter 6

Conclusion and Future Work

The objective of this research was to do a hydrodynamic analysis of an Autonomous Underwater Vehicle (AUV) with the ability to change and adapt it's shape to different requirements. With the scope to do that a CFD and an optimization studies were necessary. The optimization procedure achieved good results with a mean error of 3.41 % for the first objective (C_D) and 0.27 % for the second objective (k). This discrepancy means that the ANN has learned and adapted well and the results are viable.

The goal was achieved by reaching to ten different geometries that excel their performance for each of the ten flow velocities. The geometrical variables chosen show a relevant impact on the body performance, mainly the Diameter (D) and the Rear Radius (R_r) of the body, these variables couldn't be neglected when designing this type of vehicle. The optimal geometries attained show considerable improvements comparing to a non-optimal geometry, accomplishing the study case mission. Regarding the Drag Coefficient (C_D) the mean improvement was 15.2 % and for the Turbulent Kinetic Energy (k) the improvement was 0.83 %. The number of solutions is small and could not provide a clear view of the evolution of the geometrical variables through the increase of flow velocity. Despite of that, these developments show the benefits of utilizing a morphing geometry instead of a fixed geometry.

Regarding the CFD study the method adopted was sufficient to provide good and stable results, although the utilization of a non-student version could provide better results due to a non-existing limitation of elements which hands over the possibility of a greater mesh refinement and a bigger domain for the input variables. That would culminate in more accurate and sizable simulations and results. Still concerning this procedure a vaster domain for the velocity field would grant a further knowledge of the body morphing procedure. The geometry used is a simplification of a real vehicle, further studies could embrace a tri-dimensional geometry that would mimic, in a more realistic way, a real world vehicle. Also different propulsion methods could be examined to determine which one of them better applies to each requirement.

The relationship between the output variables (C_D and k) behaviour couldn't be established for all the flow velocity field, except for the velocities of three and four meters per second, where their behaviour was opposed to one another. This could be upgraded by selecting different parameters along the optimization procedure.

Appendix A

Ansys Design Modeler

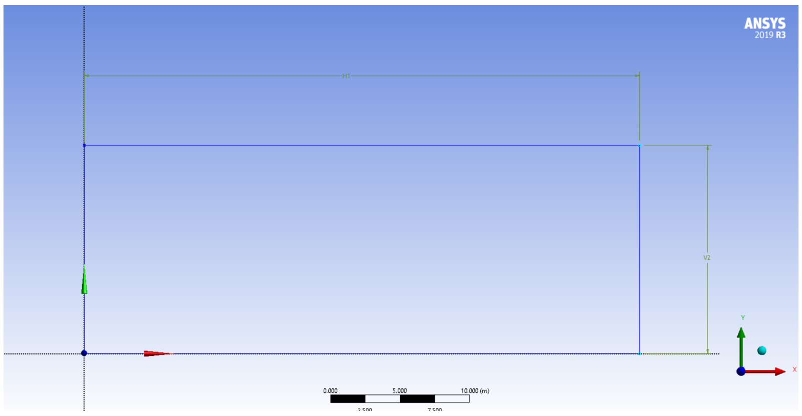


Figure A.1: Geometry design: domain

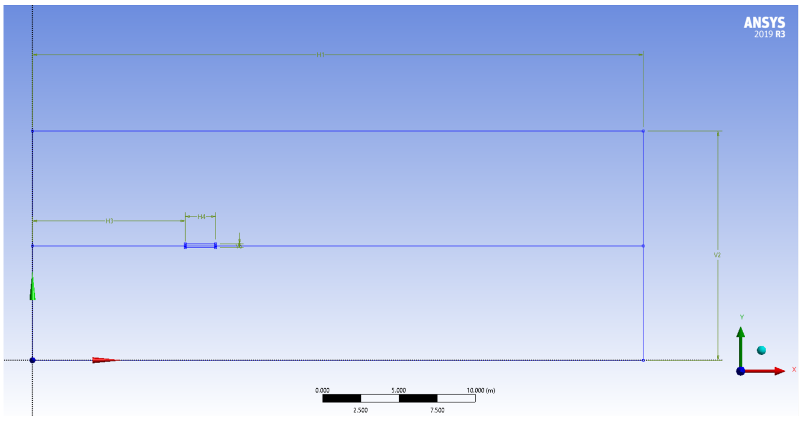


Figure A.2: Geometry design: body representation in the domain

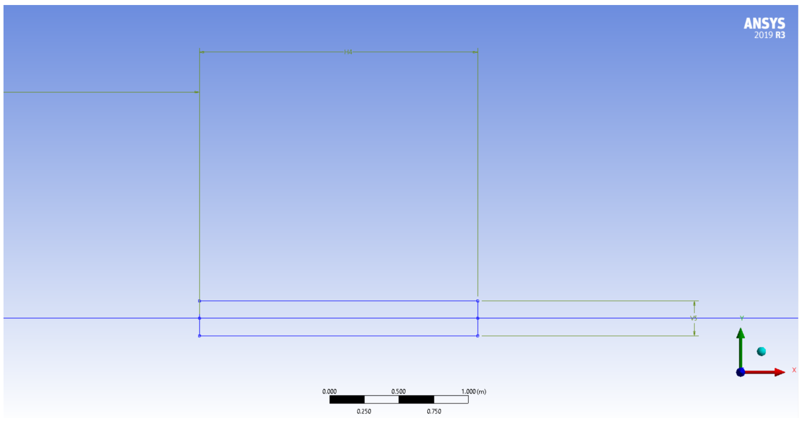


Figure A.3: Geometry design: definition of Length (L) and Diameter (D)

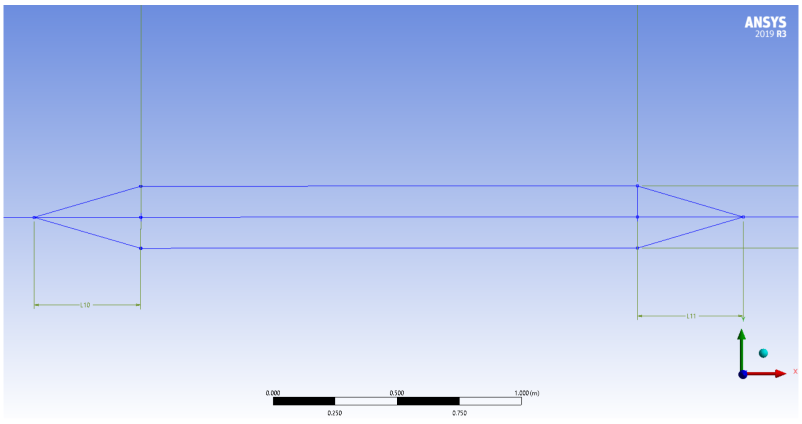


Figure A.4: Geometry design: definition of the Front Length (L_f) and Rear Length (L_r)

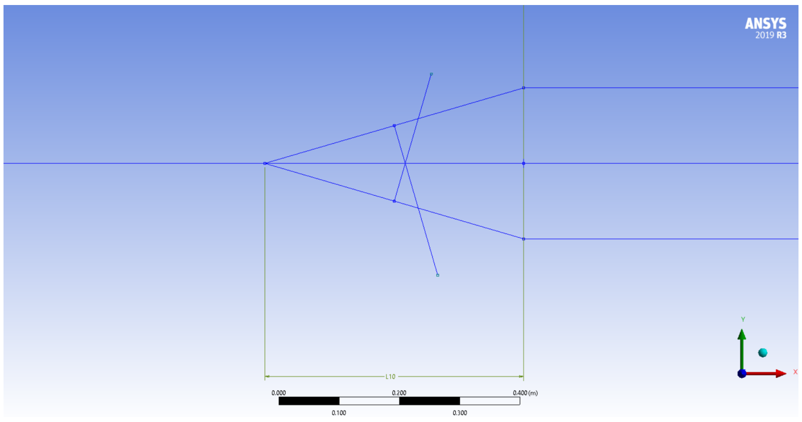


Figure A.5: Geometry design: perpendicular lines to radius definition (front part)

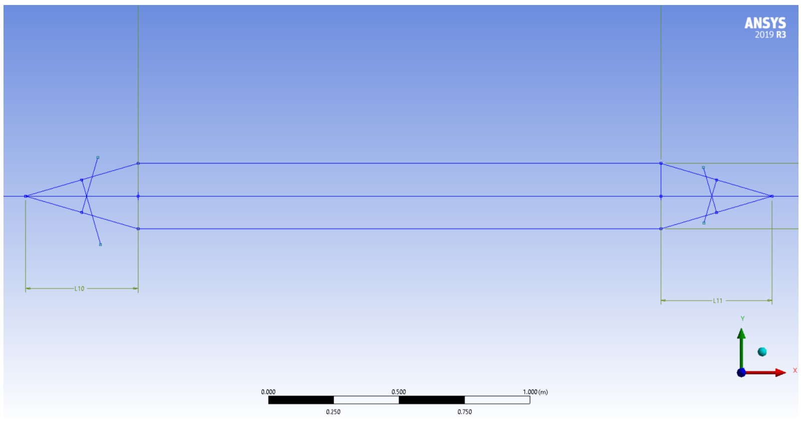


Figure A.6: Geometry design: perpendicular lines to radius definition (full body)

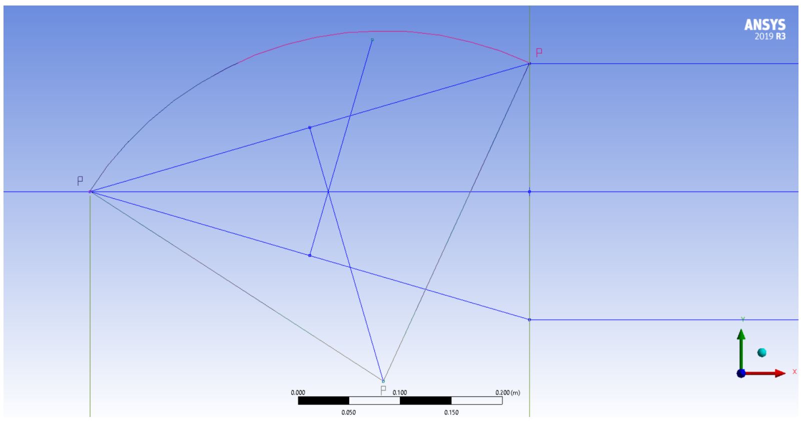


Figure A.7: Geometry design: front Radius (R_f) partial schematic draw

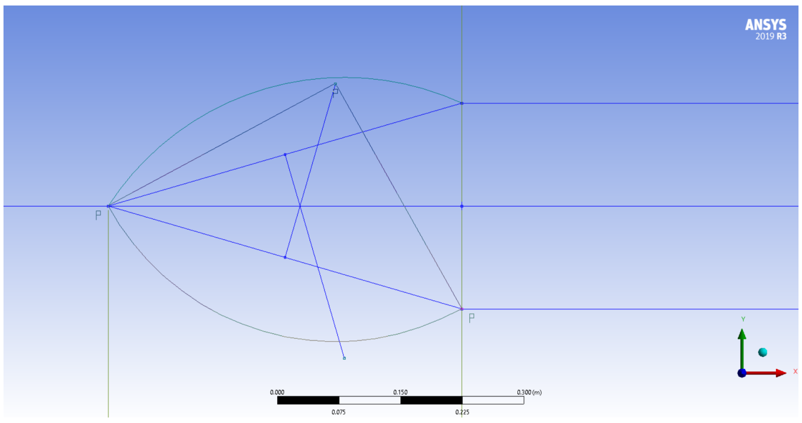


Figure A.8: Geometry design: front Radius (R_f) schema

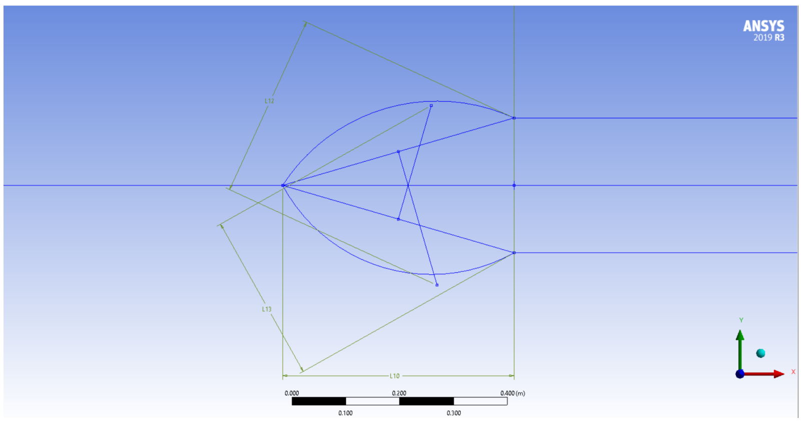


Figure A.9: Geometry design: front Radius (R_f) schema with dimensions defined

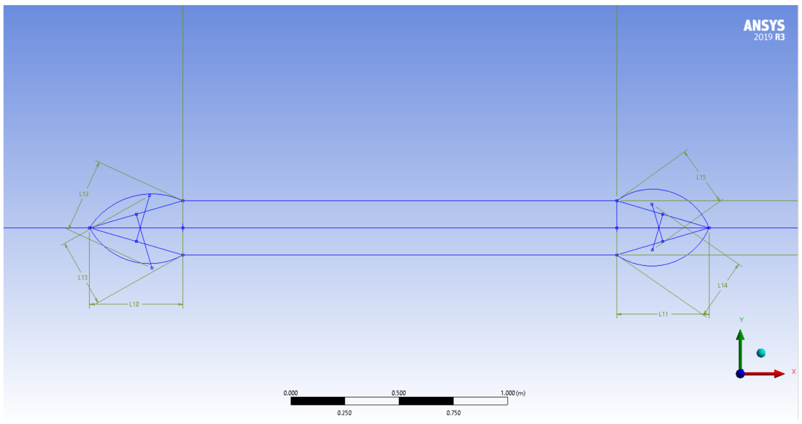


Figure A.10: Geometry design: front Radius (R_f) and Rear Radius (R_r)

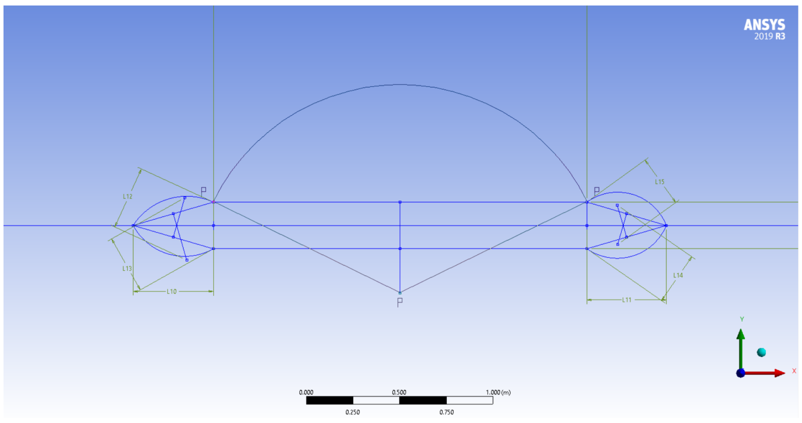


Figure A.11: Geometry design: middle Radius (R_m) partial schema

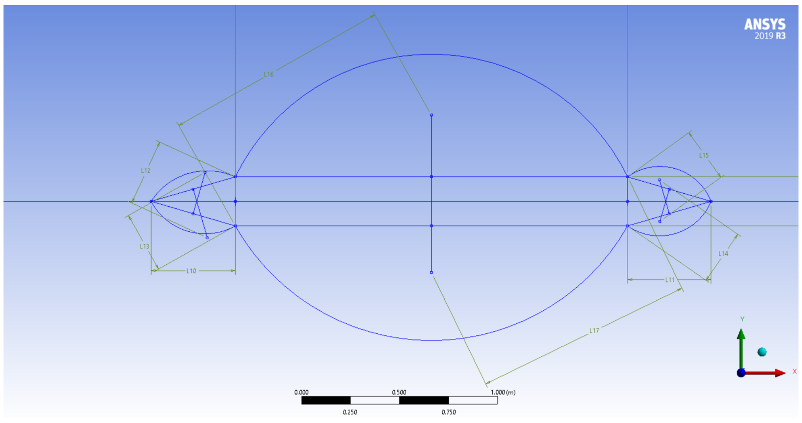


Figure A.12: Geometry design: full body parameters defined

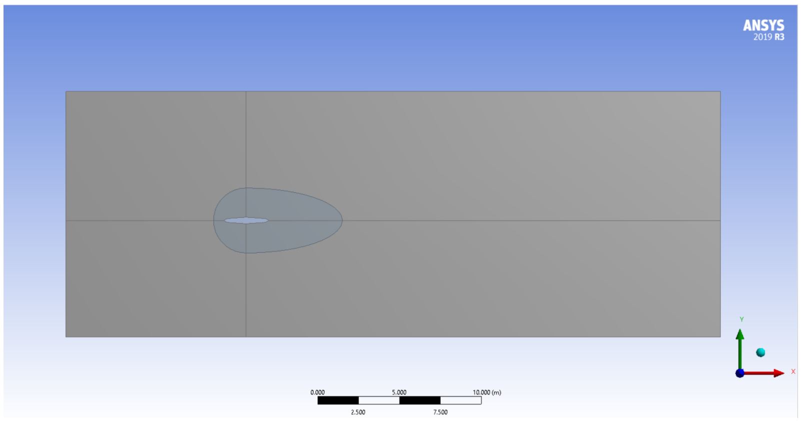


Figure A.13: Geometry design: body of control for "INITIAL" meshes

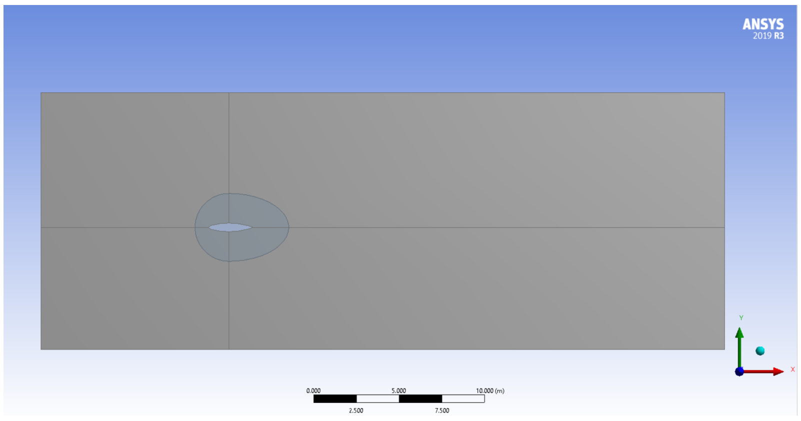


Figure A.14: Geometry design: body of control for "FINAL" meshes

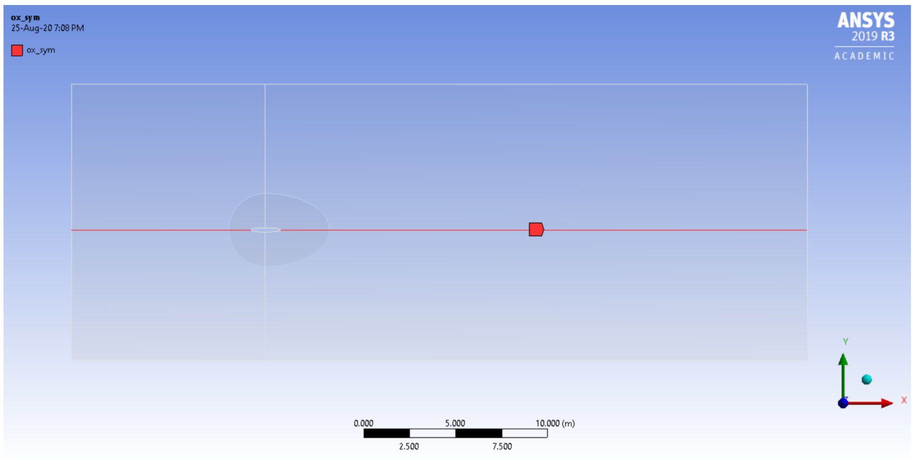


Figure A.15: Geometry design: O_x symmetry definition

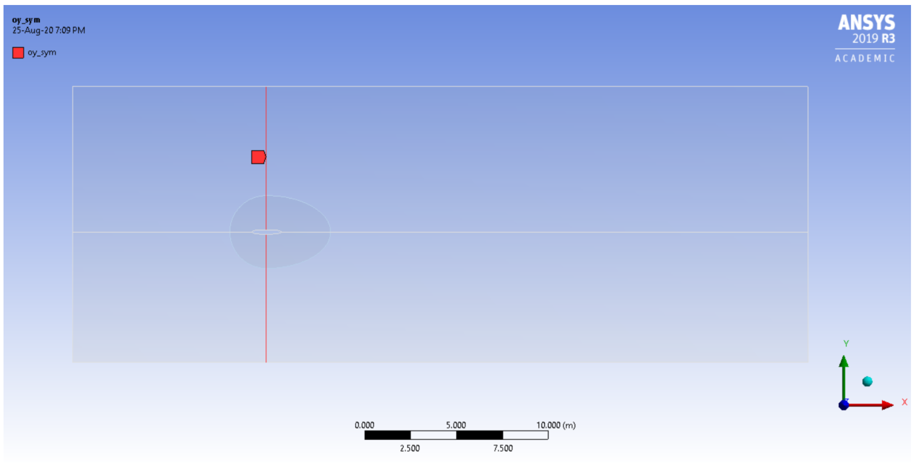


Figure A.16: Geometry design: O_y symmetry definition

Appendix B

Mesh Construction

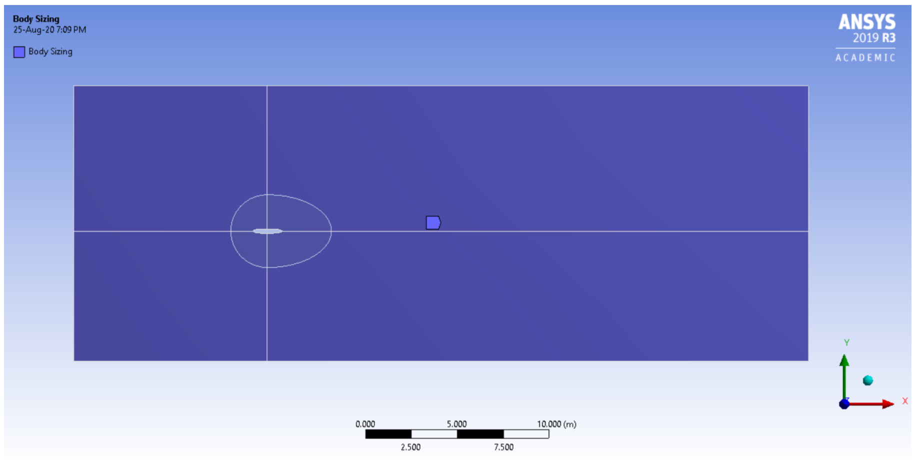


Figure B.1: Mesh Construction: Body Sizing

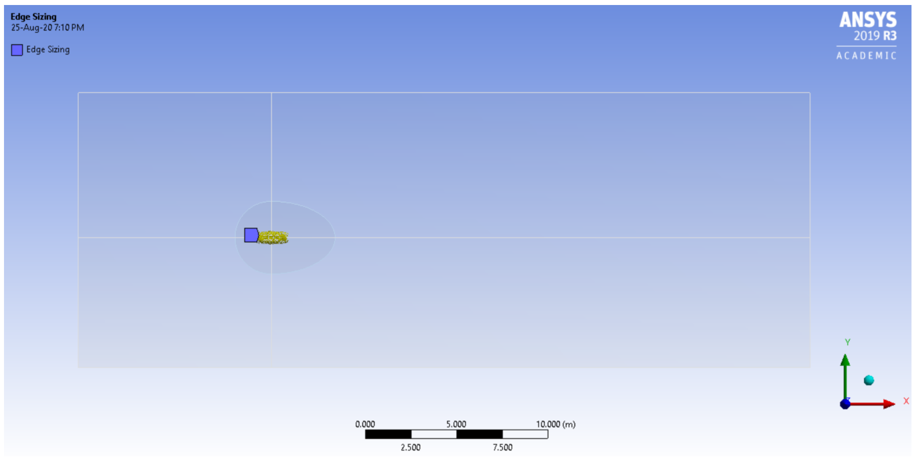


Figure B.2: Mesh Construction: Edge Sizing (body)

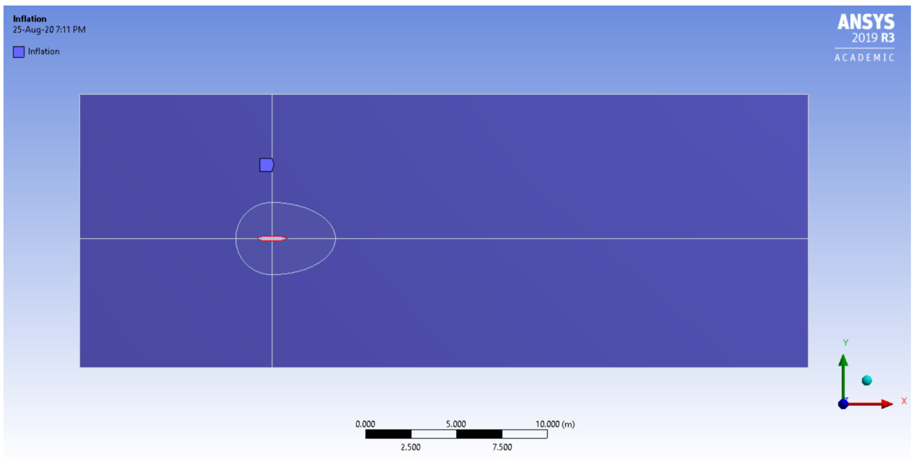


Figure B.3: Mesh Construction: Inflation

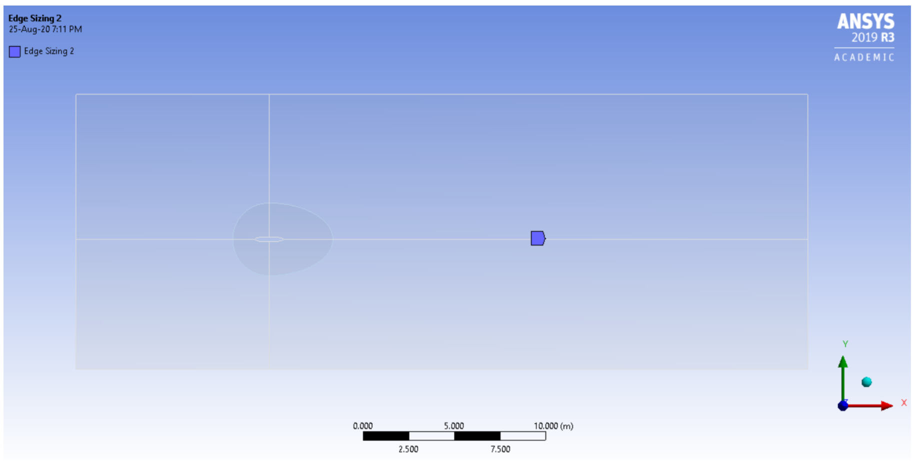


Figure B.4: Mesh Construction: Edge Sizing (tail)

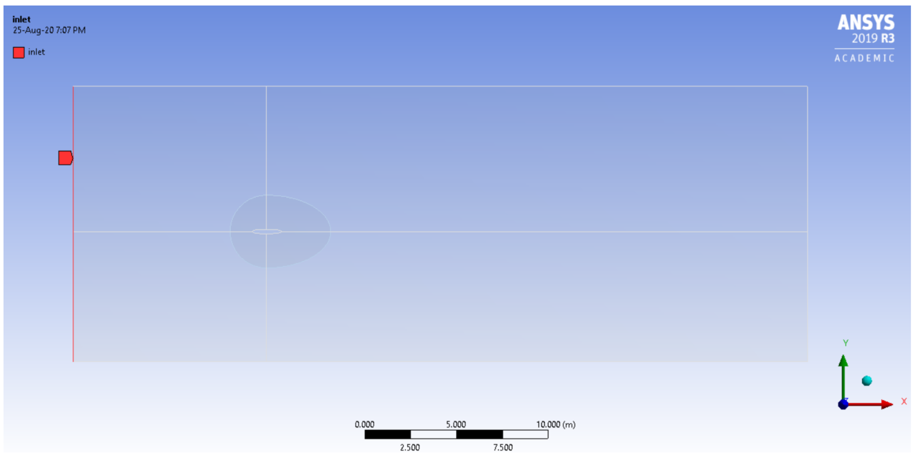


Figure B.5: Boundary definition: Inlet

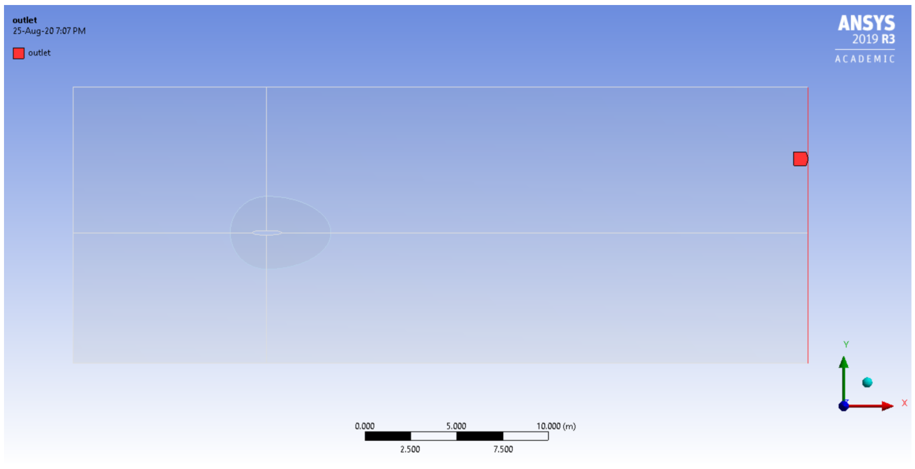


Figure B.6: Boundary definition: Outlet

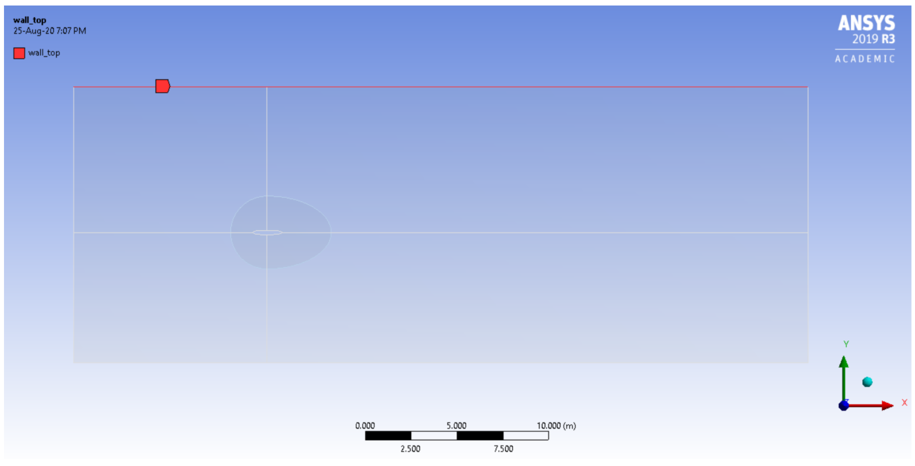


Figure B.7: Boundary definition: Wall Top

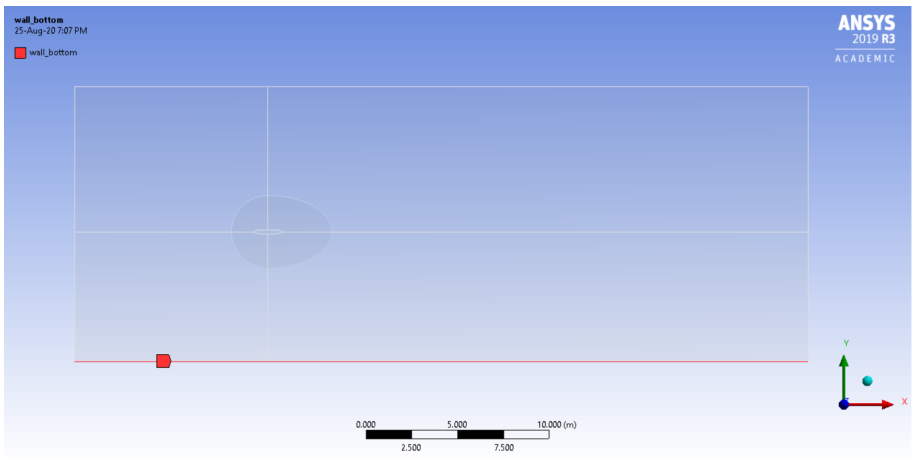


Figure B.8: Boundary definition: Wall Bottom

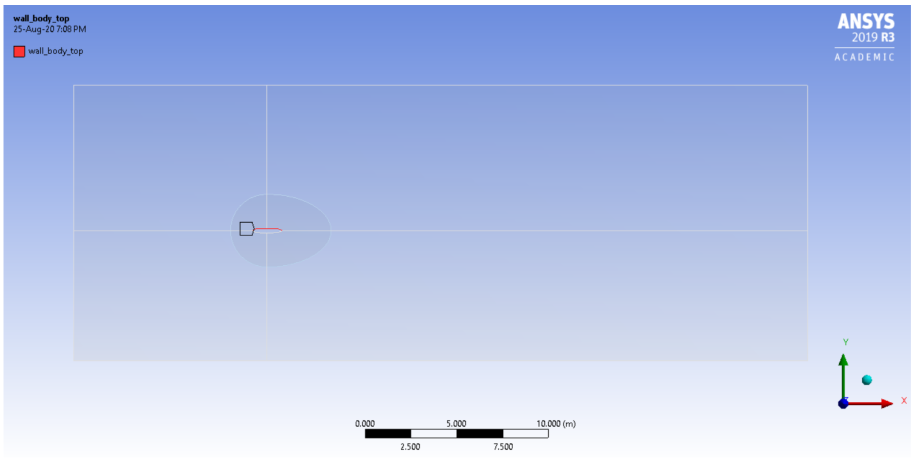


Figure B.9: Boundary definition: Wall Body Top

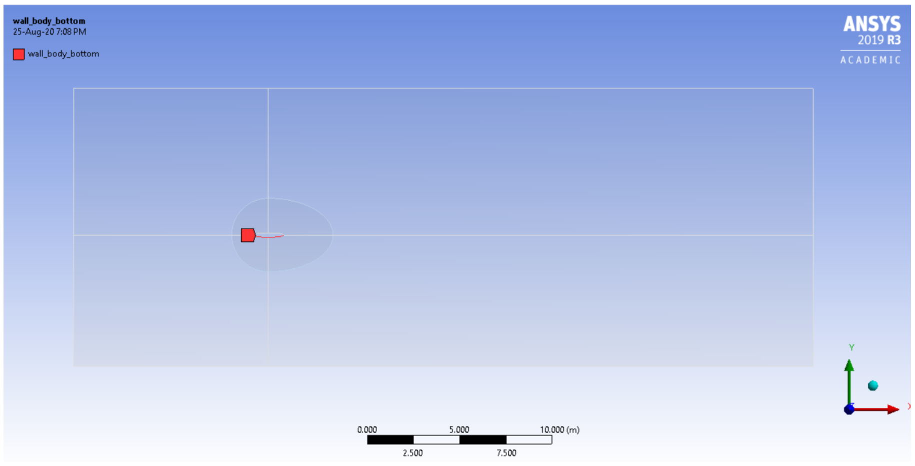


Figure B.10: Boundary definition: Wall Body Bottom

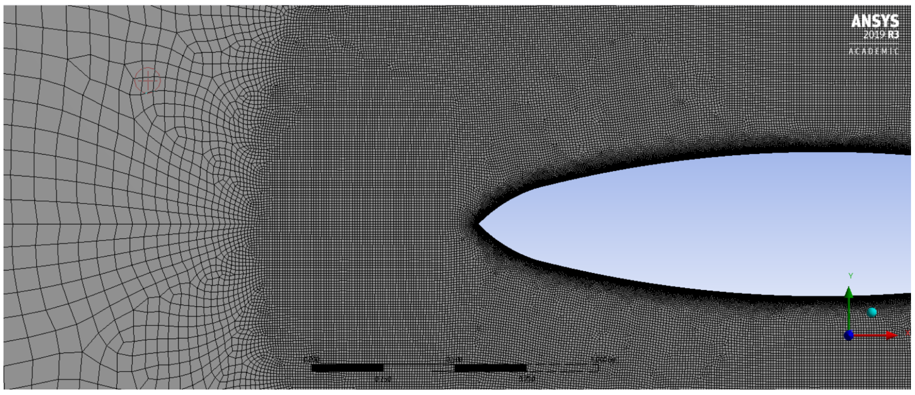


Figure B.11: Final Mesh closeup: Front part of the body

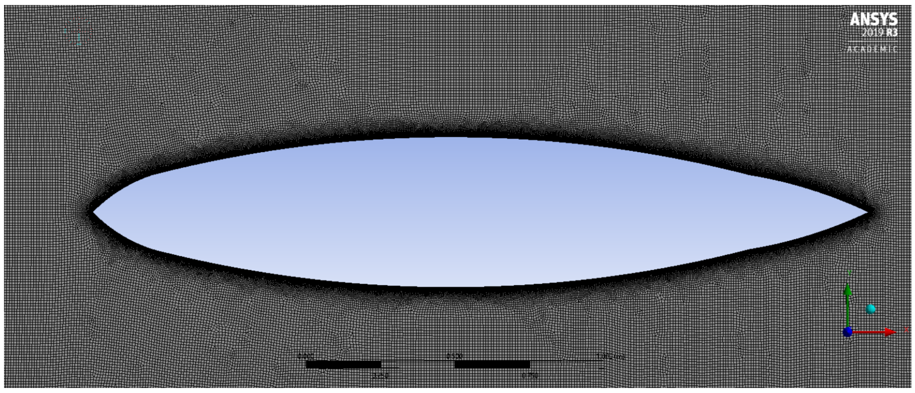


Figure B.12: Final Mesh closeup: Full body

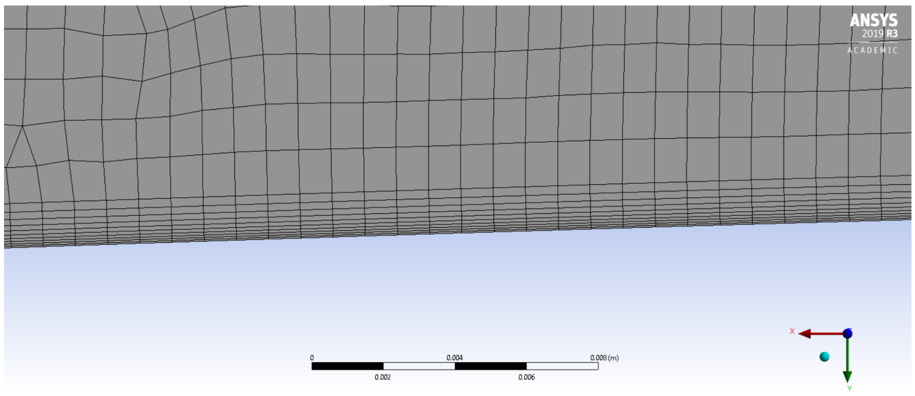


Figure B.13: Final Mesh closeup: Inflation

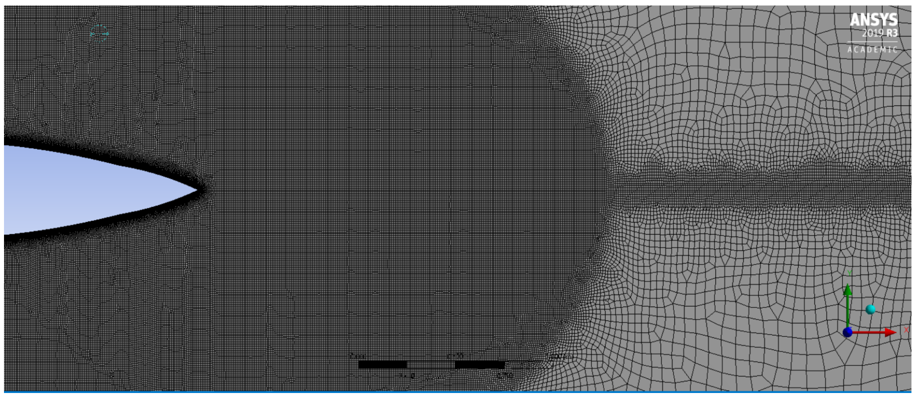
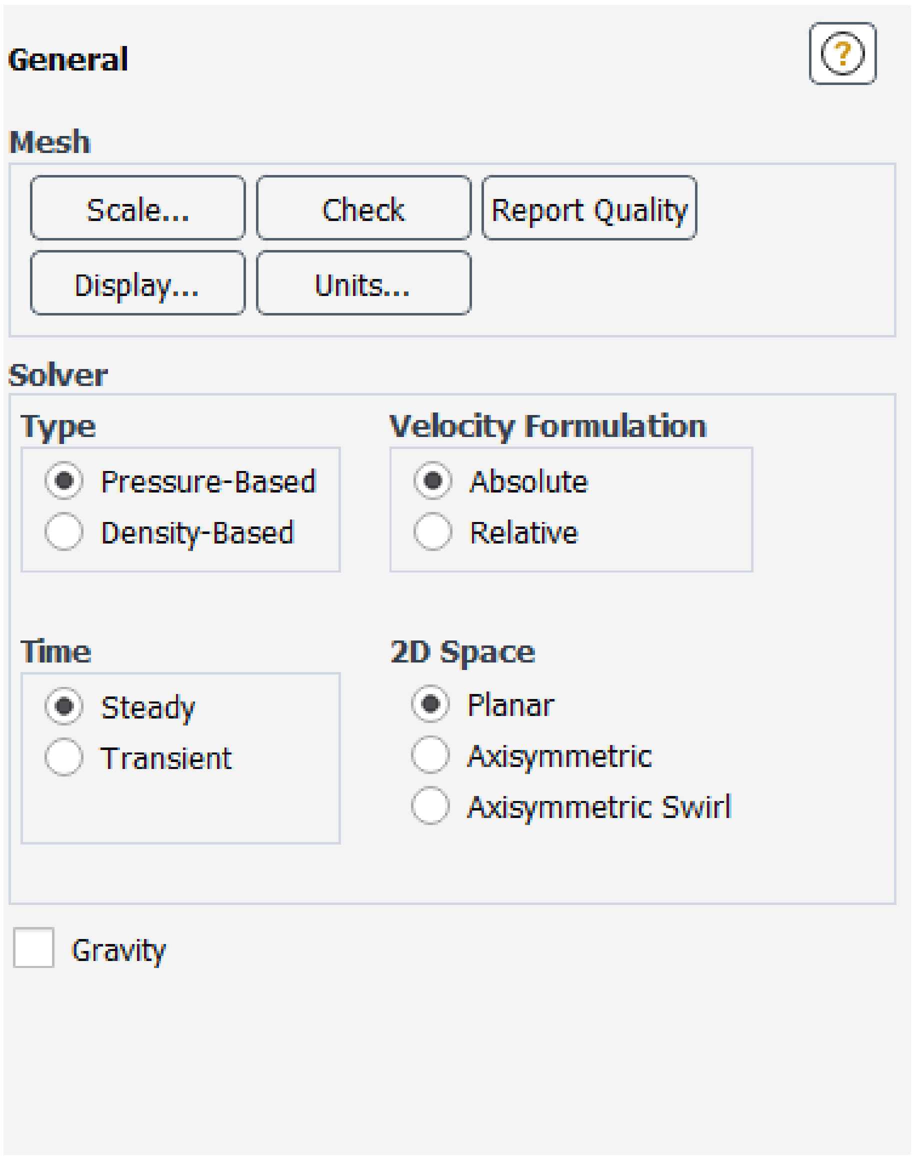


Figure B.14: Final Mesh closeup: Rear part of the body

Appendix C

Ansys Fluent: Step-by-step solution setup


General

Mesh

Solver

Type: [✓ Pressure-Based] Density-Based

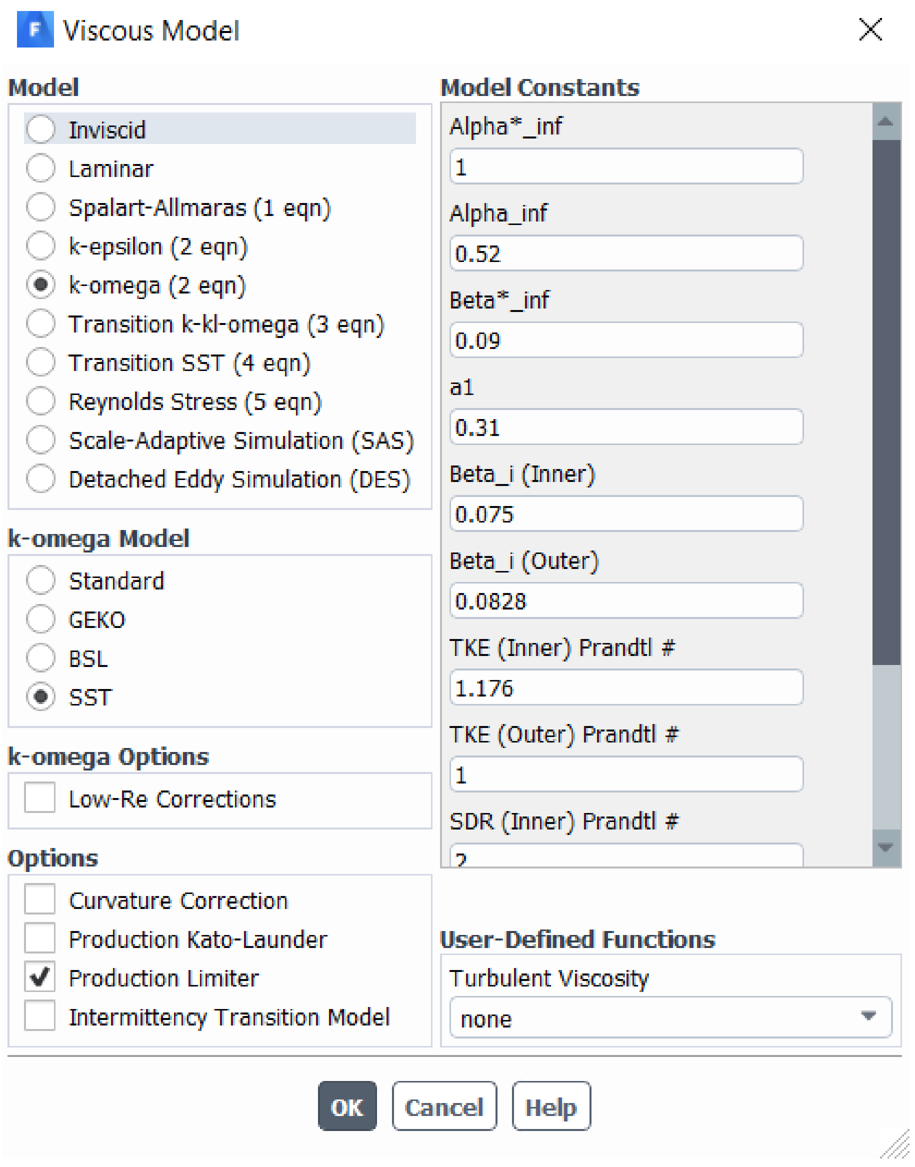
Velocity Formulation: [✓ Absolute] Relative

Time: [✓ Steady] Transient

2D Space: [✓ Planar] Axisymmetric Axisymmetric Swirl

☐ Gravity

Figure C.1: Setup: General



Viscous Model

Model

- ◯ Inviscid
- ◯ Laminar
- ◯ Spalart-Allmaras (1 eqn)
- ◯ k-epsilon (2 eqn)
- ● k-omega (2 eqn)
- ◯ Transition k-kl-omega (3 eqn)
- ◯ Transition SST (4 eqn)
- ◯ Reynolds Stress (5 eqn)
- ◯ Scale-Adaptive Simulation (SAS)
- ◯ Detached Eddy Simulation (DES)

k-omega Model

- ◯ Standard
- ◯ GEKO
- ◯ BSL
- ● SST

k-omega Options

- ☐ Low-Re Corrections

Options

- ☐ Curvature Correction
- ☐ Production Kato-Launder
- ✓ Production Limiter
- ☐ Intermittency Transition Model

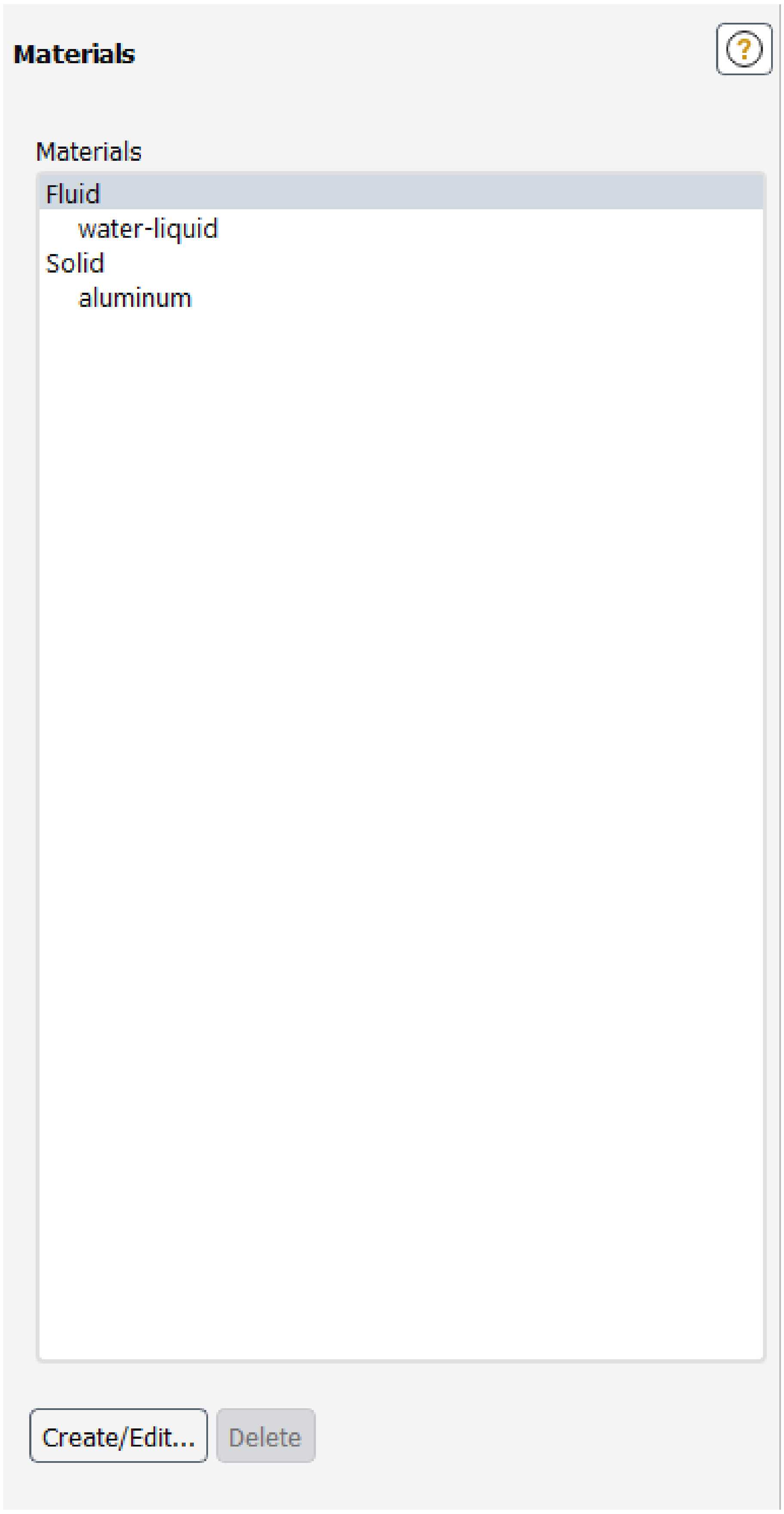
Model Constants

Alpha*_inf	1
Alpha_inf	0.52
Beta*_inf	0.09
a1	0.31
Beta_i (Inner)	0.075
Beta_i (Outer)	0.0828
TKE (Inner) Prandtl #	1.176
TKE (Outer) Prandtl #	1
SDR (Inner) Prandtl #	2

User-Defined Functions

Turbulent Viscosity: none

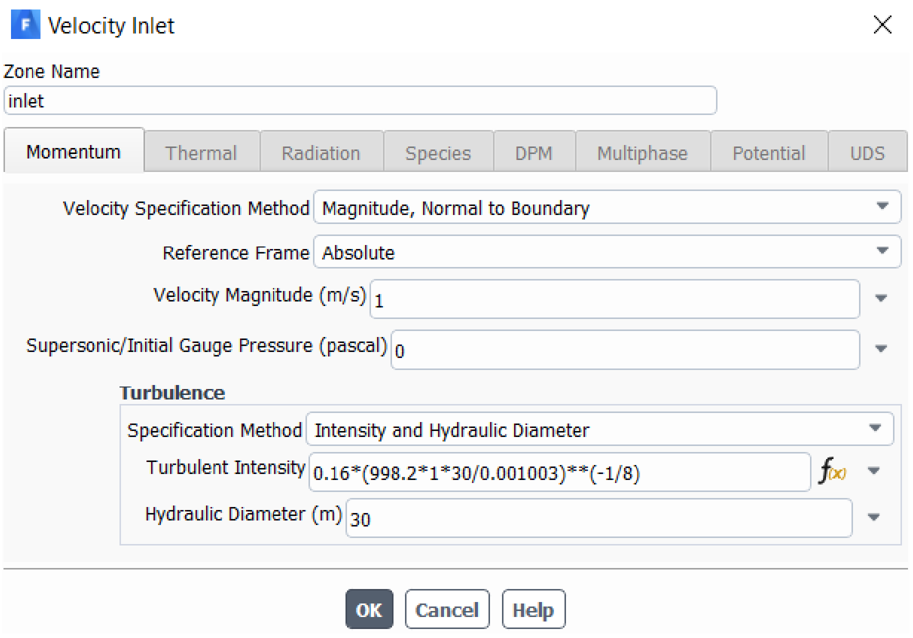
Figure C.2: Setup: Models - Viscous (SST k-omega)


Materials

Materials

Fluid
water-liquid
Solid
aluminum

Figure C.3: Setup: Materials



Zone Name: inlet

Velocity Specification Method: Magnitude, Normal to Boundary

Reference Frame: Absolute

Velocity Magnitude (m/s): 1

Supersonic/Initial Gauge Pressure (pascal): 0

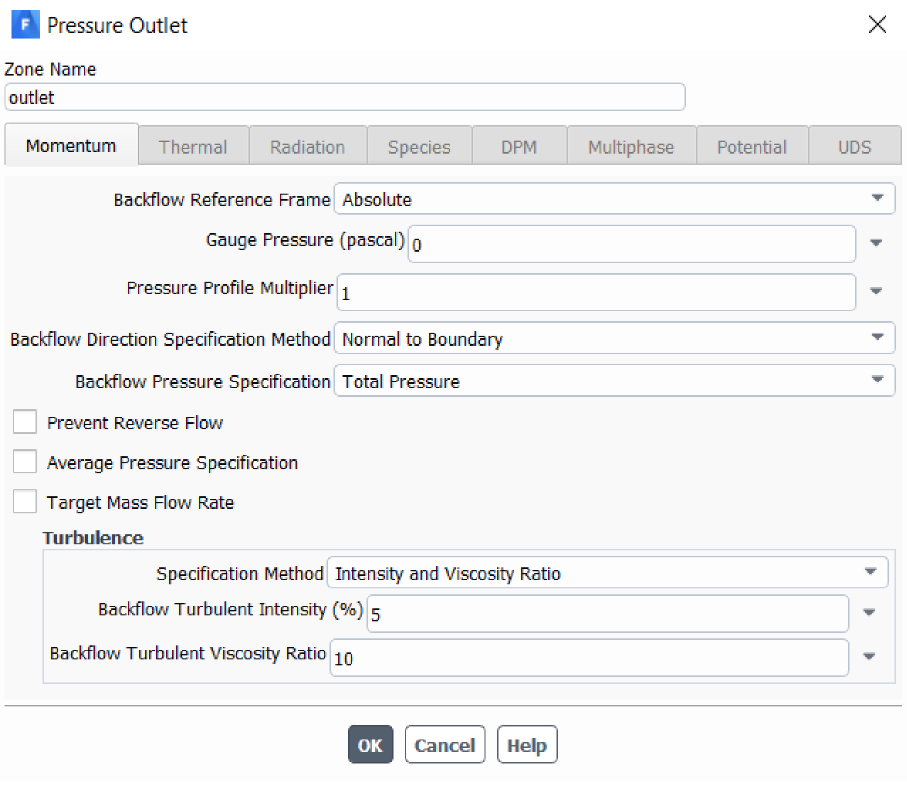
Turbulence

Specification Method: Intensity and Hydraulic Diameter

Turbulent Intensity: 0.16*(998.2*1*30/0.001003)**(-1/8)

Hydraulic Diameter (m): 30

Figure C.4: Setup: Boundary Conditions - Inlet



Zone Name: outlet

Backflow Reference Frame: Absolute

Gauge Pressure (pascal): 0

Pressure Profile Multiplier: 1

Backflow Direction Specification Method: Normal to Boundary

Backflow Pressure Specification: Total Pressure

☐ Prevent Reverse Flow

☐ Average Pressure Specification

☐ Target Mass Flow Rate

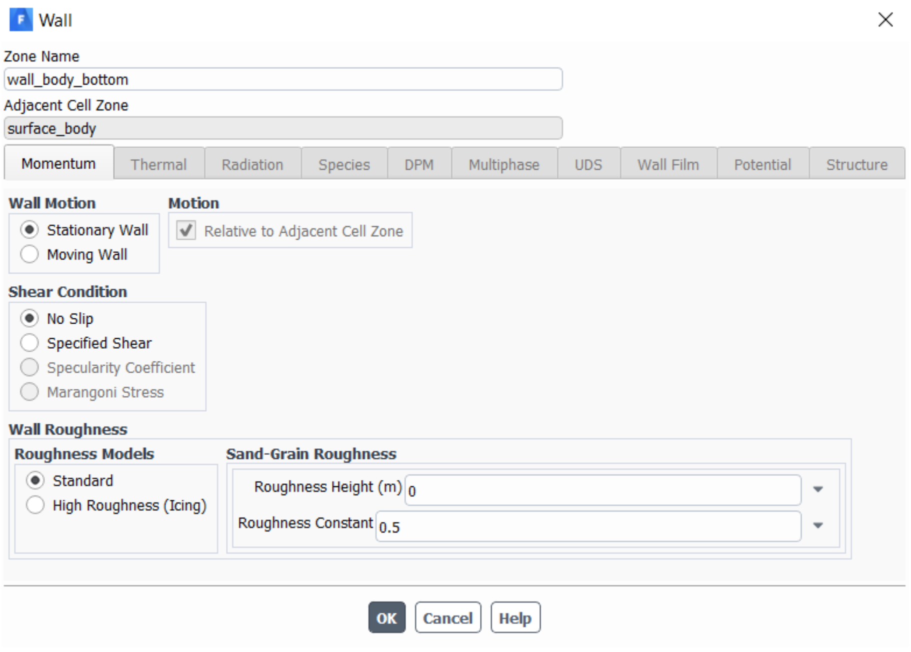
Turbulence

Specification Method: Intensity and Viscosity Ratio

Backflow Turbulent Intensity (%): 5

Backflow Turbulent Viscosity Ratio: 10

Figure C.5: Setup: Boundary Conditions - Outlet


Wall

Zone Name
wall_body_bottom
Adjacent Cell Zone
surface_body

Wall Motion
[● Stationary Wall] Moving Wall

Motion
[✓ Relative to Adjacent Cell Zone]

Shear Condition
[● No Slip] Specified Shear Specularity Coefficient Marangoni Stress

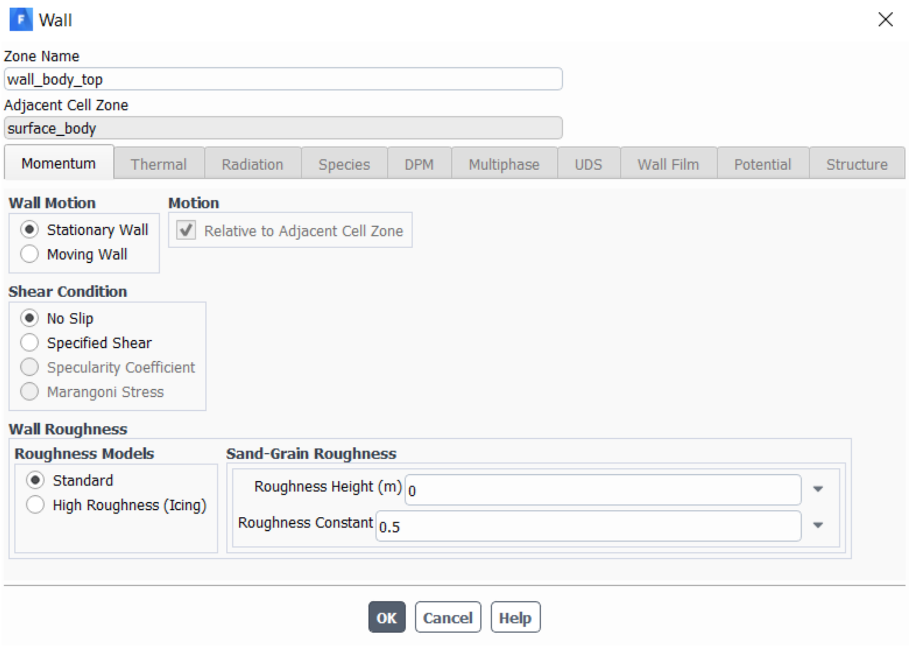
Wall Roughness

Roughness Models
[● Standard] High Roughness (Icing)

Sand-Grain Roughness

Roughness Height (m)	0
Roughness Constant	0.5

Figure C.6: Setup: Boundary Conditions - Wall Body Bottom


Wall

Zone Name
wall_body_top
Adjacent Cell Zone
surface_body

Wall Motion
[● Stationary Wall] Moving Wall

Motion
[✓ Relative to Adjacent Cell Zone]

Shear Condition
[● No Slip] Specified Shear Specularity Coefficient Marangoni Stress

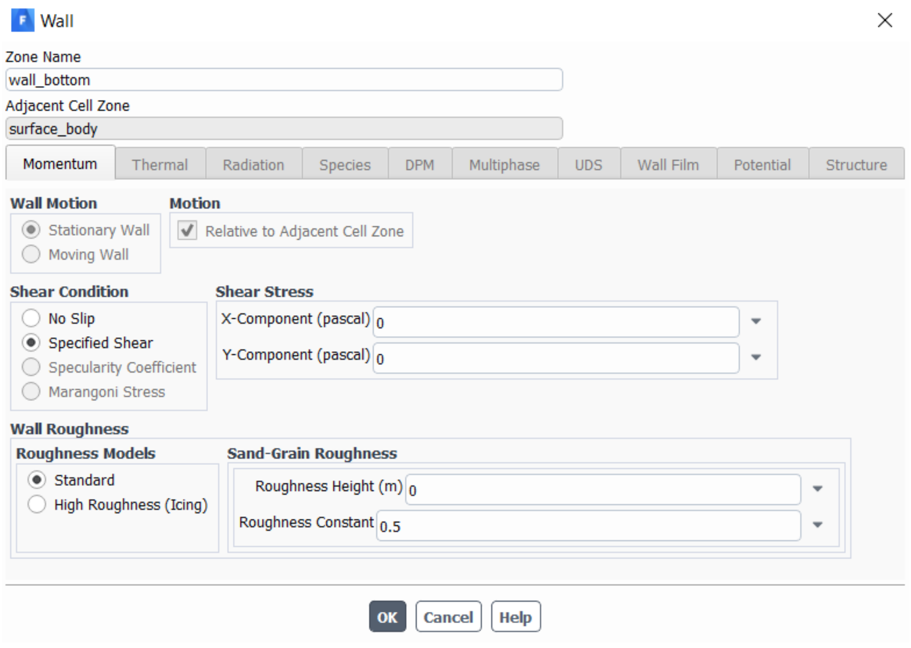
Wall Roughness

Roughness Models
[● Standard] High Roughness (Icing)

Sand-Grain Roughness

Roughness Height (m)	0
Roughness Constant	0.5

Figure C.7: Setup: Boundary Conditions - Wall Body top



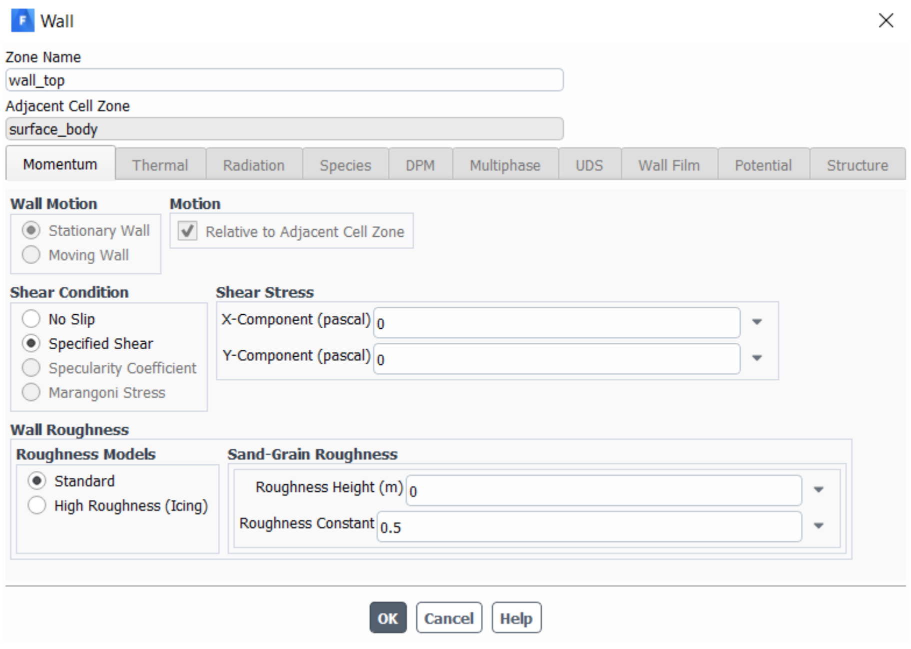
Zone Name: wall_bottom
Adjacent Cell Zone: surface_body

Wall Motion: [● Stationary Wall] ○ Moving Wall
Motion: [✓] Relative to Adjacent Cell Zone

Shear Condition: ○ No Slip [● Specified Shear] ○ Specularity Coefficient ○ Marangoni Stress
Shear Stress
X-Component (pascal) 0
Y-Component (pascal) 0

Wall Roughness
Roughness Models: [● Standard] ○ High Roughness (Icing)
Sand-Grain Roughness
Roughness Height (m) 0
Roughness Constant 0.5

Figure C.8: Setup: Boundary Conditions - Wall Bottom



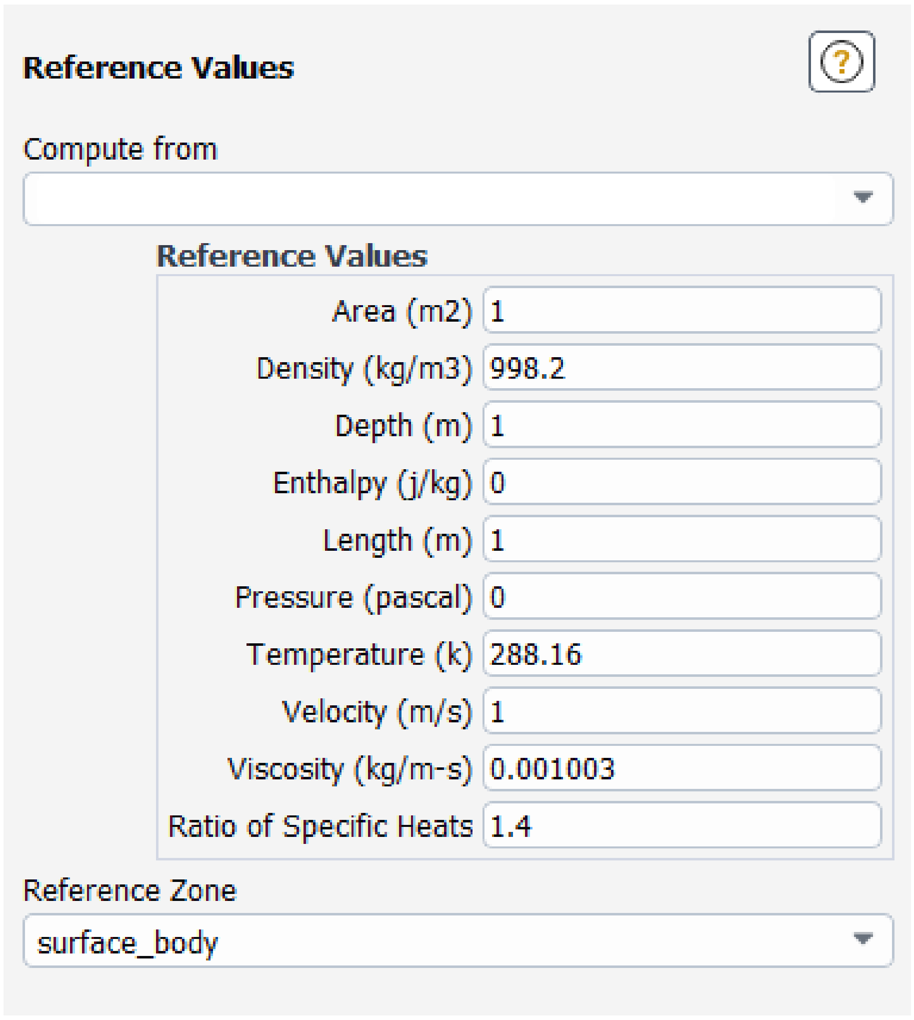
Zone Name: wall_top
Adjacent Cell Zone: surface_body

Wall Motion: [● Stationary Wall] ○ Moving Wall
Motion: [✓] Relative to Adjacent Cell Zone

Shear Condition: ○ No Slip [● Specified Shear] ○ Specularity Coefficient ○ Marangoni Stress
Shear Stress
X-Component (pascal) 0
Y-Component (pascal) 0

Wall Roughness
Roughness Models: [● Standard] ○ High Roughness (Icing)
Sand-Grain Roughness
Roughness Height (m) 0
Roughness Constant 0.5

Figure C.9: Setup: Boundary Conditions - Wall Top



Reference Values

Compute from

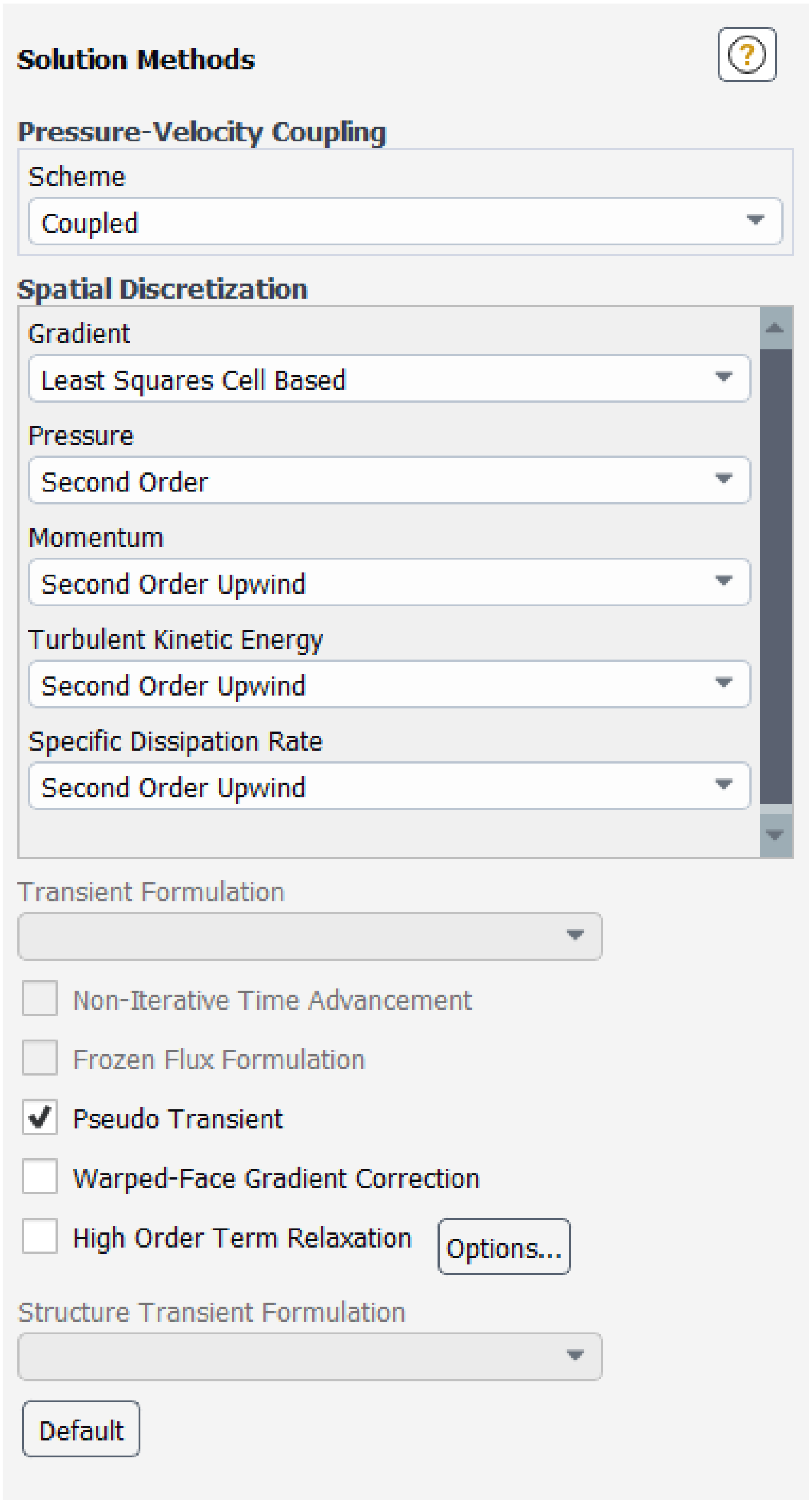
Reference Values

Area (m2)	1
Density (kg/m3)	998.2
Depth (m)	1
Enthalpy (j/kg)	0
Length (m)	1
Pressure (pascal)	0
Temperature (k)	288.16
Velocity (m/s)	1
Viscosity (kg/m-s)	0.001003
Ratio of Specific Heats	1.4

Reference Zone

surface_body

Figure C.10: Setup: Reference Values


Solution Methods

Pressure-Velocity Coupling

Scheme: Coupled

Spatial Discretization

Gradient: Least Squares Cell Based

Pressure: Second Order

Momentum: Second Order Upwind

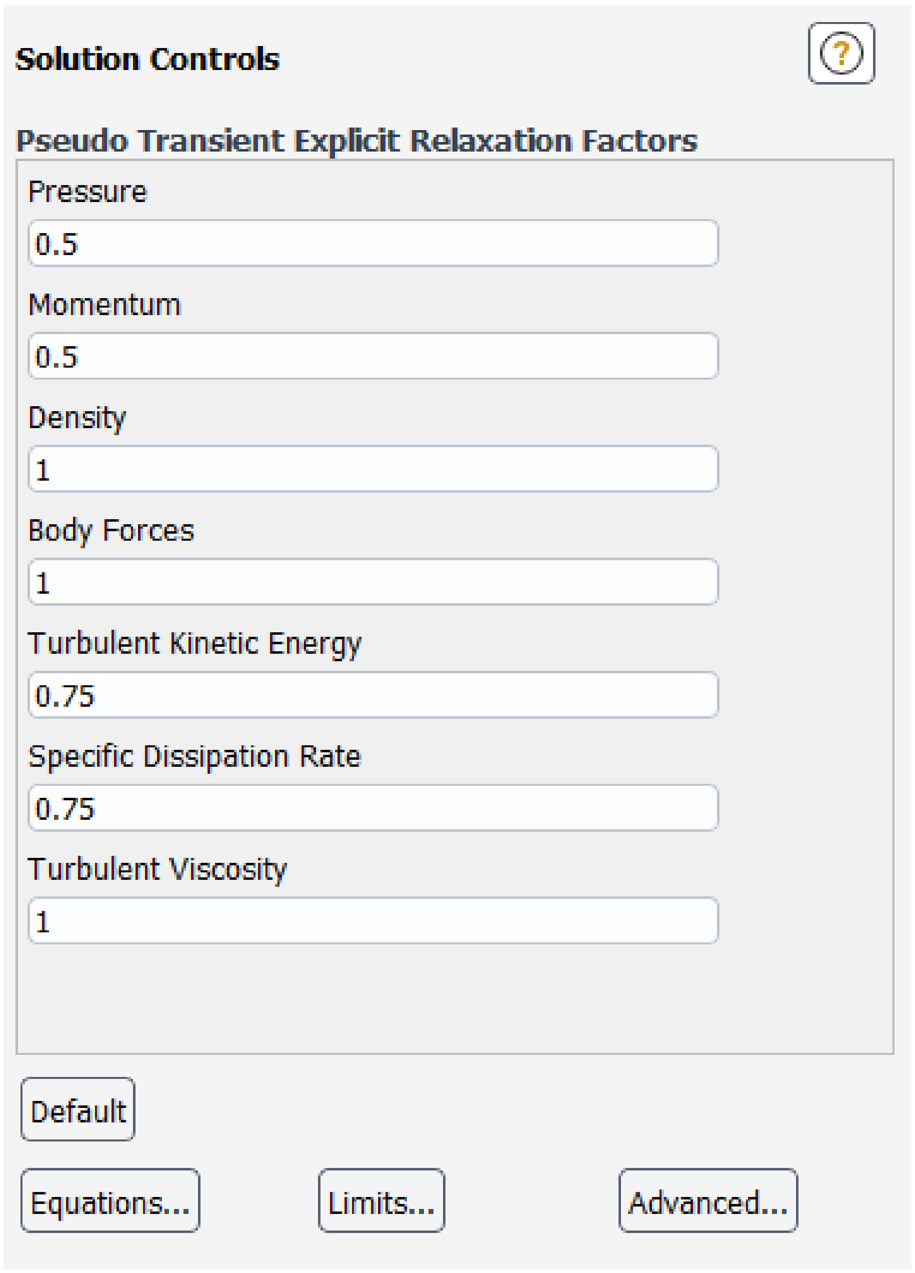
Turbulent Kinetic Energy: Second Order Upwind

Specific Dissipation Rate: Second Order Upwind

Transient Formulation

- [ ] Non-Iterative Time Advancement
- [ ] Frozen Flux Formulation
- [x] Pseudo Transient
- [ ] Warped-Face Gradient Correction
- [ ] High Order Term Relaxation

Figure C.11: Solution: Methods



Solution Controls

Pseudo Transient Explicit Relaxation Factors

Pressure	0.5
Momentum	0.5
Density	1
Body Forces	1
Turbulent Kinetic Energy	0.75
Specific Dissipation Rate	0.75
Turbulent Viscosity	1

Figure C.12: Solution: Controls

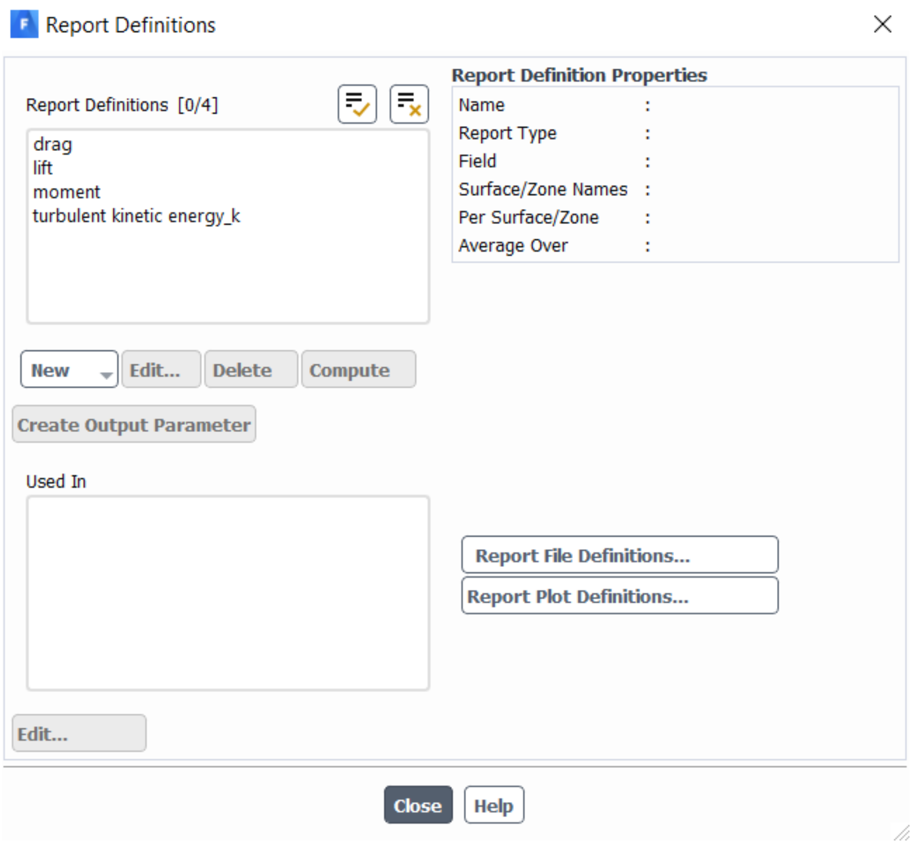


Figure C.13: Solution: Reports Definitions

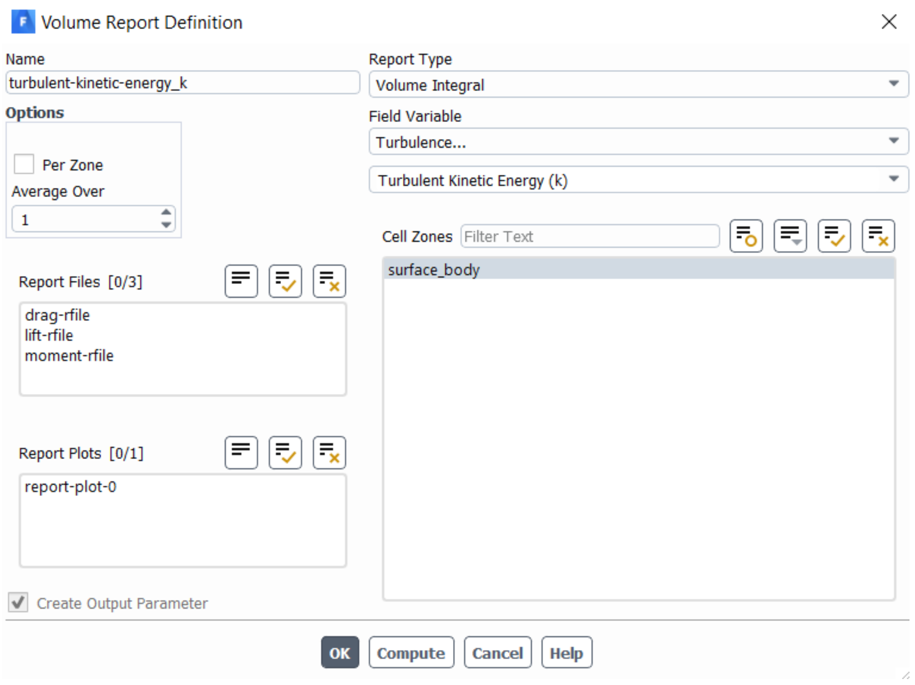
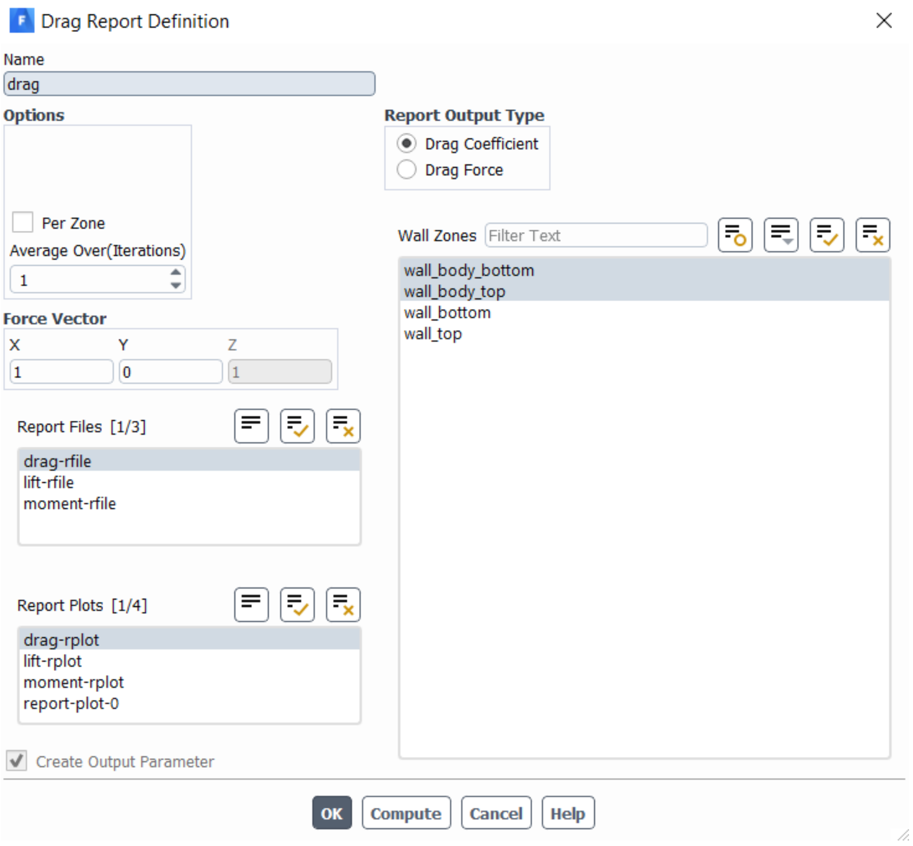


Figure C.14: Solution: Reports Definitions - k


Drag Report Definition

Name: drag

Options
- [ ] Per Zone
- Average Over(Iterations): 1

Report Output Type
- [✓ Drag Coefficient]
- Drag Force

Force Vector

X	Y	Z
1	0	1

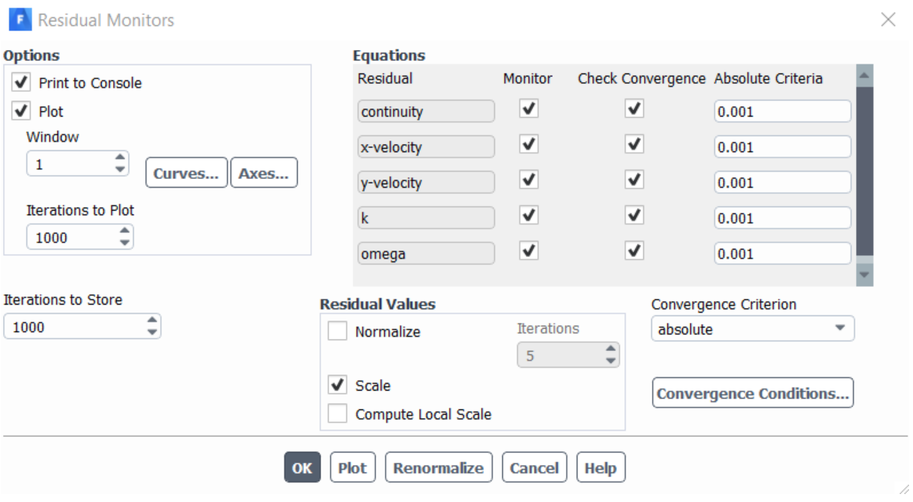
Wall Zones: Filter Text
- wall_body_bottom (selected)
- wall_body_top (selected)
- wall_bottom
- wall_top

Report Files [1/3]
- drag-rfile (selected)
- lift-rfile
- moment-rfile

Report Plots [1/4]
- drag-rplot (selected)
- lift-rplot
- moment-rplot
- report-plot-0

- [x] Create Output Parameter

Figure C.15: Solution: Reports Definitions - C_D


Residual Monitors

Options
- [x] Print to Console
- [x] Plot
- Window: 1
- Iterations to Plot: 1000

Iterations to Store: 1000

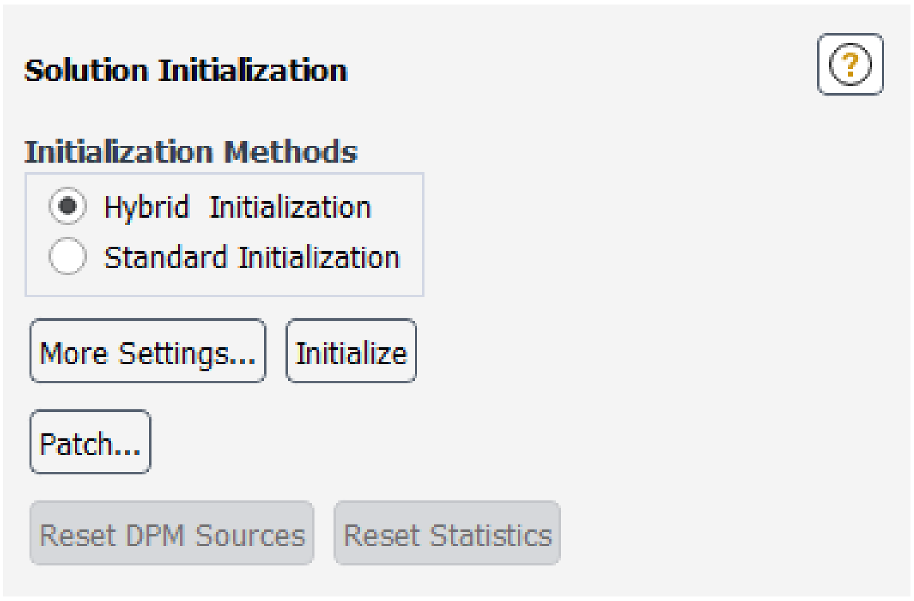
Equations

Residual	Monitor	Check Convergence	Absolute Criteria
continuity	✓	✓	0.001
x-velocity	✓	✓	0.001
y-velocity	✓	✓	0.001
k	✓	✓	0.001
omega	✓	✓	0.001

Residual Values
- [ ] Normalize — Iterations: 5
- [x] Scale
- [ ] Compute Local Scale

Convergence Criterion: absolute

Figure C.16: Solution: Monitors - Residual

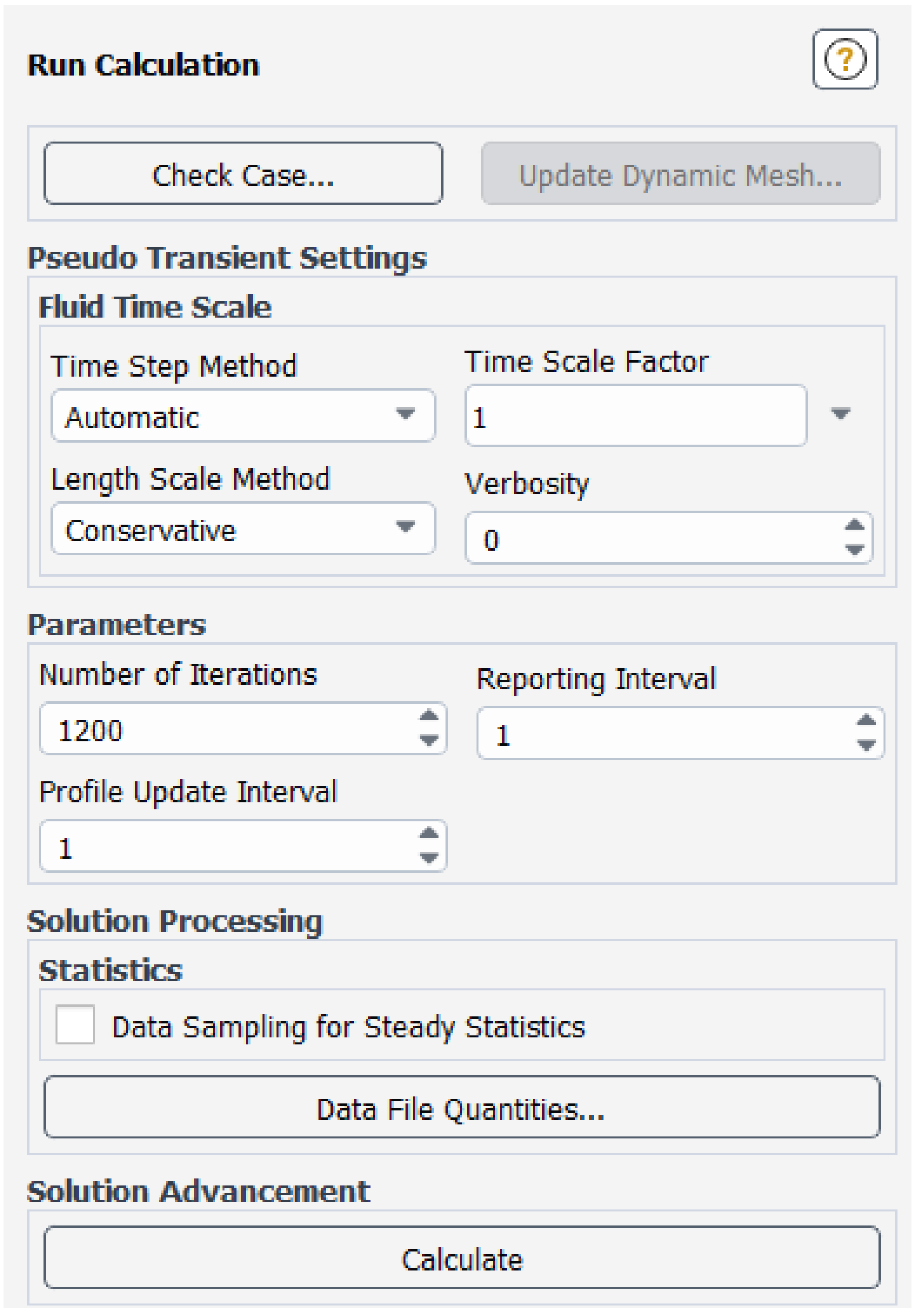


Solution Initialization

Initialization Methods

- [✓ Hybrid Initialization]
- Standard Initialization

Figure C.17: Solution: Initialization



Run Calculation

Check Case...

Update Dynamic Mesh...

Pseudo Transient Settings

Fluid Time Scale

Time Step Method: Automatic

Time Scale Factor: 1

Length Scale Method: Conservative

Verbosity: 0

Parameters

Number of Iterations: 1200

Reporting Interval: 1

Profile Update Interval: 1

Solution Processing

Statistics

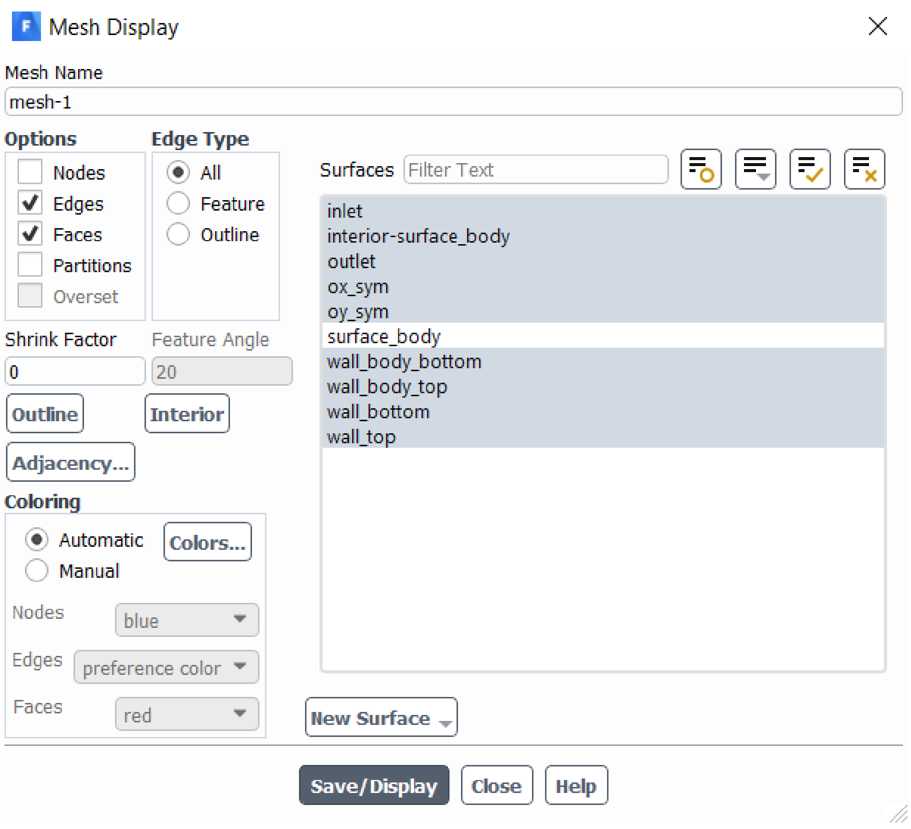
☐ Data Sampling for Steady Statistics

Data File Quantities...

Solution Advancement

Calculate

Figure C.18: Solution: Run Calculation


Mesh Display

Mesh Name: mesh-1

Options
- [ ] Nodes
- [✓] Edges
- [✓] Faces
- [ ] Partitions
- [ ] Overset

Edge Type: [✓ All] Feature Outline

Shrink Factor: 0

Feature Angle: 20

Coloring: [✓ Automatic] Manual

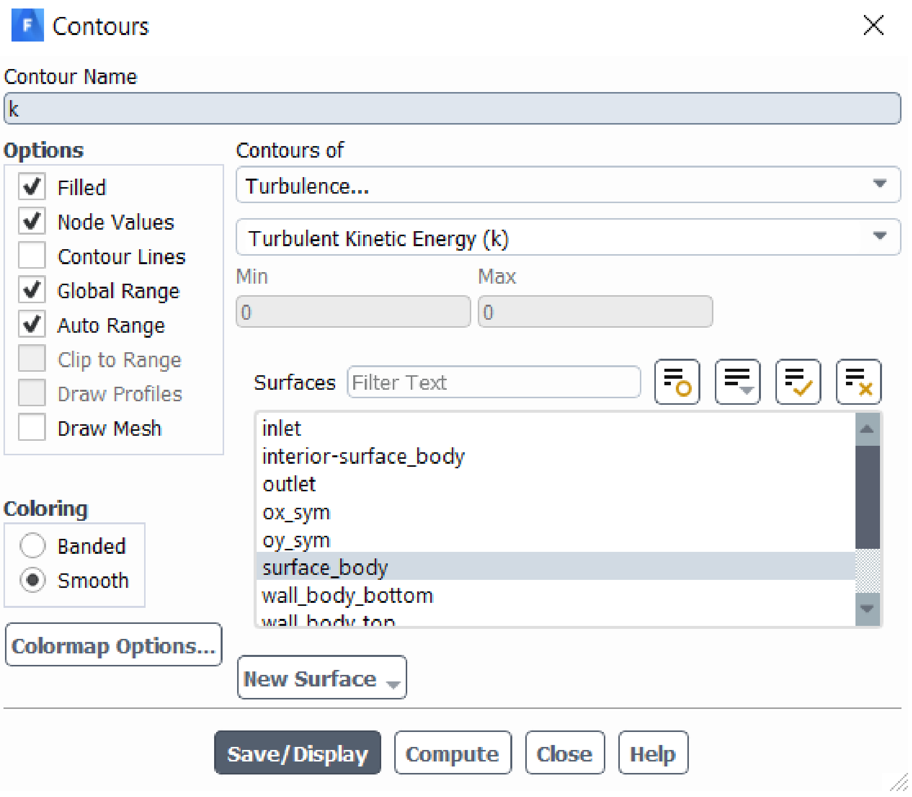
Nodes: blue

Edges: preference color

Faces: red

Surfaces:
inlet
interior-surface_body
outlet
ox_sym
oy_sym
surface_body
wall_body_bottom
wall_body_top
wall_bottom
wall_top

Figure C.19: Results: Mesh Display


Contours

Contour Name: k

Options
- [✓] Filled
- [✓] Node Values
- [ ] Contour Lines
- [✓] Global Range
- [✓] Auto Range
- [ ] Clip to Range
- [ ] Draw Profiles
- [ ] Draw Mesh

Contours of: Turbulence...

Turbulent Kinetic Energy (k)

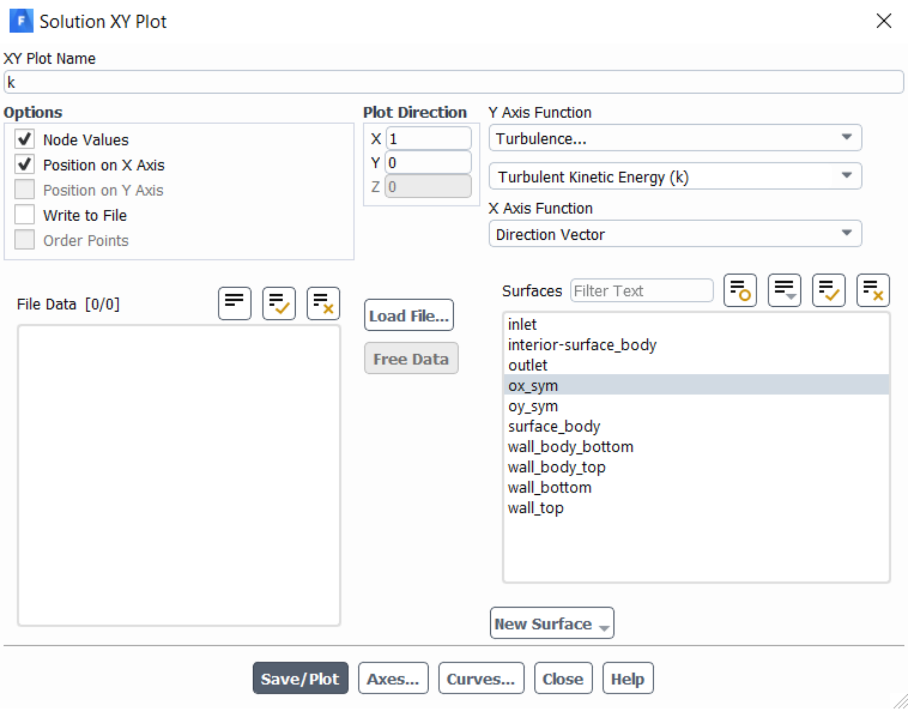
Min: 0 Max: 0

Surfaces: Filter Text

inlet
interior-surface_body
outlet
ox_sym
oy_sym
[✓ surface_body]
wall_body_bottom
wall_body_top

Coloring
- Banded
- [✓ Smooth]

Figure C.20: Results: Contours - k


Solution XY Plot

XY Plot Name: k

Options
- [✓] Node Values
- [✓] Position on X Axis
- [ ] Position on Y Axis
- [ ] Write to File
- [ ] Order Points

Plot Direction
X: 1
Y: 0
Z: 0

Y Axis Function: Turbulence...

Turbulent Kinetic Energy (k)

X Axis Function: Direction Vector

File Data [0/0]

Surfaces: Filter Text

inlet
interior-surface_body
outlet
[✓ ox_sym]
oy_sym
surface_body
wall_body_bottom
wall_body_top
wall_bottom
wall_top

Figure C.21: Results: Plots - XY Plot

Appendix D

Uniform Design Method (UDM)

Table D.1: Study case variables domain divided in equal partitions

27 domain points in sequence							
Design points	R_f	L_f	D	L	R_r	L_r	R_m
1	0.450000	0.280000	0.200000	0.800000	0.450000	0.320000	7.200000
2	0.458654	0.285385	0.203846	0.815385	0.458654	0.326154	7.338462
3	0.467308	0.290769	0.207692	0.830769	0.467308	0.332308	7.476923
4	0.475962	0.296154	0.211538	0.846154	0.475962	0.338462	7.615385
5	0.484615	0.301538	0.215385	0.861538	0.484615	0.344615	7.753846
6	0.493269	0.306923	0.219231	0.876923	0.493269	0.350769	7.892308
7	0.501923	0.312308	0.223077	0.892308	0.501923	0.356923	8.030769
8	0.510577	0.317692	0.226923	0.907692	0.510577	0.363077	8.169231
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692
10	0.527885	0.328462	0.234615	0.938462	0.527885	0.375385	8.446154
11	0.536538	0.333846	0.238462	0.953846	0.536538	0.381538	8.584615
12	0.545192	0.339231	0.242308	0.969231	0.545192	0.387692	8.723077
13	0.553846	0.344615	0.246154	0.984615	0.553846	0.393846	8.861538
14	0.562500	0.350000	0.250000	1.000000	0.562500	0.400000	9.000000
15	0.571154	0.355385	0.253846	1.015385	0.571154	0.406154	9.138462
16	0.579808	0.360769	0.257692	1.030769	0.579808	0.412308	9.276923
17	0.588462	0.366154	0.261538	1.046154	0.588462	0.418462	9.415385
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846
19	0.605769	0.376923	0.269231	1.076923	0.605769	0.430769	9.692308
20	0.614423	0.382308	0.273077	1.092308	0.614423	0.436923	9.830769
21	0.623077	0.387692	0.276923	1.107692	0.623077	0.443077	9.969231
22	0.631731	0.393077	0.280769	1.123077	0.631731	0.449231	10.107692
23	0.640385	0.398462	0.284615	1.138462	0.640385	0.455385	10.246154
24	0.649038	0.403846	0.288462	1.153846	0.649038	0.461538	10.384615
25	0.657692	0.409231	0.292308	1.169231	0.657692	0.467692	10.523077
26	0.666346	0.414615	0.296154	1.184615	0.666346	0.473846	10.661538
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000

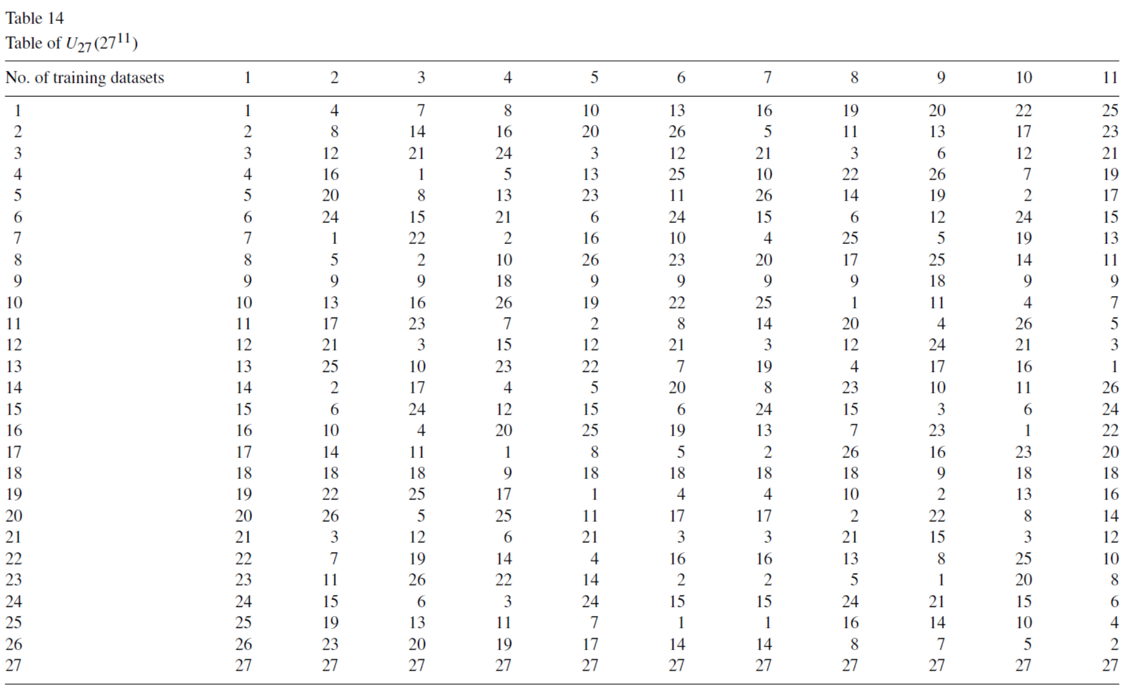


Table 14
Table of $U_{27}(27^{11})$

No. of training datasets	1	2	3	4	5	6	7	8	9	10	11
1	1	4	7	8	10	13	16	19	20	22	25
2	2	8	14	16	20	26	5	11	13	17	23
3	3	12	21	24	3	12	21	3	6	12	21
4	4	16	1	5	13	25	10	22	26	7	19
5	5	20	8	13	23	11	26	14	19	2	17
6	6	24	15	21	6	24	15	6	12	24	15
7	7	1	22	2	16	10	4	25	5	19	13
8	8	5	2	10	26	23	20	17	25	14	11
9	9	9	9	18	9	9	9	9	18	9	9
10	10	13	16	26	19	22	25	1	11	4	7
11	11	17	23	7	2	8	14	20	4	26	5
12	12	21	3	15	12	21	3	12	24	21	3
13	13	25	10	23	22	7	19	4	17	16	1
14	14	2	17	4	5	20	8	23	10	11	26
15	15	6	24	12	15	6	24	15	3	6	24
16	16	10	4	20	25	19	13	7	23	1	22
17	17	14	11	1	8	5	2	26	16	23	20
18	18	18	18	9	18	18	18	18	9	18	18
19	19	22	25	17	1	4	4	10	2	13	16
20	20	26	5	25	11	17	17	2	22	8	14
21	21	3	12	6	21	3	3	21	15	3	12
22	22	7	19	14	4	16	16	13	8	25	10
23	23	11	26	22	14	2	2	5	1	20	8
24	24	15	6	3	24	15	15	24	21	15	6
25	25	19	13	11	7	1	1	16	14	10	4
26	26	23	20	19	17	14	14	8	7	5	2
27	27	27	27	27	27	27	27	27	27	27	27

Figure D.1: Uniform Design Method table

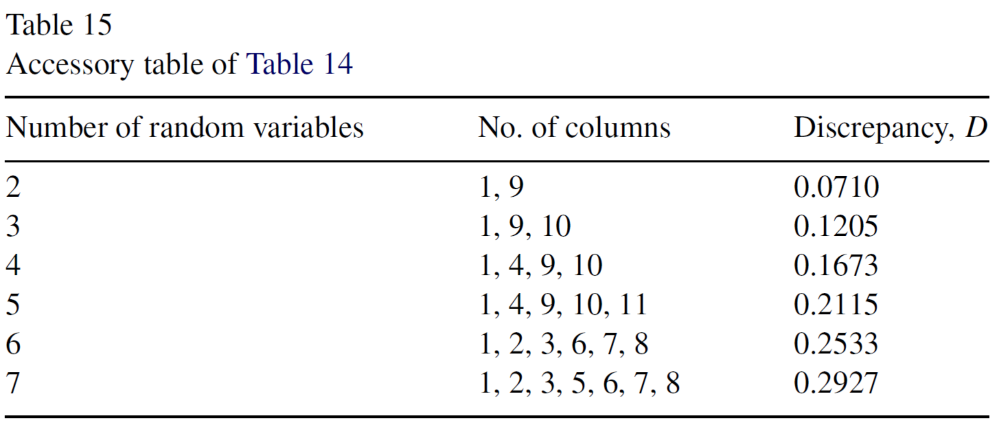


Table 15
Accessory table of Table 14

Number of random variables	No. of columns	Discrepancy, D
2	1, 9	0.0710
3	1, 9, 10	0.1205
4	1, 4, 9, 10	0.1673
5	1, 4, 9, 10, 11	0.2115
6	1, 2, 3, 6, 7, 8	0.2533
7	1, 2, 3, 5, 6, 7, 8	0.2927

Figure D.2: Uniform Design Method accessory table

Table D.2: Uniform Design Method points for the problem of seven variables and respective discrepancy

UDM design points for discrepancy $D_p = 0.2927$							
Design points	**(1)**	**(2)**	**(3)**	**(5)**	**(6)**	**(7)**	**(8)**
1	1	4	7	10	13	16	19
2	2	8	14	20	26	5	11
3	3	12	21	3	12	21	3
4	4	16	1	13	25	10	22
5	5	20	8	23	11	26	14
6	6	24	15	6	24	15	6
7	7	1	22	16	10	4	25
8	8	5	2	26	23	20	17
9	9	9	9	9	9	9	9
10	10	13	16	19	22	25	1
11	11	17	23	2	8	14	20
12	12	21	3	12	21	3	12
13	13	25	10	22	7	19	4
14	14	2	17	5	20	8	23
15	15	6	24	15	6	24	15
16	16	10	4	25	19	13	7
17	17	14	11	8	5	2	26
18	18	18	18	18	18	18	18
19	19	22	25	1	4	4	10
20	20	26	5	11	17	17	2
21	21	3	12	21	3	3	21
22	22	7	19	4	16	16	13
23	23	11	26	14	2	2	5
24	24	15	6	24	15	15	24
25	25	19	13	7	1	1	16
26	26	23	20	17	14	14	8
27	27	27	27	27	27	27	27

Table D.3: Input variable values introduced into software Ansys Fluent: after recombination

Input variables used in Ansys Fluent							
Design points	R_f **(1)**	L_f **(2)**	D **(3)**	L **(5)**	R_r **(6)**	L_r **(7)**	R_m **(8)**
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000

Appendix E

MATLAB Code

E.1 Matlab Code - Artificial Neural Network construction and Analysis of Sensitivity (Sobol)

```
clear all



N=27;


R_f = xlsread('UDM-ANN_runs.xlsx', 'ANSYS_10', 'F6:F32');
L_f = xlsread('UDM-ANN_runs.xlsx', 'ANSYS_10', 'G6:G32');
D = xlsread('UDM-ANN_runs.xlsx', 'ANSYS_10', 'H6:H32');
L = xlsread('UDM-ANN_runs.xlsx', 'ANSYS_10', 'I6:I32');
R_r = xlsread('UDM-ANN_runs.xlsx', 'ANSYS_10', 'J6:J32');
L_r = xlsread('UDM-ANN_runs.xlsx', 'ANSYS_10', 'K6:K32');
R_m = xlsread('UDM-ANN_runs.xlsx', 'ANSYS_10', 'L6:L32');


Var_In=transpose([R_f,L_f,D,L,R_r,L_r,R_m]);
[Store_Ent,P_In] = mapminmax(Var_In);
Uniform_In = mapminmax('apply',Var_In,P_In);


C_D = xlsread('UDM-ANN_runs.xlsx', 'ANSYS_10', 'M6:M32');
k = xlsread('UDM-ANN_runs.xlsx', 'ANSYS_10', 'N6:N32');


Var_Out = transpose([C_D,k]);
[Store_Out,P_Out] = mapminmax(Var_Out);
Uniform_Out = mapminmax('apply',Var_Out,P_Out);
```

```
net = feedforwardnet(6);
net = configure(net,Uniform_In,Uniform_Out);
view(net);
[net,tr] = train(net,Uniform_In,Uniform_Out);
net_nova10=net; save net_nova10;

Verif_ent=mapminmax('reverse',Uniform_In,P_In);
Fluent_val = mapminmax('reverse',Uniform_Out,P_Out);
Network_val_uniform = sim(net,Uniform_In);
NET_val = mapminmax('reverse',Network_val_uniform,P_Out);

Ninput=100;
for i=1:10
Var_rand= -1+2*rand(7,Ninput);
end

Var_rand= -1+2*rand(7,Ninput);
Network_rand = sim(net,Var_rand);
penalF1=(sum(sum(Network_rand(1,:)<-1))+
sum(sum(Network_rand(1,:)>+1)))/Ninput;
penalF2=(sum(sum(Network_rand(2,:)<-1))+
sum(sum(Network_rand(2,:)>+1)))/Ninput;
fprintf('penalizacao');disp(penalF1);disp(penalF2);

Ent_rand=mapminmax('reverse',Var_rand,P_In);
NET_rand = mapminmax('reverse',Network_rand,P_Out);

figure(1);
subplot(2,3,1); histogram(Uniform_In);title('7 IN');
subplot(2,3,2);histogram(Uniform_Out(1,:));title('7 C D');
subplot(2,3,3); histogram(Uniform_Out(2,:));title('7 k');

subplot(2,3,4); histogram(Var_rand);title('100 rand IN');
subplot(2,3,5);histogram(Network_rand(1,:));title('100 net C D');
subplot(2,3,6); histogram(Network_rand(2,:));title('100 net k');

figure(2);
subplot(2,3,1); histogram(Verif_ent);title('7 IN');
subplot(2,3,2);histogram(NET_val(1,:));title('7 C D');
subplot(2,3,3);histogram(NET_val(2,:));title('7 k');

subplot(2,3,4); histogram(Ent_rand);title('100 rand IN');
subplot(2,3,5);histogram(NET_rand(1,:));title('100 net C D');
subplot(2,3,6);histogram(NET_rand(2,:));title('100 net k');


Ninput=100;
```

```
for i=1:10
Var_rand= -1+2*rand(7,Ninput);
end

Network_rand = sim(net,Var_rand);
NET_val = mapminmax('reverse',Network_rand,P_Out);


Nfixo=50;
for i=1:10
Var_fixo=-1+2*rand(1,Nfixo);
end

dim_MAXIMA=7*Nfixo*Ninput;
A_total1=zeros(1,dim_MAXIMA);
A_total2=zeros(1,dim_MAXIMA);
 mediaNfixo_1=zeros(1,7);
 varianciaNfixo_1= zeros(1,7);
 mediaNfixo_2= zeros(1,7);
 varianciaNfixo_2= zeros(1,7);
cc1=1;
cc2=Ninput;
for i=1:1:7
    disp(i)
    A_7_Out1=zeros(1,Nfixo);
    A_7_Out2=zeros(1,Nfixo);
        for kk=1:1:Nfixo
        New_7_rand=Var_rand;
        P=Var_fixo(1,kk)*ones(1,Ninput);
        New_7_rand(i,:)=P;


         Out_NET = sim(net,New_7_rand);
         NEW_2_Out = mapminmax('reverse',Out_NET,P_Out);


        A_total1(cc1:cc2)=NEW_2_Out(1,:);
        A_total2(cc1:cc2)=NEW_2_Out(2,:);
        cc1=cc1+Ninput;
        cc2=cc2+Ninput;


         Media_NEW_2_Out= mean(NEW_2_Out.');
         A_7_Out1(kk)=Media_NEW_2_Out(1);
         A_7_Out2(kk)=Media_NEW_2_Out(2);
        end
    mediaNfixo_1(i)= mean(A_7_Out1);
    varianciaNfixo_1(i)= var(A_7_Out1);
    mediaNfixo_2(i)= mean( A_7_Out2);
    varianciaNfixo_2(i)= var( A_7_Out2);

end

fprintf('MEDIA_da_resposta do sistema')
MEDIA_do_sistema1=mean(A_total1);
MEDIA_do_sistema2=mean(A_total2);


VAR_do_sistema1=var(A_total1);
VAR_do_sistema2=var(A_total2);
 Global_s1=zeros(1,7);
 Global_s2=zeros(1,7);

for i=1:1:7
     Global_s1(1,i)= varianciaNfixo_1(i)/VAR_do_sistema1;
     Global_s2(1,i)= varianciaNfixo_2(i)/VAR_do_sistema2;
end
fprintf('mediaNfixo')
disp(mediaNfixo_1)
disp(mediaNfixo_2)
fprintf('Global_s')
disp(Global_s1)
disp(Global_s2)
fprintf('Soma Global_s')

disp(sum(Global_s1))
disp(sum(Global_s2))
```

E.2 Matlab Code - Optimization

```
clear
close
format compact
warning off


nDes = 7;


nG = 250;
nPi= 10;
nP =  8;
nSC = 4;
```

```
nAC = 0;
nDC = 0;
nI  = 8;

load('net_final9.mat')
NITER=1;
for iter=1 :NITER
[REVERSE,Xbest, Fbest, Fbest1, Fbest2] = Genetic_Ricardo(nDes, nG,
nPi, nP, nSC, nAC, nDC, nI);
resul(:,iter)=REVERSE

[n m] = size(Xbest);

R_final(iter,:)=[Xbest(n,:),Fbest(n),Fbest1(n),Fbest2(n)];
disp(iter);
end
 xlswrite('res.xlsx', resul','V9','A2:I2')
 xlswrite('res.xlsx', Fbest','V9','A4:A253')
  xlswrite('res.xlsx', Fbest1','V9','B4:B253')
   xlswrite('res.xlsx', Fbest2','V9','C4:C253')

Xbest
Fbest
figure(2)
  plot(Fbest1,'--ks','LineWidth',2,...
                 'MarkerEdgeColor','k',...
                'MarkerFaceColor','b',...
                 'MarkerSize',5);
xlabel('n de geracoes')
ylabel('Funcao objectivo C D')
title('Grafico de  evolucao')
figure(3)
  plot(Fbest2,'--ks','LineWidth',2,...
                 'MarkerEdgeColor','k',...
                'MarkerFaceColor','r',...
                 'MarkerSize',5);
xlabel('n de geracoes')
ylabel('Funcao objectivo k')
title('Grafico de  evolucao')
 [n m] = size(R_final);
 R_final
 REVERSE
```

Appendix F

Ansys Fluent: Output Tables

Table F.1: Ansys Fluent results for the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) for Flow Velocity (U_{in}) equal to 1 m/s

Input experimental values used in ANN learning process									
Design points	L	D	L_f	L_r	R_f	R_r	R_m	C_D	k
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308	0.033144	0.305473
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615	0.031071	0.305108
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923	0.039649	0.310327
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692	0.025899	0.301980
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000	0.037812	0.310093
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308	0.031897	0.304188
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077	0.039754	0.311056
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385	0.028202	0.302645
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692	0.031545	0.305447
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000	0.034409	0.307167
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769	0.040431	0.311515
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077	0.025976	0.301877
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385	0.038530	0.308811
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154	0.031987	0.305182
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462	0.047291	0.318555
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769	0.028328	0.303449
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538	0.032874	0.306336
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846	0.034451	0.307676
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154	0.042198	0.310531
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462	0.030320	0.304799
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231	0.036310	0.309592
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538	0.036521	0.308418
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846	0.043124	0.311773
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615	0.031577	0.304466
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923	0.035285	0.307400
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231	0.036925	0.310261
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000	0.037794	0.309361

Table F.2: Ansys Fluent results for the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) for Flow Velocity (U_{in}) equal to 2 m/s

Input experimental values used in ANN learning process									
Design points	L	D	L_f	L_r	R_f	R_r	R_m	C_D	k
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308	0.107094	1.028056
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615	0.108687	1.034550
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923	0.138756	1.044894
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692	0.091444	1.019145
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000	0.132279	1.043819
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308	0.109181	1.033012
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077	0.132328	1.044656
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385	0.100345	1.023369
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692	0.111672	1.029018
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000	0.117168	1.033338
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769	0.141607	1.051201
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077	0.092429	1.020020
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385	0.131030	1.043396
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154	0.112150	1.030565
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462	0.167922	1.073896
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769	0.098947	1.025657
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538	0.114288	1.035269
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846	0.125533	1.042056
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154	0.146241	1.054646
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462	0.101657	1.024083
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231	0.123311	1.041290
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538	0.120572	1.035821
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846	0.153235	1.060358
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615	0.109218	1.032829
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923	0.126196	1.038929
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231	0.136220	1.043982
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000	0.138222	1.049559

Table F.3: Ansys Fluent results for the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) for Flow Velocity (U_{in}) equal to 3 m/s

Input experimental values used in ANN learning process									
Design points	L	D	L_f	L_r	R_f	R_r	R_m	C_D	k
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308	0.225504	2.094556
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615	0.223466	2.098569
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923	0.284190	2.130627
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692	0.187562	2.076563
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000	0.276007	2.120333
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308	0.226477	2.096817
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077	0.276817	2.124674
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385	0.209473	2.088599
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692	0.230946	2.098135
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000	0.251552	2.113044
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769	0.292556	2.134694
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077	0.192234	2.079668
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385	0.275823	2.123247
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154	0.231051	2.098951
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462	0.341929	2.161093
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769	0.211961	2.090762
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538	0.240075	2.102270
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846	0.256837	2.114912
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154	0.303540	2.138652
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462	0.215439	2.091258
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231	0.257627	2.113756
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538	0.251163	2.112309
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846	0.315219	2.150319
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615	0.227105	2.097237
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923	0.260378	2.113362
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231	0.270176	2.123096
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000	0.283478	2.128719

Table F.4: Ansys Fluent results for the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) for Flow Velocity (U_{in}) equal to 4 m/s

Input experimental values used in ANN learning process									
Design points	L	D	L_f	L_r	R_f	R_r	R_m	C_D	k
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308	0.382805	3.472379
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615	0.378005	3.475310
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923	0.480158	3.526595
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692	0.317865	3.441608
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000	0.467962	3.513840
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308	0.383417	3.474876
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077	0.469070	3.521206
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385	0.354453	3.461241
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692	0.390222	3.476228
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000	0.425666	3.499137
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769	0.494643	3.533857
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077	0.325742	3.445646
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385	0.467013	3.516603
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154	0.388685	3.475622
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462	0.582262	3.584328
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769	0.359310	3.465050
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538	0.406603	3.484238
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846	0.434139	3.503447
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154	0.512892	3.544624
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462	0.364189	3.464409
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231	0.438038	3.502314
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538	0.425089	3.496328
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846	0.537607	3.561799
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615	0.386359	3.476778
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923	0.440502	3.501512
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231	0.457831	3.519007
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000	0.475757	3.532071

Table F.5: Ansys Fluent results for the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) for Flow Velocity (U_{in}) equal to 5 m/s

Input experimental values used in ANN learning process									
Design points	L	D	L_f	L_r	R_f	R_r	R_m	C_D	k
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308	0.575794	5.123599
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615	0.577421	5.147451
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923	0.732087	5.225649
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692	0.478753	5.090375
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000	0.698178	5.195675
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308	0.584096	5.155518
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077	0.704759	5.217022
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385	0.540601	5.118979
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692	0.595312	5.147490
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000	0.642343	5.182548
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769	0.747661	5.224945
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077	0.491530	5.094536
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385	0.705891	5.197011
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154	0.585174	5.141247
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462	0.884380	5.332457
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769	0.542194	5.122397
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538	0.619283	5.150490
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846	0.656478	5.182475
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154	0.784001	5.251793
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462	0.549572	5.119416
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231	0.678542	5.195763
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538	0.646645	5.171892
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846	0.822971	5.328762
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615	0.590082	5.146311
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923	0.665935	5.181451
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231	0.689213	5.207390
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000	0.715628	5.219178

Table F.6: Ansys Fluent results for the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) for Flow Velocity (U_{in}) equal to 6 m/s

Input experimental values used in ANN learning process									
Design points	L	D	L_f	L_r	R_f	R_r	R_m	C_D	k
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308	0.794652	7.067825
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615	0.784321	7.075231
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923	0.995391	7.169621
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692	0.660256	7.008179
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000	0.970797	7.150886
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308	0.793568	7.072703
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077	0.972069	7.164964
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385	0.739404	7.049300
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692	0.810032	7.074890
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000	0.884738	7.123108
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769	1.027343	7.192478
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077	0.676723	7.016158
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385	0.971851	7.156932
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154	0.804613	7.077263
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462	1.207499	7.297197
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769	0.747382	7.052587
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538	0.843314	7.090899
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846	0.906043	7.140275
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154	1.068077	7.210005
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462	0.759794	7.054968
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231	0.911547	7.138162
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538	0.878138	7.110300
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846	1.118972	7.253207
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615	0.800757	7.074690
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923	0.908098	7.125724
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231	0.948976	7.155095
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000	1.002597	7.197261

Table F.7: Ansys Fluent results for the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) for Flow Velocity (U_{in}) equal to 7 m/s

Input experimental values used in ANN learning process									
Design points	L	D	L_f	L_r	R_f	R_r	R_m	C_D	k
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308	1.043755	9.257411
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615	1.031848	9.264347
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923	1.311602	9.402122
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692	0.866577	9.178395
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000	1.268342	9.375689
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308	1.043603	9.265977
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077	1.279707	9.384646
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385	0.972584	9.231149
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692	1.065052	9.269546
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000	1.162114	9.334130
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769	1.361594	9.438195
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077	0.890169	9.190361
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385	1.280264	9.385513
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154	1.056890	9.268993
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462	1.592623	9.579948
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769	0.982657	9.237205
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538	1.106828	9.286324
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846	1.187056	9.348793
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154	1.426387	9.494033
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462	0.997356	9.242821
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231	1.198463	9.334985
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538	1.158456	9.338375
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846	1.481128	9.500232
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615	1.055587	9.266165
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923	1.205792	9.338315
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231	1.245087	9.375955
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000	1.300801	9.406790

Table F.8: Ansys Fluent results for the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) for Flow Velocity (U_{in}) equal to 8 m/s

Input experimental values used in ANN learning process									
Design points	L	D	L_f	L_r	R_f	R_r	R_m	C_D	k
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308	1.319913	11.696540
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615	1.303946	11.707521
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923	1.661356	11.864556
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692	1.098644	11.603073
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000	1.619013	11.833211
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308	1.319727	11.703185
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077	1.617072	11.851503
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385	1.230478	11.664818
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692	1.347209	11.712122
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000	1.468996	11.786899
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769	1.713330	11.895981
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077	1.129534	11.617112
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385	1.627434	11.843465
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154	1.334017	11.708419
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462	2.020516	12.060603
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769	1.242212	11.673582
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538	1.398793	11.736117
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846	1.499838	11.798116
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154	1.784265	11.928596
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462	1.262449	11.675094
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231	1.518532	11.793597
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538	1.461000	11.768591
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846	1.872708	11.988842
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615	1.333110	11.710246
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923	1.518024	11.786682
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231	1.575264	11.834599
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000	1.663555	11.893417

Table F.9: Ansys Fluent results for the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) for Flow Velocity (U_{in}) equal to 9 m/s

Input experimental values used in ANN learning process									
Design points	L	D	L_f	L_r	R_f	R_r	R_m	C_D	k
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308	1.623730	14.374714
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615	1.605731	14.389204
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923	2.046662	14.580880
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692	1.352628	14.261700
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000	1.997502	14.544881
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308	1.624902	14.383428
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077	1.991531	14.562491
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385	1.513250	14.337495
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692	1.657171	14.393642
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000	1.808188	14.484943
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769	2.104320	14.612731
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077	1.388017	14.279427
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385	2.004557	14.554528
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154	1.642547	14.389727
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462	2.490606	14.817008
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769	1.529811	14.348511
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538	1.726634	14.425018
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846	1.847874	14.501942
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154	2.193805	14.654121
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462	1.554509	14.349210
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231	1.871648	14.496234
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538	1.797029	14.463648
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846	2.311187	14.731705
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615	1.643391	14.394370
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923	1.876763	14.489449
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231	1.945339	14.548224
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000	2.049367	14.612981

Table F.10: Ansys Fluent results for the Drag Coefficient (C_D) and Turbulent Kinetic Energy (k) for Flow Velocity (U_{in}) equal to 10 m/s

Input experimental values used in ANN learning process									
Design points	L	D	L_f	L_r	R_f	R_r	R_m	C_D	k
1	0.450000	0.296154	0.223077	0.938462	0.553846	0.412308	9.692308	1.958199	17.288104
2	0.458654	0.317692	0.250000	1.092308	0.666346	0.344615	8.584615	1.935473	17.303844
3	0.467308	0.339231	0.276923	0.830769	0.545192	0.443077	7.476923	2.464225	17.531866
4	0.475962	0.360769	0.200000	0.984615	0.657692	0.375385	10.107692	1.630426	17.153289
5	0.484615	0.382308	0.226923	1.138462	0.536538	0.473846	9.000000	2.405913	17.487819
6	0.493269	0.403846	0.253846	0.876923	0.649038	0.406154	7.892308	1.960952	17.299877
7	0.501923	0.280000	0.280769	1.030769	0.527885	0.338462	10.523077	2.410550	17.530479
8	0.510577	0.301538	0.203846	1.184615	0.640385	0.436923	9.415385	1.822372	17.241667
9	0.519231	0.323077	0.230769	0.923077	0.519231	0.369231	8.307692	1.995536	17.308713
10	0.527885	0.344615	0.257692	1.076923	0.631731	0.467692	7.200000	2.180707	17.419999
11	0.536538	0.366154	0.284615	0.815385	0.510577	0.400000	9.830769	2.535312	17.567048
12	0.545192	0.387692	0.207692	0.969231	0.623077	0.332308	8.723077	1.674630	17.175568
13	0.553846	0.409231	0.234615	1.123077	0.501923	0.430769	7.615385	2.418341	17.502011
14	0.562500	0.285385	0.261538	0.861538	0.614423	0.363077	10.246154	1.978179	17.305572
15	0.571154	0.306923	0.288462	1.015385	0.493269	0.461538	9.138462	2.992453	17.804504
16	0.579808	0.328462	0.211538	1.169231	0.605769	0.393846	8.030769	1.842270	17.256141
17	0.588462	0.350000	0.238462	0.907692	0.484615	0.326154	10.661538	2.081323	17.345754
18	0.597115	0.371538	0.265385	1.061538	0.597115	0.424615	9.553846	2.231119	17.436849
19	0.605769	0.393077	0.292308	0.800000	0.475962	0.338462	8.446154	2.647227	17.622632
20	0.614423	0.414615	0.215385	0.953846	0.588462	0.418462	7.338462	1.873862	17.258062
21	0.623077	0.290769	0.242308	1.107692	0.467308	0.332308	9.969231	2.255931	17.431197
22	0.631731	0.312308	0.269231	0.846154	0.579808	0.412308	8.861538	2.164366	17.391331
23	0.640385	0.333846	0.296154	1.000000	0.458654	0.326154	7.753846	2.781947	17.706992
24	0.649038	0.355385	0.219231	1.153846	0.571154	0.406154	10.384615	1.979479	17.308805
25	0.657692	0.376923	0.246154	0.892308	0.450000	0.320000	9.276923	2.263591	17.423593
26	0.666346	0.398462	0.273077	1.046154	0.562500	0.400000	8.169231	2.341505	17.489524
27	0.675000	0.420000	0.300000	1.200000	0.675000	0.480000	10.800000	2.454450	17.568763

Appendix G

Sensitivity Index (Sobol)

Table G.1: Sensitivity Index values for every Geometrical Variable and Sum of this Index

Velocity	Physical Variables	Geometrical Variables							Sum of Sobol
		R_f	L_f	D	L	R_r	L_r	R_m	
1 m/s	k	0.00008825	0.00755706	0.48961754	0.00422376	0.34941119	0.10634567	0.00125661	0.95850007
	C_D	0.00142055	0.00592207	0.52023182	0.00925733	0.32063910	0.10422027	0.00281396	0.96450511
2 m/s	k	0.01104991	0.00328498	0.54804540	0.02574279	0.30532136	0.07519352	0.00437867	0.97301663
	C_D	0.00126894	0.01594495	0.56906313	0.01122709	0.27353324	0.09724902	0.00159268	0.96987905
3 m/s	k	0.04800942	0.10247749	0.42881531	0.00133361	0.13687988	0.04853452	0.07897356	0.84502379
	C_D	0.00005271	0.00080198	0.33316246	0.00418769	0.37058900	0.03355015	0.01548030	0.75782429
4 m/s	k	0.00090120	0.01077043	0.57609757	0.05451542	0.32784620	0.01676941	0.02766518	1.01456540
	C_D	0.00405650	0.00205722	0.69127572	0.09168592	0.23236479	0.01420269	0.01165832	1.04730117
5 m/s	k	0.00685969	0.00155641	0.51167706	0.07996957	0.27643671	0.02700708	0.00038180	0.90388832
	C_D	0.01639817	0.00042642	0.52508390	0.04430723	0.28956994	0.04419307	0.01033933	0.93031807
6 m/s	k	0.00088470	0.00671946	0.69090047	0.06874913	0.21521207	0.01646809	0.00344459	1.00237852
	C_D	0.00086753	0.00384754	0.66687659	0.07113662	0.21992683	0.02454650	0.00359219	0.99079381
7 m/s	k	0.00090988	0.00146882	0.63313403	0.03958060	0.23039906	0.04068045	0.00564437	0.95181720
	C_D	0.00069628	0.00121002	0.62623417	0.04146936	0.22476934	0.04884804	0.00626161	0.94948884
8 m/s	k	0.00305515	0.01458161	0.45437439	0.01778511	0.47313170	0.04208042	0.06634847	1.07135685
	C_D	0.00523617	0.00031991	0.51182573	0.02727866	0.48795810	0.03567975	0.00948676	1.07778507
9 m/s	k	0.01056930	0.00969851	0.70168576	0.08650876	0.17195207	0.03241855	0.00065923	1.01349217
	C_D	0.00780423	0.01077237	0.71841828	0.06887202	0.20806261	0.01723088	0.00098722	1.03214761
10 m/s	k	0.01341899	0.00433286	0.63769916	0.06940722	0.23401317	0.01343834	0.00215168	0.97446141
	C_D	0.00933560	0.00218234	0.65981985	0.06512942	0.22364190	0.01540173	0.00098748	0.97649833

Table G.2: Sensitivity Index percentage values (%) for every Geometrical Variable

Velocity	Physical Variables	Geometrical Variables						
		R_f	L_f	D	L	R_r	L_r	R_m
1 m/s	k	0.0092	0.7884	51.0816	0.4407	36.4540	11.0950	0.1311
	C_D	0.1473	0.6140	53.9377	0.9598	33.2439	10.8056	0.2918
2 m/s	k	1.1356	0.3376	56.3244	2.6457	31.3788	7.7279	0.4500
	C_D	0.1308	1.6440	58.6736	1.1576	28.2028	10.0269	0.1642
3 m/s	k	5.6814	12.1272	50.7459	0.1578	16.1983	5.7436	9.3457
	C_D	0.0070	0.1058	43.9630	0.5526	48.9017	4.4272	2.0427
4 m/s	k	0.0888	1.0616	56.7827	5.3733	32.3140	1.6529	2.7268
	C_D	0.3873	0.1964	66.0054	8.7545	22.1870	1.3561	1.1132
5 m/s	k	0.7589	0.1722	56.6084	8.8473	30.5831	2.9879	0.0422
	C_D	1.7626	0.0458	56.4413	4.7626	31.1259	4.7503	1.1114
6 m/s	k	0.0883	0.6704	68.9261	6.8586	21.4701	1.6429	0.3436
	C_D	0.0876	0.3883	67.3073	7.1798	22.1970	2.4775	0.3626
7 m/s	k	0.0956	0.1543	66.5184	4.1584	24.2062	4.2740	0.5930
	C_D	0.0733	0.1274	65.9549	4.3675	23.6727	5.1447	0.6595
8 m/s	k	0.2852	1.3610	42.4111	1.6601	44.1619	3.9278	6.1929
	C_D	0.4858	0.0297	47.4887	2.5310	45.2742	3.3105	0.8802
9 m/s	k	1.0429	0.9569	69.2345	8.5357	16.9663	3.1987	0.0650
	C_D	0.7561	1.0437	69.6042	6.6727	20.1582	1.6694	0.0956
10 m/s	k	1.3771	0.4446	65.4412	7.1226	24.0146	1.3791	0.2208
	C_D	0.9560	0.2235	67.5700	6.6697	22.9024	1.5772	0.1011

Table G.3: Sensitivity index values for all the flow velocities and geometrical input variables regarding the output variable C_D

	Drag Coefficient - C_D						
Velocity	Geometrical Variables						
(m/s)	R_f	L_f	D	L	R_r	L_r	R_m
1	0.1473	0.6140	53.9377	0.9598	33.2439	10.8056	0.2918
2	0.1308	1.6440	58.6736	1.1576	28.2028	10.0269	0.1642
3	0.0070	0.1058	43.9630	0.5526	48.9017	4.4272	2.0427
4	0.3873	0.1964	66.0054	8.7545	22.1870	1.3561	1.1132
5	1.7626	0.0458	56.4413	4.7626	31.1259	4.7503	1.1114
6	0.0876	0.3883	67.3073	7.1798	22.1970	2.4775	0.3626
7	0.0733	0.1274	65.9549	4.3675	23.6727	5.1447	0.6595
8	0.4858	0.0297	47.4887	2.5310	45.2742	3.3105	0.8802
9	0.7561	1.0437	69.6042	6.6727	20.1582	1.6694	0.0956
10	0.9560	0.2235	67.5700	6.6697	22.9024	1.5772	0.1011

Table G.4: Sensitivity index values for all the flow velocities and geometrical input variables regarding the output variable k

	Turbulent kinetic energy - k						
Velocity	Geometrical Variables						
(m/s)	R_f	L_f	D	L	R_r	L_r	R_m
1	0.0092	0.7884	51.0816	0.4407	36.4540	11.0950	0.1311
2	1.1356	0.3376	56.3244	2.6457	31.3788	7.7279	0.4500
3	5.6814	12.1272	50.7459	0.1578	16.1983	5.7436	9.3457
4	0.0888	1.0616	56.7827	5.3733	32.3140	1.6529	2.7268
5	0.7589	0.1722	56.6084	8.8473	30.5831	2.9879	0.0422
6	0.0883	0.6704	68.9261	6.8586	21.4701	1.6429	0.3436
7	0.0956	0.1543	66.5184	4.1584	24.2062	4.2740	0.5930
8	0.2852	1.3610	42.4111	1.6601	44.1619	3.9278	6.1929
9	1.0429	0.9569	69.2345	8.5357	16.9663	3.1987	0.0650
10	1.3771	0.4446	65.4412	7.1226	24.0146	1.3791	0.2208

Table G.5: Maximum and minimum sensitivity index values for all the geometrical variables regarding both output variables: C_D and k

Output Variable		C_D		k	
Sobol Index		Max	Min	Max	Min
Geometrical Variables	R_f	1.762641	0.006956	5.681428	0.009207
	L_f	1.644015	0.029682	12.12717	0.154317
	D	69.60422	43.96302	69.23445	42.41112
	L	8.754494	0.552593	8.847284	0.157819
	R_r	48.90171	20.15822	44.16191	16.19835
	L_r	10.80557	1.356123	11.09501	1.379053
	R_m	2.04273	0.095647	9.345721	0.04224

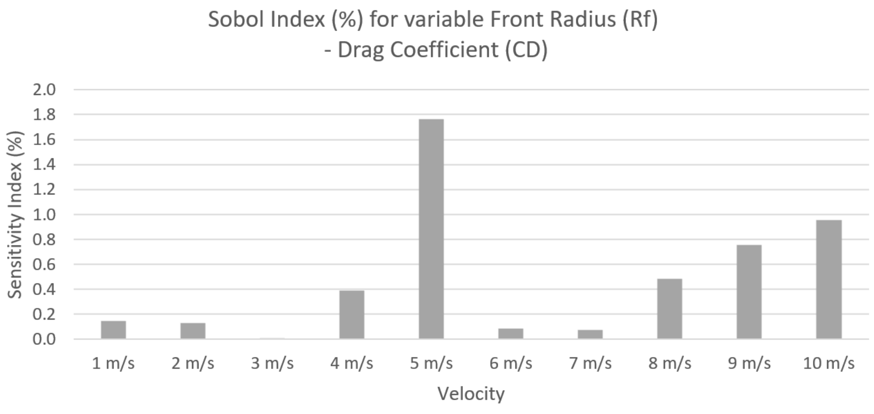


Figure G.1: Sensitivity index percentage value obtained for the Drag Coefficient (C_D) in order of the Front Radius (R_f) geometrical variable and the ten different flow velocities

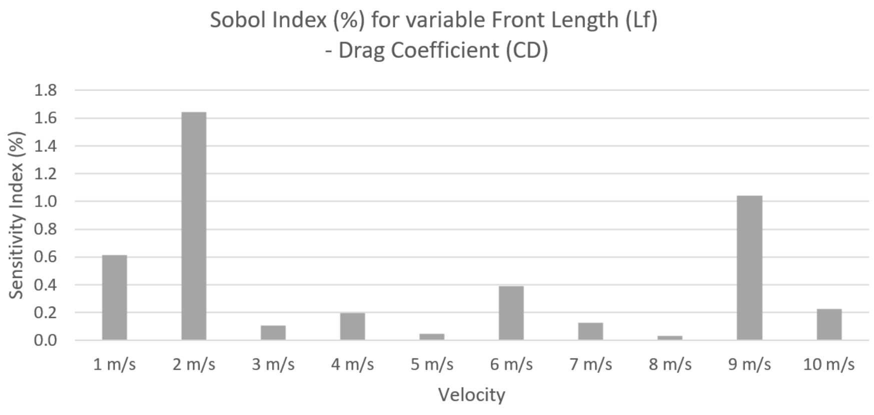


Figure G.2: Sensitivity index percentage value obtained for the Drag Coefficient (C_D) as a function of the Front Length (L_f) geometrical variable and the ten different flow velocities

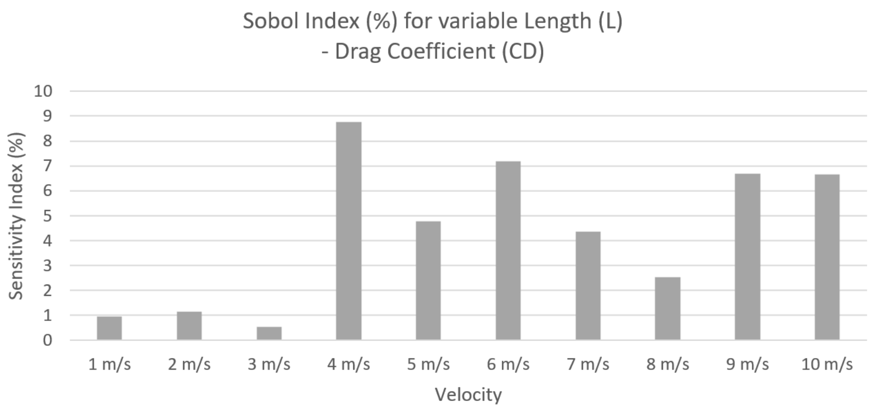


Figure G.3: Sensitivity index percentage value obtained for the Drag Coefficient (C_D) as a function of the Length (L) geometrical variable and the ten different flow velocities

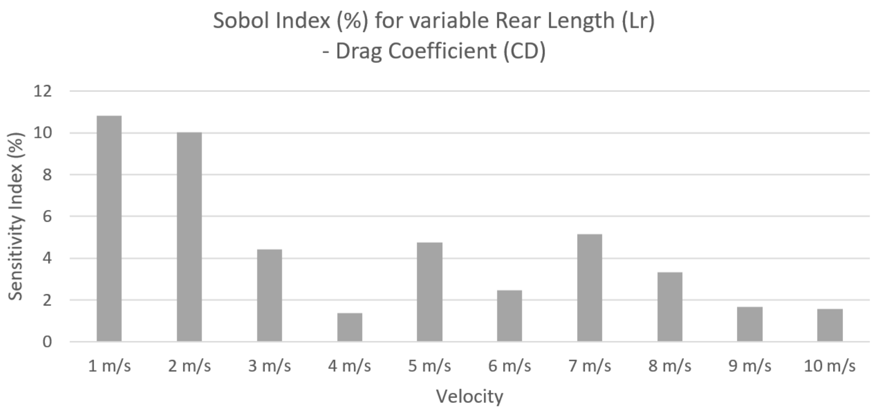


Figure G.4: Sensitivity index percentage value obtained for the Drag Coefficient (C_D) as a function of the Rear Length (L_r) geometrical variable and the ten different flow velocities

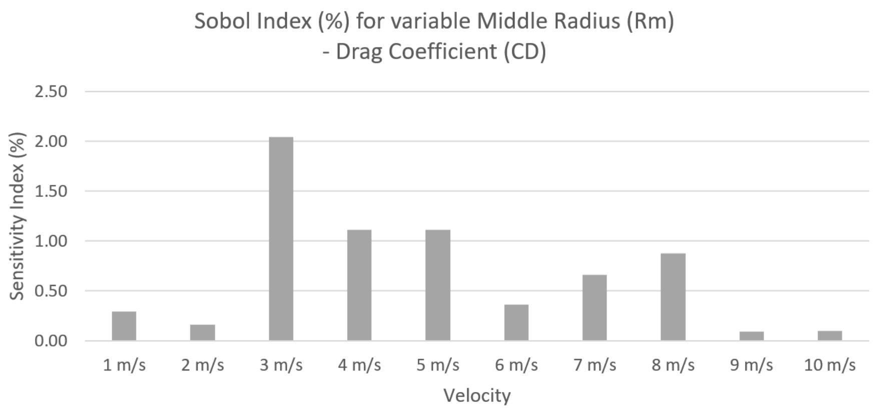


Figure G.5: Sensitivity index percentage value obtained for the Drag Coefficient (C_D) as a function of the Middle Radius (R_m) geometrical variable and the ten different flow velocities

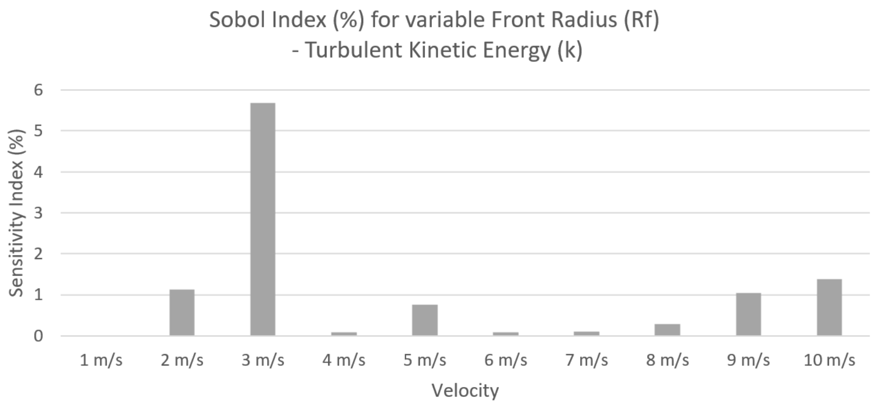


Figure G.6: Sensitivity index percentage value obtained for the Turbulent Kinetic Energy (k) as a function of the Front Radius (R_f) geometrical variable and the ten different flow velocities

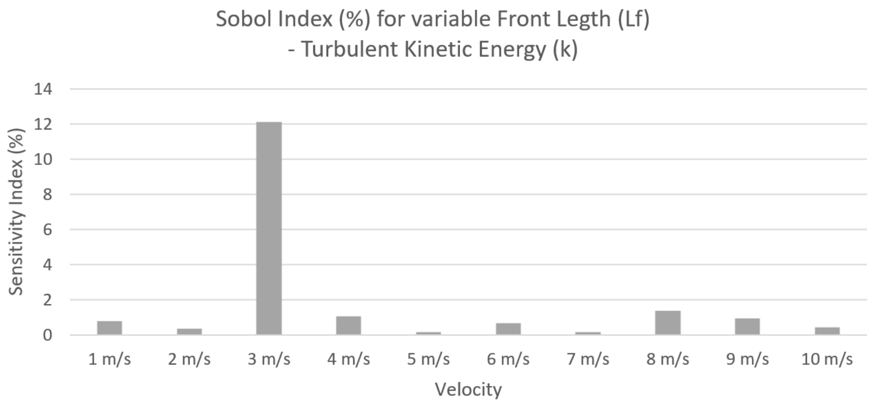


Figure G.7: Sensitivity index percentage value obtained for the Turbulent Kinetic Energy (k) as a function of the Front Length (L_f) geometrical variable and the ten different flow velocities

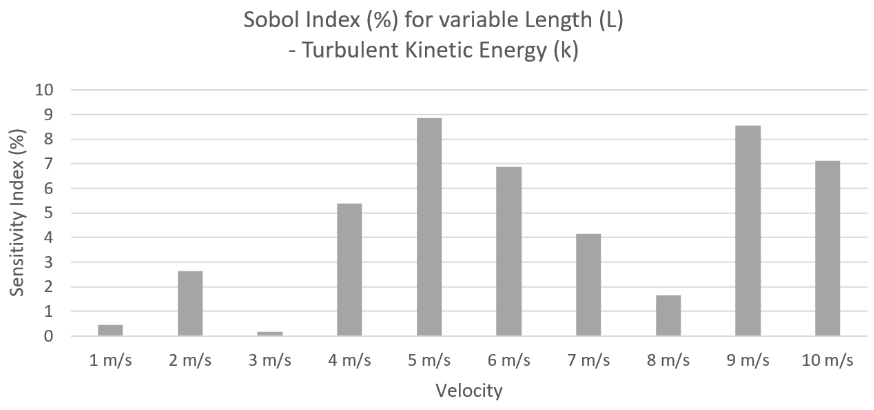


Figure G.8: Sensitivity index percentage value obtained for the Turbulent Kinetic Energy (k) as a function of the Length (L) geometrical variable and the ten different flow velocities

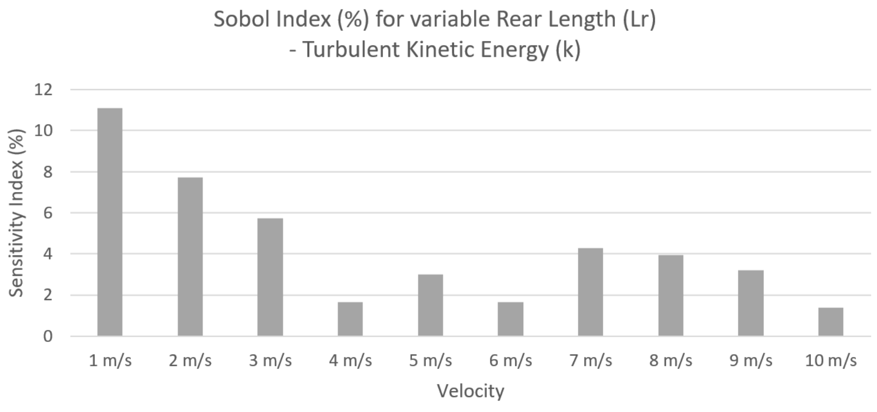


Figure G.9: Sensitivity index percentage value obtained for the Turbulent Kinetic Energy (k) as a function of the Rear Length (L_r) geometrical variable and the ten different flow velocities

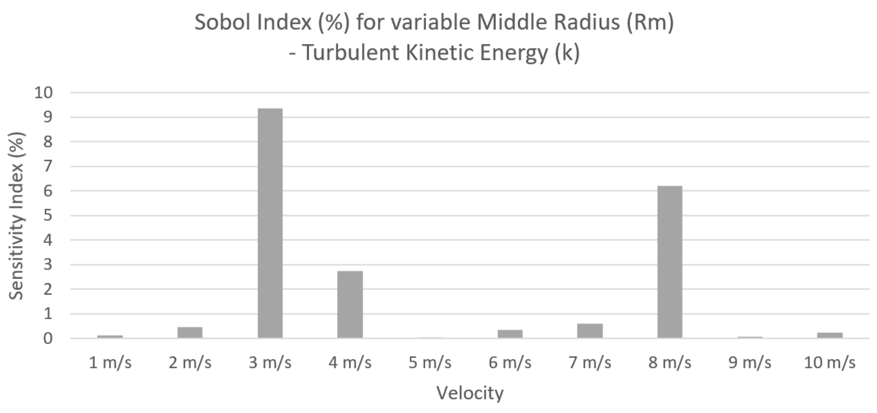


Figure G.10: Sensitivity index percentage value obtained for the Turbulent Kinetic Energy (k) as a function of the Middle Radius (R_m) geometrical variable and the ten different flow velocities

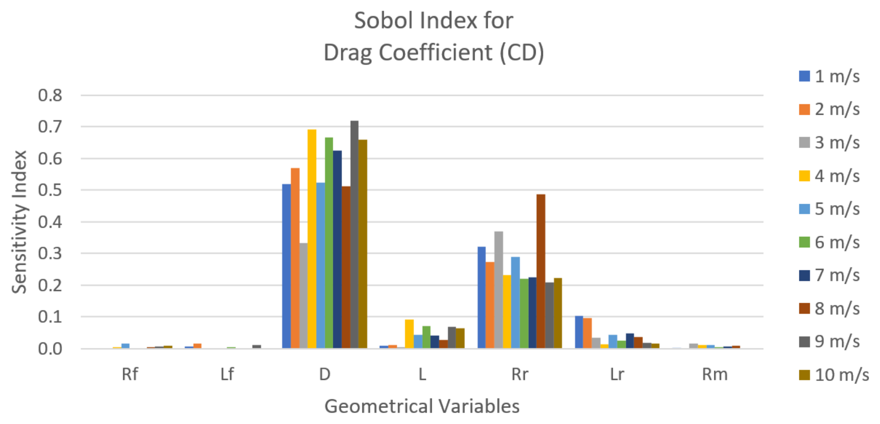


Figure G.11: Global sensitivity index value obtained for the Drag Coefficient (C_D) as a function of the seven different geometrical variables and the ten different flow velocities

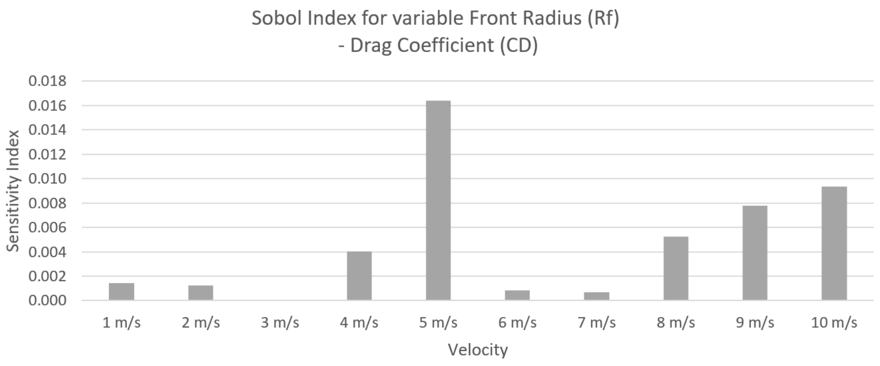


Figure G.12: Sensitivity index value obtained for the Drag Coefficient (C_D) as a function of the Front Radius (R_f) geometrical variable and the ten different flow velocities

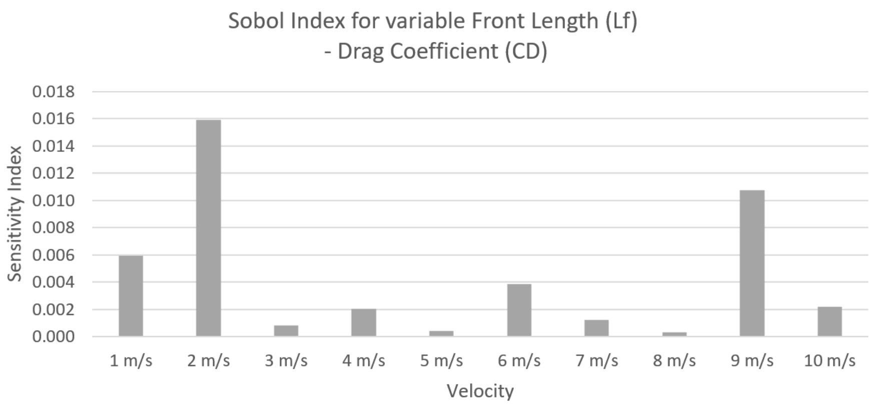


Figure G.13: Sensitivity index value obtained for the Drag Coefficient (C_D) as a function of the Front Length (L_f) geometrical variable and the ten different flow velocities

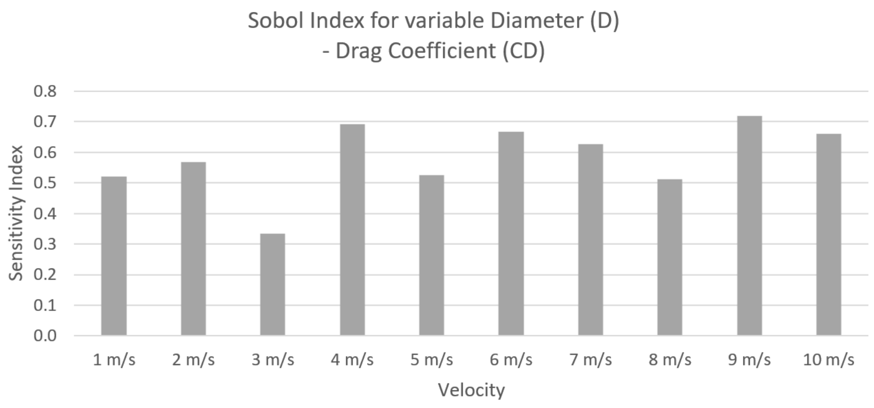


Figure G.14: Sensitivity index value obtained for the Drag Coefficient (C_D) as a function of the Diameter (D) geometrical variable and the ten different flow velocities

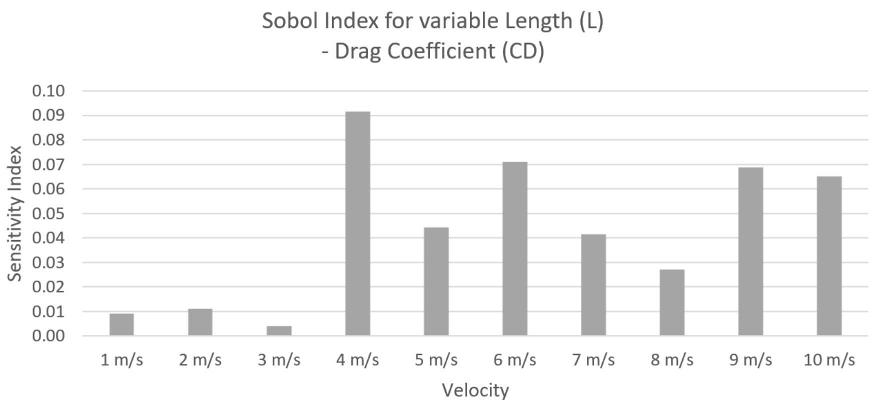


Figure G.15: Sensitivity index value obtained for the Drag Coefficient (C_D) as a function of the Length (L) geometrical variable and the ten different flow velocities

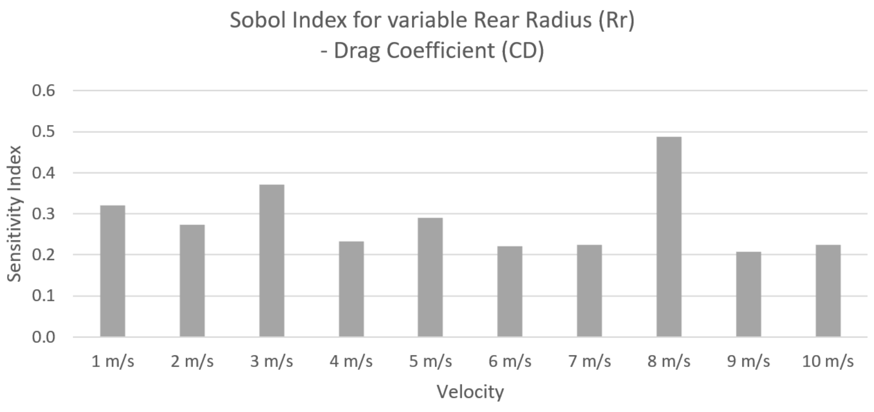


Figure G.16: Sensitivity index value obtained for the Drag Coefficient (C_D) as a function of the Rear Radius (R_r) geometrical variable and the ten different flow velocities

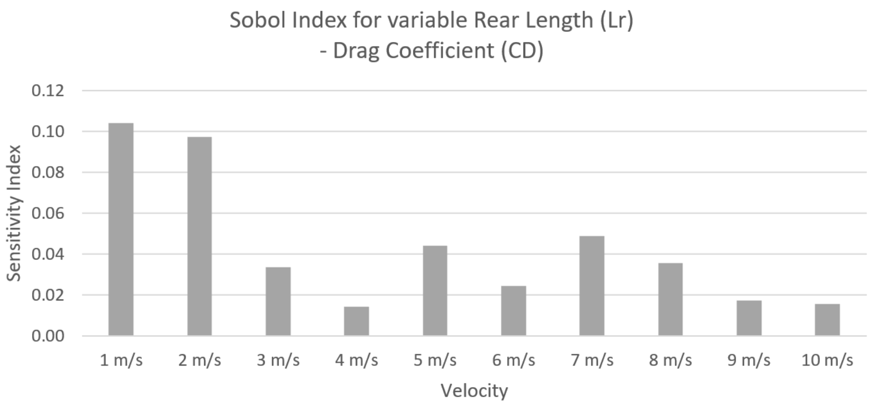


Figure G.17: Sensitivity index value obtained for the Drag Coefficient (C_D) as a function of the Rear Length (L_r) geometrical variable and the ten different flow velocities

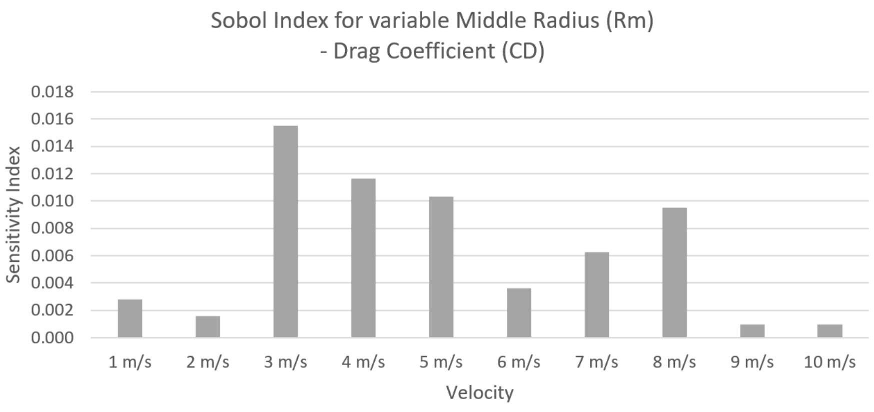


Figure G.18: Sensitivity index value obtained for the Drag Coefficient (C_D) as a function of the Middle Radius (R_m) geometrical variable and the ten different flow velocities

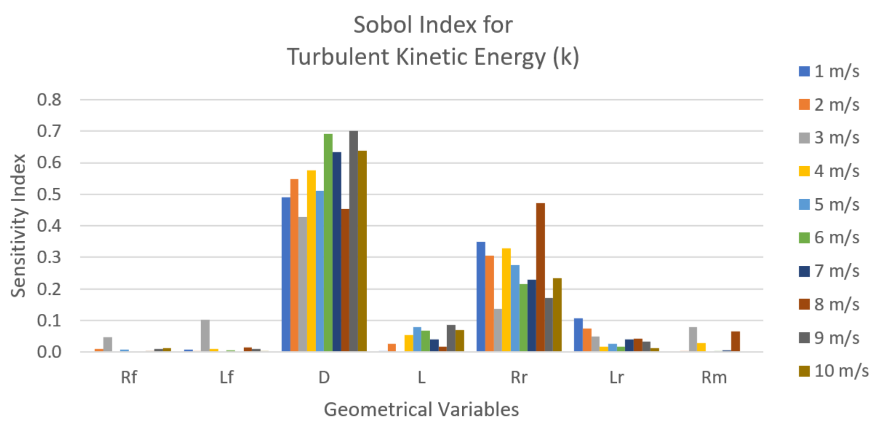


Figure G.19: Global sensitivity value index obtained for the Turbulent Kinetic Energy (k) as a function of the seven different geometrical variables and the ten different flow velocities

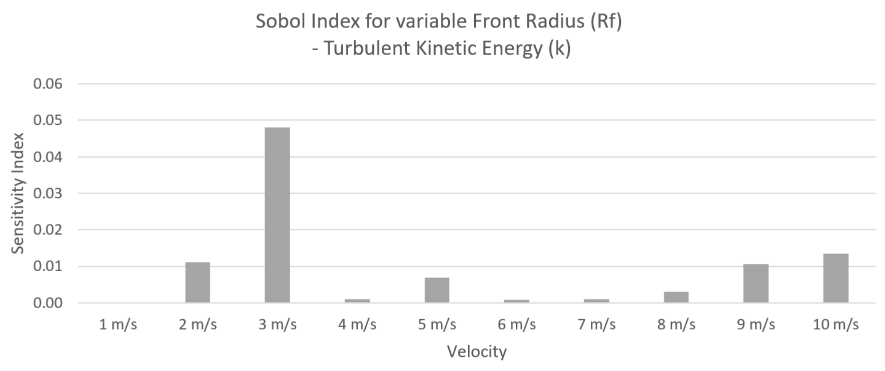


Figure G.20: Sensitivity index value obtained for the Turbulent Kinetic Energy (k) as a function of the Front Radius (R_f) geometrical variable and the ten different flow velocities

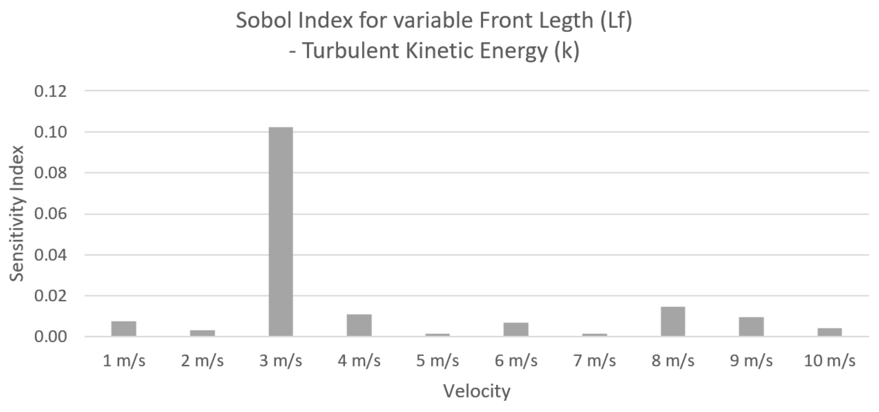


Figure G.21: Sensitivity index value obtained for the Turbulent Kinetic Energy (k) as a function of the Front Length (L_f) geometrical variable and the ten different flow velocities

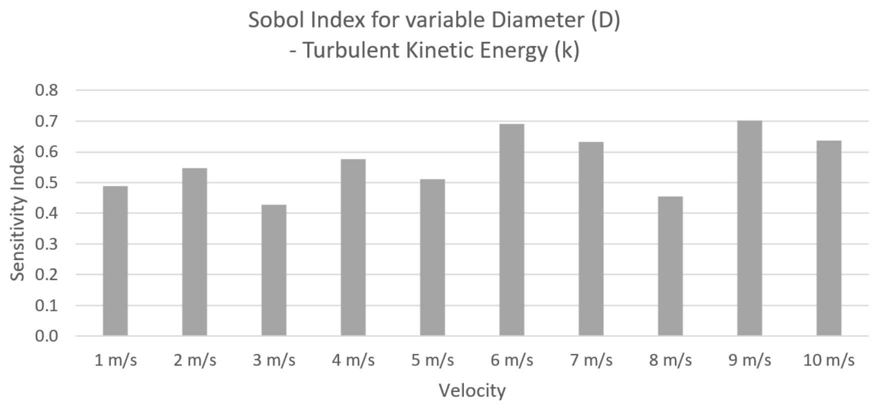


Figure G.22: Sensitivity index value obtained for the Turbulent Kinetic Energy (k) as a function of the Diameter (D) geometrical variable and the ten different flow velocities

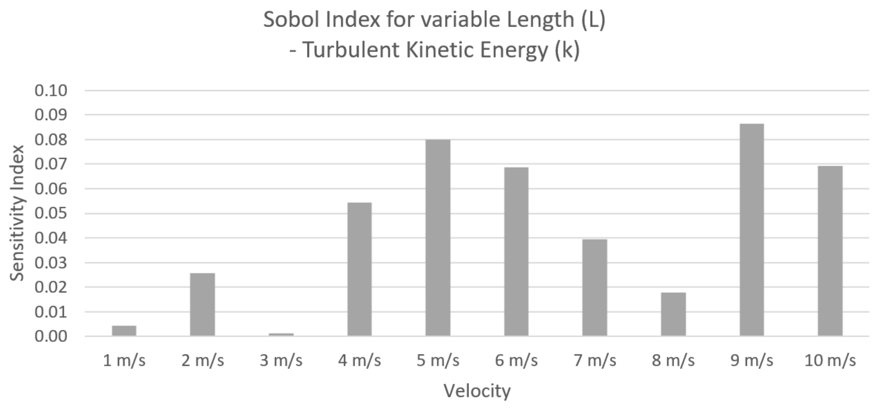


Figure G.23: Sensitivity index value obtained for the Turbulent Kinetic Energy (k) as a function of the Length (L) geometrical variable and the ten different flow velocities

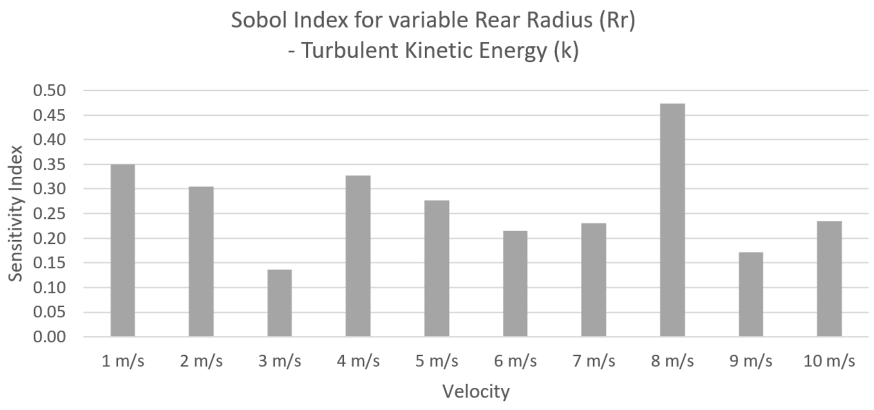


Figure G.24: Sensitivity index value obtained for the Turbulent Kinetic Energy (k) as a function of the Rear Radius (R_r) geometrical variable and the ten different flow velocities

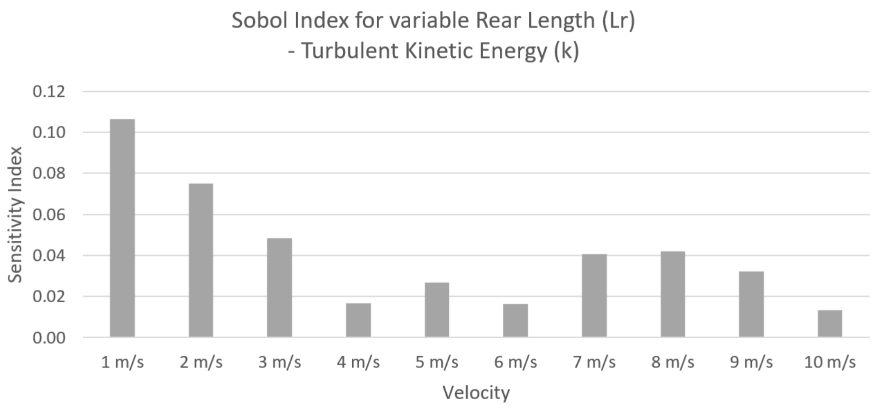


Figure G.25: Sensitivity index value obtained for the Turbulent Kinetic Energy (k) as a function of the Rear Length (L_f) geometrical variable and the ten different flow velocities

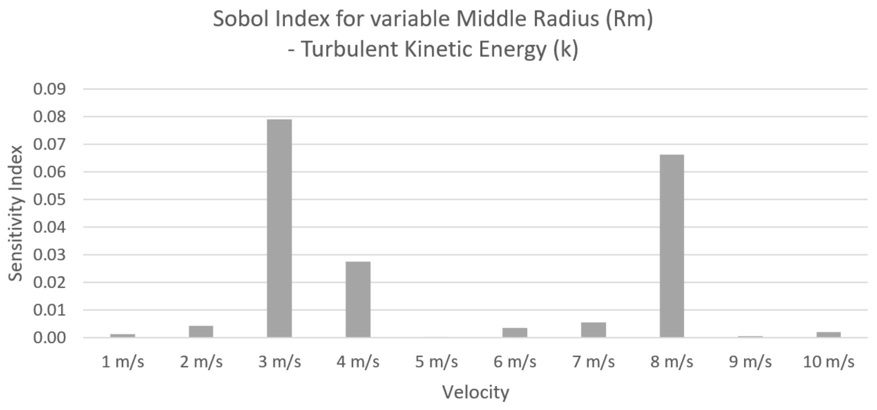


Figure G.26: Sensitivity index value obtained for the Turbulent Kinetic Energy (k) as a function of the Middle Radius (R_m) geometrical variable and the ten different flow velocities

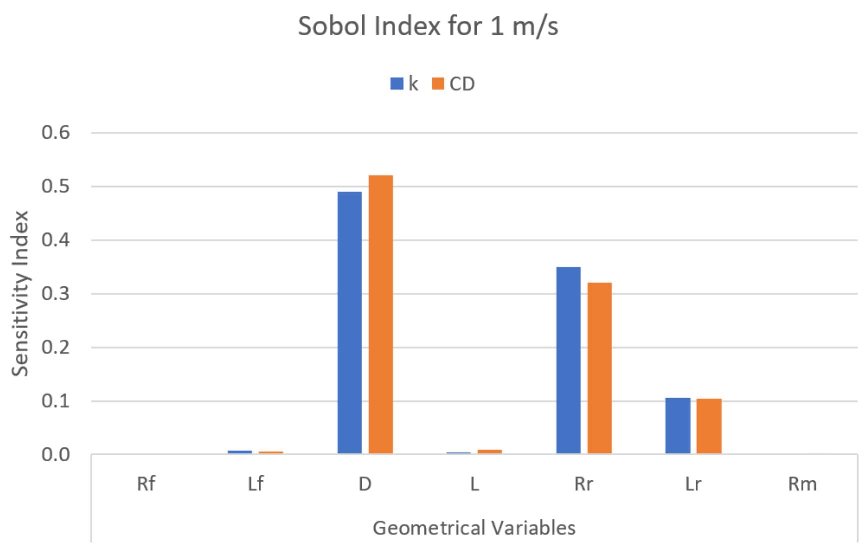


Figure G.27: Sensitivity index value obtained for velocity equal to 1 m/s for both output variables: C_D and k

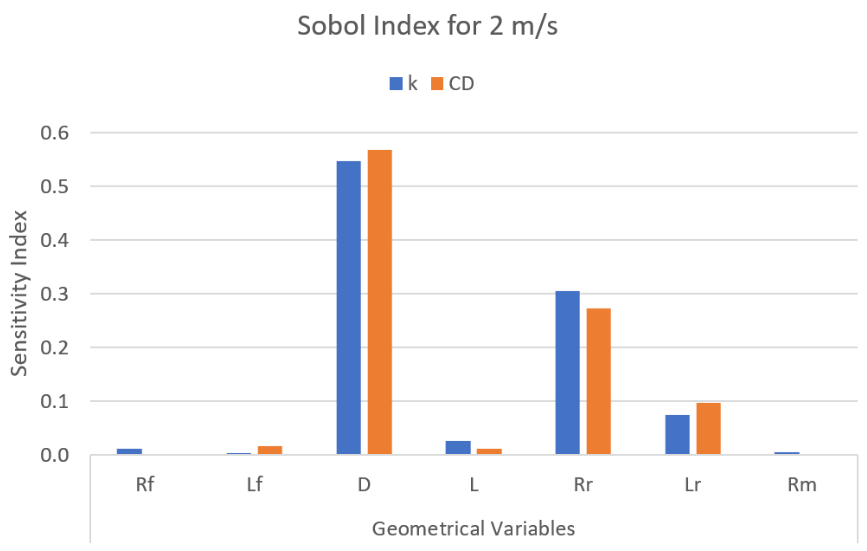


Figure G.28: Sensitivity index value obtained for velocity equal to 2 m/s for both output variables: C_D and k

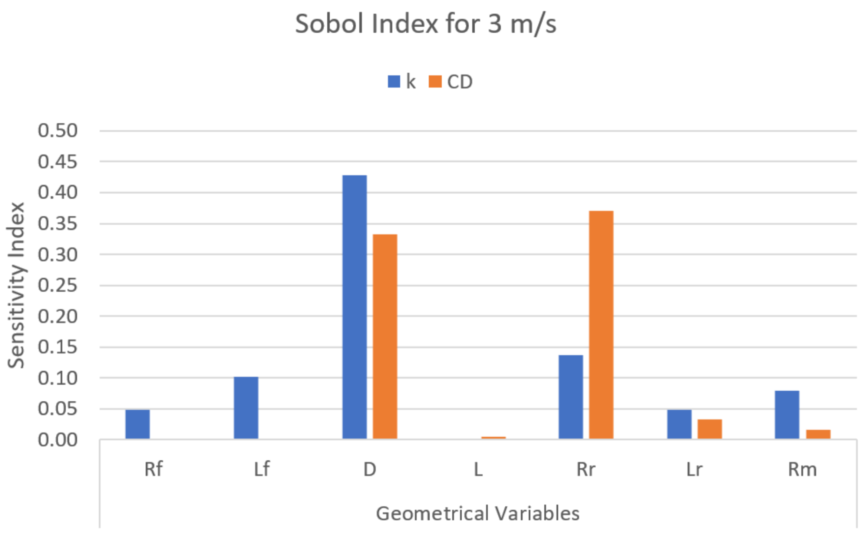


Figure G.29: Sensitivity index value obtained for velocity equal to 3 m/s for both output variables: C_D and k

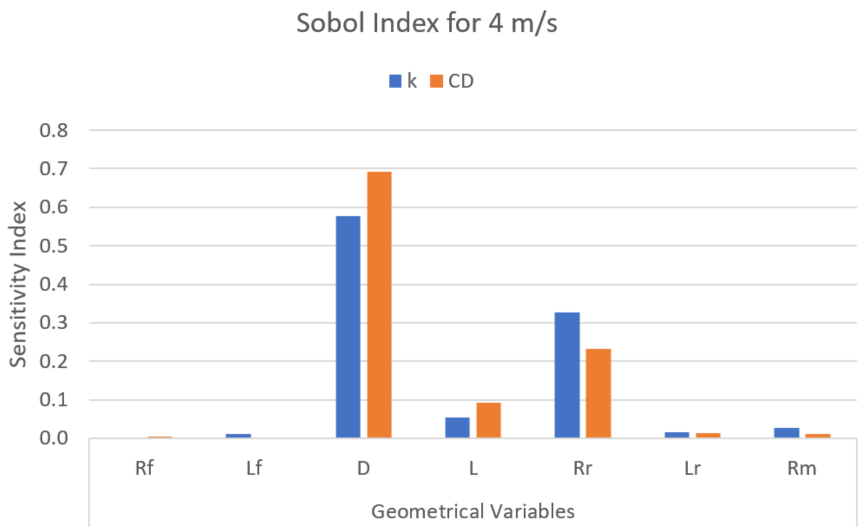


Figure G.30: Sensitivity index value obtained for velocity equal to 4 m/s for both output variables: C_D and k

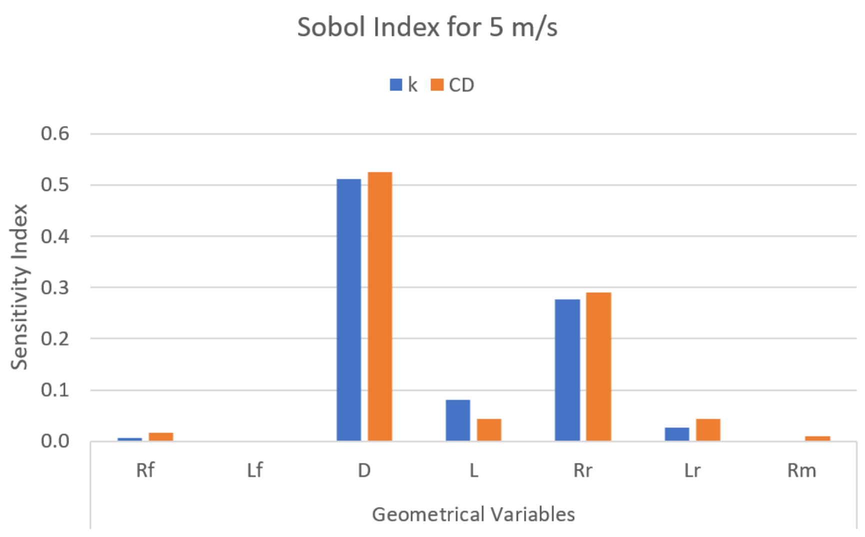


Figure G.31: Sensitivity index value obtained for velocity equal to 5 m/s for both output variables: C_D and k

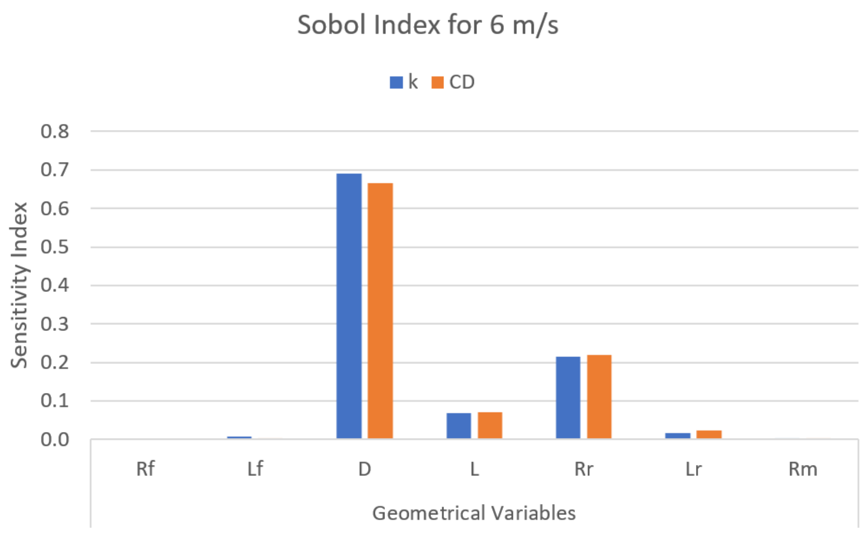


Figure G.32: Sensitivity index value obtained for velocity equal to 6 m/s for both output variables: C_D and k

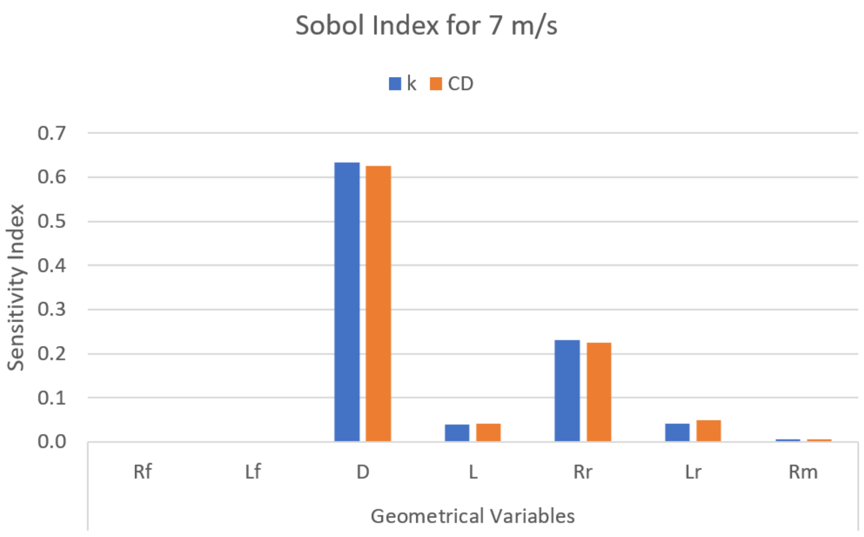


Figure G.33: Sensitivity index value obtained for velocity equal to 7 m/s for both output variables: C_D and k

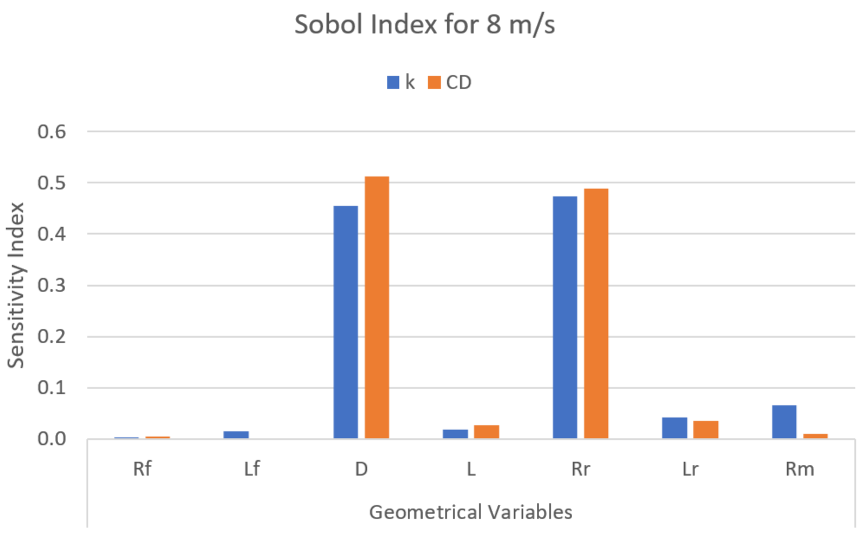


Figure G.34: Sensitivity index value obtained for velocity equal to 8 m/s for both output variables: C_D and k

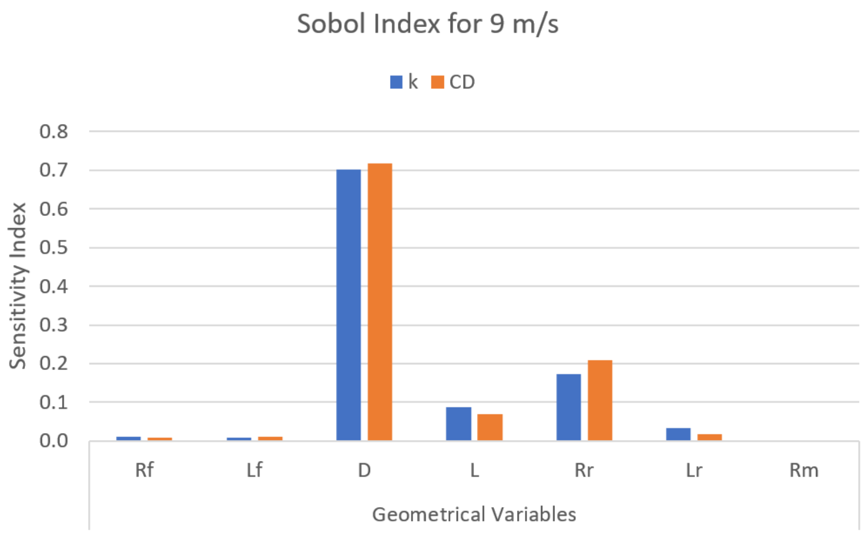


Figure G.35: Sensitivity index value obtained for velocity equal to 9 m/s for both output variables: C_D and k

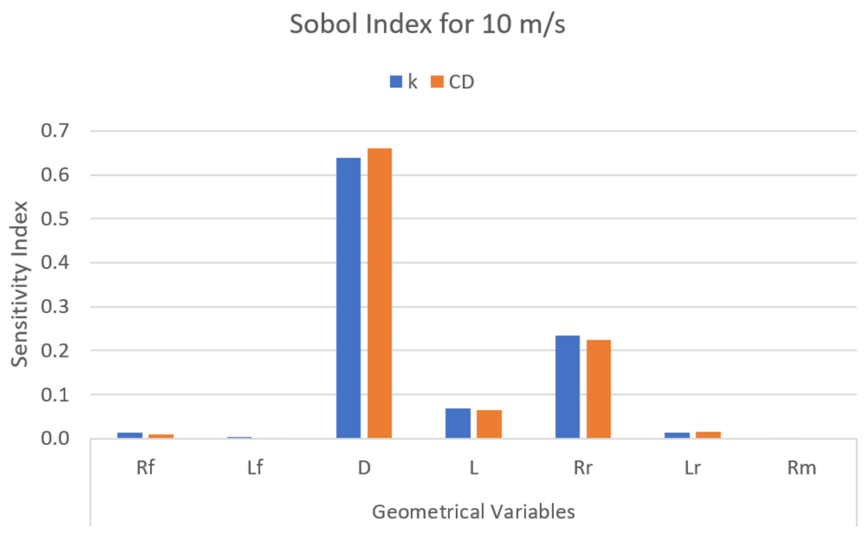


Figure G.36: Sensitivity index value obtained for velocity equal to 10 m/s for both output variables: C_D and k

Appendix H

Optimization Results

Table H.1: Optimal C_D and k values obtained from the Artificial Neural Network (ANN)

Velocity (m/s)	R_f	L_f	D	L	R_r	L_r	R_m	Solution	C_D	k
1	0.527661	0.373484	0.231481	1.080000	0.607500	0.354581	9.371613	1	0.028379	0.303270
2	0.498629	0.378000	0.231481	1.080000	0.607500	0.357161	8.268387	1	0.101740	1.024300
	0.513145	0.375742	0.231481	1.080000	0.607500	0.352000	8.790968	2	0.101630	1.024500
3	0.567581	0.364452	0.231481	1.080000	0.607500	0.352000	9.429677	1	0.207820	2.089300
	0.502258	0.378000	0.231481	1.080000	0.600242	0.362323	9.603871	2	0.213830	2.087000
	0.542177	0.357677	0.231481	1.080000	0.603871	0.357161	9.661935	3	0.209010	2.088800
	0.509516	0.375742	0.232873	1.073548	0.607500	0.375226	9.720000	4	0.212850	2.087300
	0.560323	0.348645	0.231481	1.080000	0.607500	0.354581	9.603871	5	0.207740	2.090300
	0.531290	0.378000	0.231481	1.080000	0.596613	0.364903	9.720000	6	0.212310	2.087500
	0.502258	0.341871	0.231481	1.080000	0.607500	0.354581	9.720000	7	0.209040	2.088300
	0.545806	0.366710	0.231481	1.080000	0.600242	0.354581	9.720000	8	0.209270	2.088000
4	0.520403	0.375742	0.231481	1.080000	0.603871	0.372645	9.720000	1	0.375650	3.465700
	0.513145	0.364452	0.231481	1.080000	0.607500	0.352000	9.487742	2	0.373010	3.465800
	0.516774	0.321548	0.231481	1.080000	0.603871	0.354581	9.720000	3	0.370860	3.470000
	0.516774	0.310258	0.231481	1.080000	0.607500	0.354581	9.371613	4	0.369960	3.471300
	0.524032	0.344129	0.231481	1.080000	0.603871	0.352000	9.720000	5	0.372520	3.468100
	0.553065	0.314774	0.231481	1.080000	0.607500	0.352000	9.545806	6	0.368320	3.471800
	0.553065	0.364452	0.231481	1.080000	0.607500	0.352000	9.661935	7	0.372710	3.466600
5	0.607500	0.378000	0.231481	1.080000	0.607500	0.354581	8.384516	1	0.541510	5.125700
6	0.534919	0.375742	0.232873	1.073548	0.603871	0.352000	9.720000	1	0.752290	7.056900
7	0.578468	0.373484	0.231481	1.080000	0.607500	0.352000	9.545806	1	0.980130	9.237600
8	0.531290	0.364452	0.231481	1.080000	0.607500	0.352000	9.720000	1	1.242900	11.655000
	0.538548	0.378000	0.232873	1.073548	0.607500	0.352000	9.661935	2	1.244700	11.654000
9	0.596613	0.373484	0.234281	1.067097	0.603871	0.352000	9.603871	1	1.547200	14.350000
10	0.603871	0.312516	0.231481	1.080000	0.603871	0.357161	9.661935	1	1.891700	17.269000

Table H.2: Optimal C_D and k values obtained from the CFD simulation in Ansys

Velocity (m/s)	R_f	L_f	D	L	R_r	L_r	R_m	Solution	C_D	k
1	0.527661	0.373484	0.231481	1.080000	0.607500	0.354581	9.371613	1	0.030063	0.305364
2	0.498629	0.378000	0.231481	1.080000	0.607500	0.357161	8.268387	1	0.103048	1.027315
	0.513145	0.375742	0.231481	1.080000	0.607500	0.352000	8.790968	2	0.103140	1.027268
3	0.567581	0.364452	0.231481	1.080000	0.607500	0.352000	9.429677	1	0.217758	2.093621
	0.502258	0.378000	0.231481	1.080000	0.600242	0.362323	9.603871	2	0.222013	2.095946
	0.542177	0.357677	0.231481	1.080000	0.603871	0.357161	9.661935	3	0.218730	2.094117
	0.509516	0.375742	0.232873	1.073548	0.607500	0.375226	9.720000	4	0.222206	2.095445
	0.560323	0.348645	0.231481	1.080000	0.607500	0.354581	9.603871	5	0.216644	2.093321
	0.531290	0.378000	0.231481	1.080000	0.596613	0.364903	9.720000	6	0.221714	2.095193
	0.502258	0.341871	0.231481	1.080000	0.607500	0.354581	9.720000	7	0.217502	2.093266
	0.545806	0.366710	0.231481	1.080000	0.600242	0.354581	9.720000	8	0.218824	2.093908
4	0.520403	0.375742	0.231481	1.080000	0.603871	0.372645	9.720000	1	0.391816	3.473672
	0.513145	0.364452	0.231481	1.080000	0.607500	0.352000	9.487742	2	0.371005	3.468521
	0.516774	0.321548	0.231481	1.080000	0.603871	0.354581	9.720000	3	0.364627	3.468209
	0.516774	0.310258	0.231481	1.080000	0.607500	0.354581	9.371613	4	0.363407	3.466655
	0.524032	0.344129	0.231481	1.080000	0.603871	0.352000	9.720000	5	0.378753	3.479759
	0.553065	0.314774	0.231481	1.080000	0.607500	0.352000	9.545806	6	0.364721	3.466701
	0.553065	0.364452	0.231481	1.080000	0.607500	0.352000	9.661935	7	0.384093	3.488198
5	0.607500	0.378000	0.231481	1.080000	0.607500	0.354581	8.384516	1	0.569600	5.142752
6	0.534919	0.375742	0.232873	1.073548	0.603871	0.352000	9.720000	1	0.799430	7.066403
7	0.578468	0.373484	0.231481	1.080000	0.607500	0.352000	9.545806	1	1.013080	9.25186
8	0.531290	0.364452	0.231481	1.080000	0.607500	0.352000	9.720000	1	1.274754	11.68566
	0.538548	0.378000	0.232873	1.073548	0.607500	0.352000	9.661935	2	1.285242	11.69200
9	0.596613	0.373484	0.234281	1.067097	0.603871	0.352000	9.603871	1	1.610725	14.36151
10	0.603871	0.312516	0.231481	1.080000	0.603871	0.357161	9.661935	1	1.891412	17.28877

Table H.3: Percentage difference between C_D and k obtained from ANN and Ansys for all the optimal solutions

Velocity (m/s)	Solution	Difference (%) C_D	Difference (%) k
1	1	5.600154	0.685723
2	1	1.269216	0.293445
	2	1.463590	0.269423
3	1	4.563962	0.206375
	2	3.685799	0.426824
	3	4.443883	0.253921
	4	4.210400	0.388700
	5	4.109747	0.144297
	6	4.241651	0.367150
	7	3.890398	0.237227
	8	4.365944	0.282147
4	1	4.125919	0.229506
	2	0.537508	0.078443
	3	1.680572	0.051611
	4	1.771248	0.133817
	5	1.645695	0.335063
	6	0.977194	0.146866
	7	2.963689	0.619159
5	1	4.931476	0.331581
6	1	5.896719	0.134479
7	1	3.252477	0.154132
8	1	2.498866	0.262339
	2	3.154395	0.324992
9	1	3.943900	0.080110
10	1	0.015209	0.114323

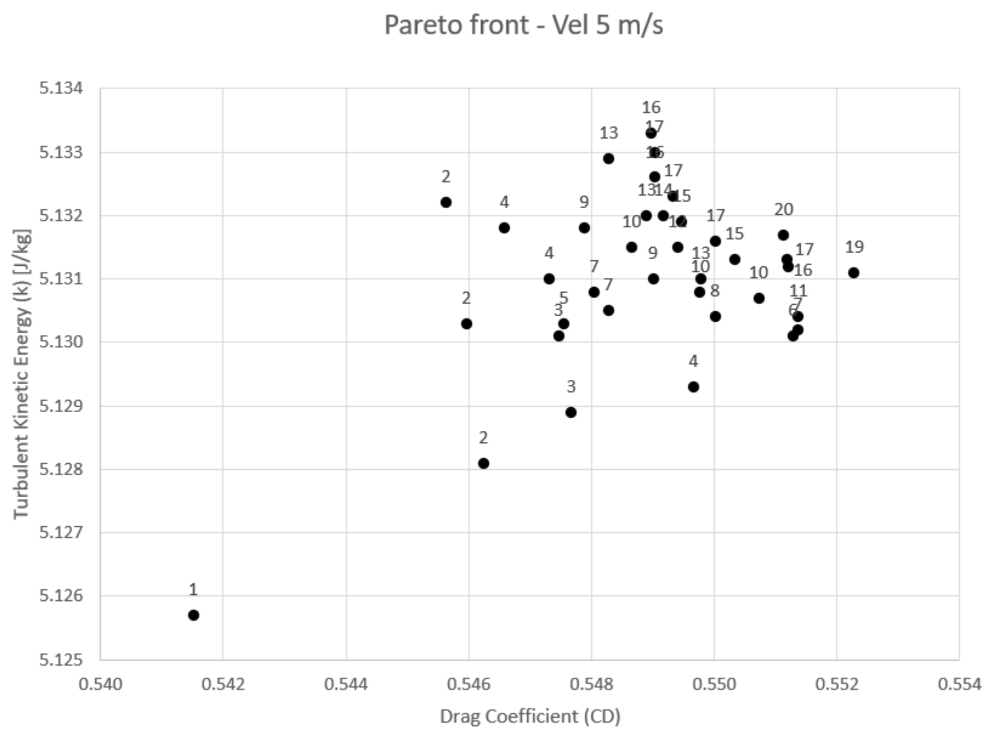


Figure H.1: Pareto efficiency - Optimal solution distribution plot for U_{in} equal to 5 m/s

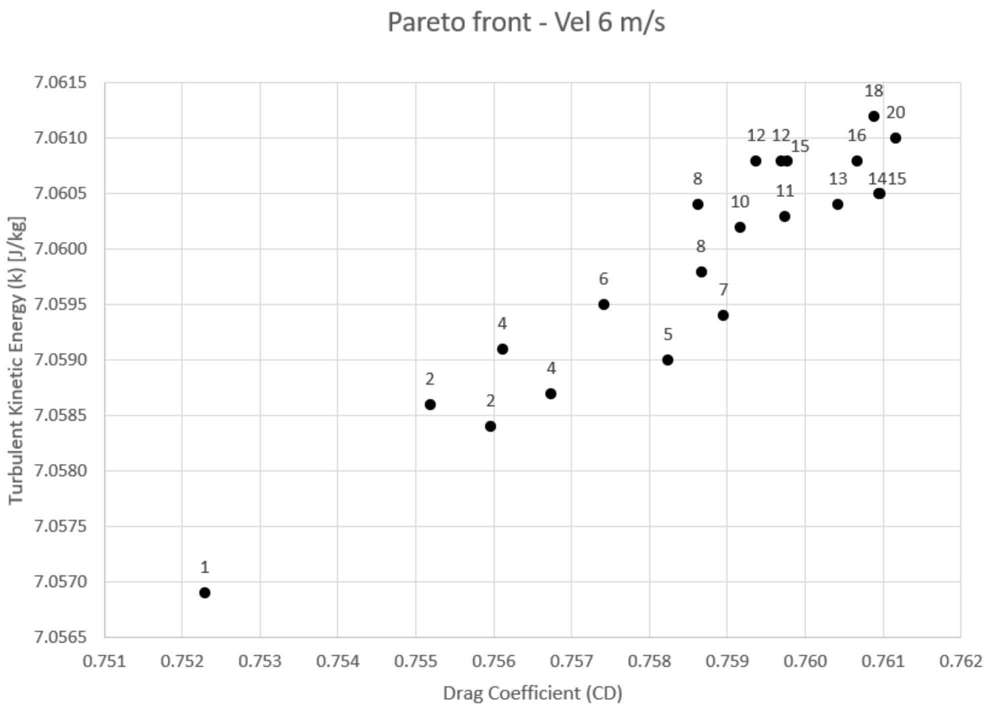


Figure H.2: Pareto efficiency - Optimal solution distribution plot for U_{in} equal to 6 m/s

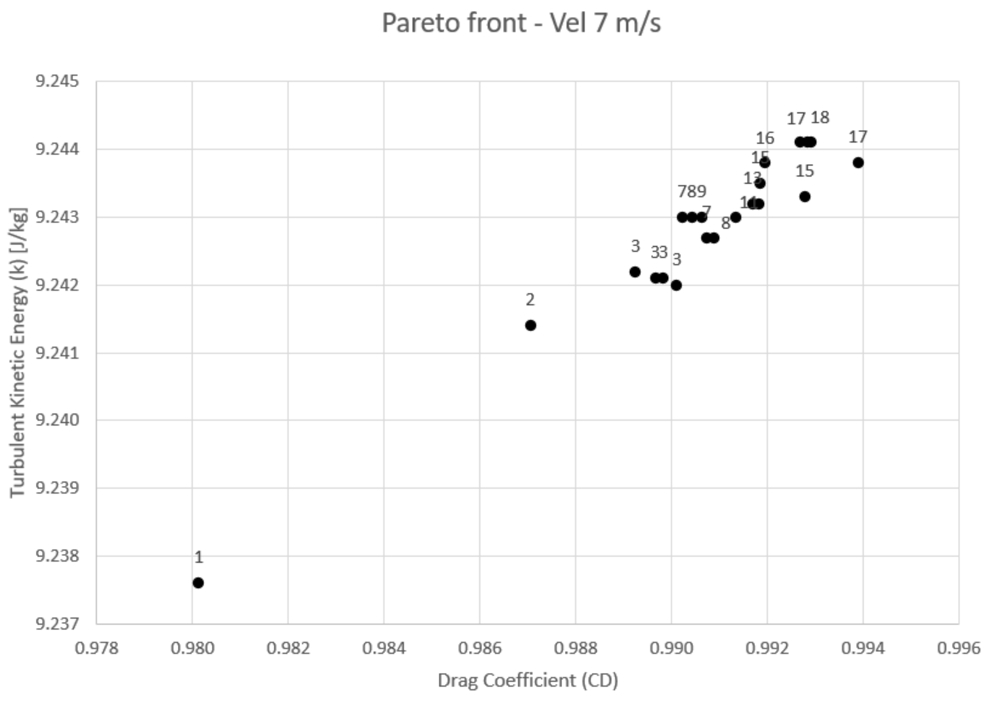


Figure H.3: Pareto efficiency - Optimal solution distribution plot for U_{in} equal to 7 m/s

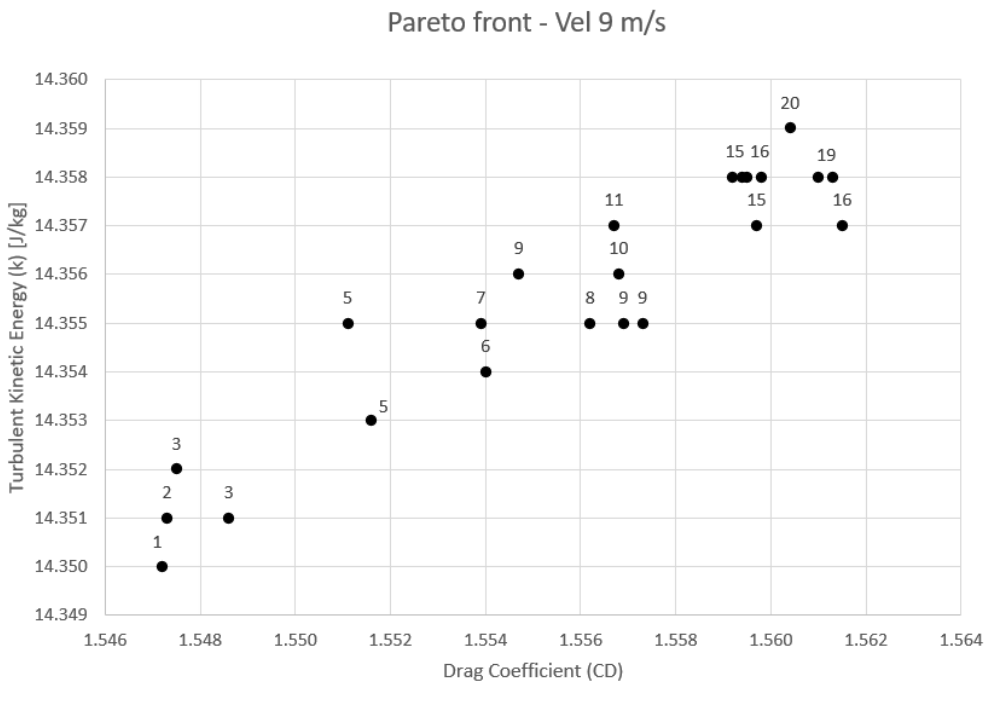


Figure H.4: Pareto efficiency - Optimal solution distribution plot for U_{in} equal to 9 m/s

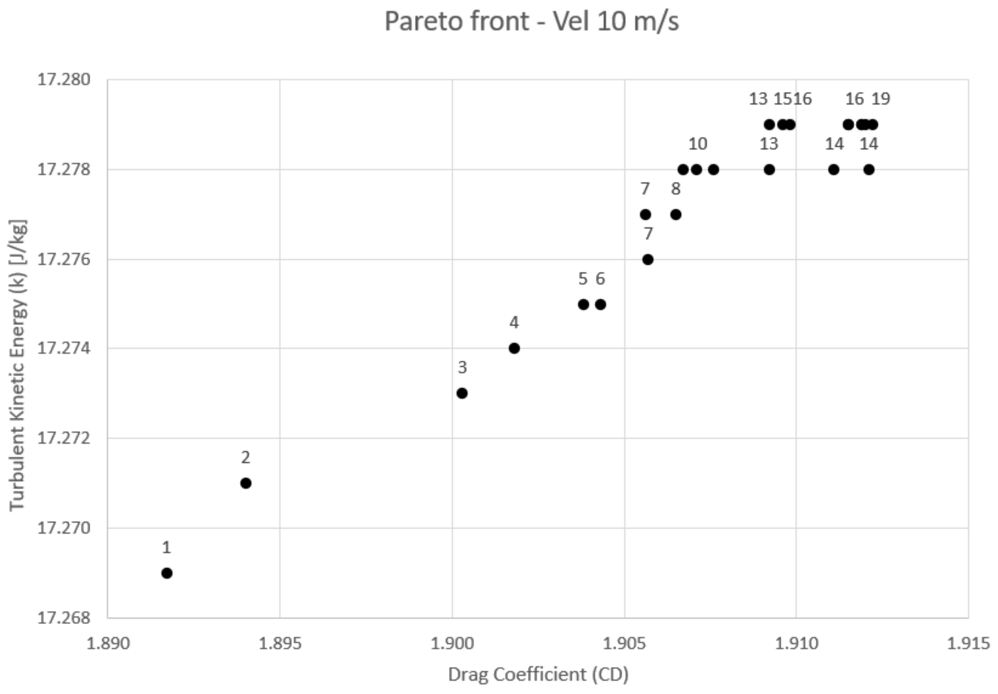


Figure H.5: Pareto efficiency - Optimal solution distribution plot for U_{in} equal to 10 m/s

Table H.4: Optimal values for the input geometrical variables for each Flow Velocity U_{in}

Velocity (m/s)	Solution	R_f	L_f	D	L	R_r	L_r	R_m
1	1	0.527661	0.373484	0.231481	1.080000	0.607500	0.354581	9.371613
2	1	0.498629	0.378000	0.231481	1.080000	0.607500	0.357161	8.268387
3	8	0.545806	0.366710	0.231481	1.080000	0.600242	0.354581	9.720000
4	5	0.524032	0.344129	0.231481	1.080000	0.603871	0.352000	9.720000
5	1	0.607500	0.378000	0.231481	1.080000	0.607500	0.354581	8.384516
6	1	0.534919	0.375742	0.232873	1.073548	0.603871	0.352000	9.720000
7	1	0.578468	0.373484	0.231481	1.080000	0.607500	0.352000	9.545806
8	2	0.538548	0.378000	0.232873	1.073548	0.607500	0.352000	9.661935
9	1	0.596613	0.373484	0.234281	1.067097	0.603871	0.352000	9.603871
10	1	0.603871	0.312516	0.231481	1.080000	0.603871	0.357161	9.661935

Table H.5: Optimal values for the output variables for each Flow Velocity (U_{in}) obtained by the Artificial Neural Network (ANN)

Velocity (m/s)	Solution	C_D	k
1	1	0.028379	0.303270
2	1	0.101740	1.024300
3	8	0.209270	2.088000
4	5	0.372520	3.468100
5	1	0.541510	5.125700
6	1	0.752290	7.056900
7	1	0.980130	9.237600
8	2	1.244700	11.654000
9	1	1.547200	14.350000
10	1	1.891700	17.269000

Table H.6: Optimal values for the output variables for each Flow Velocity (U_{in}) obtained by the CFD simulation in Ansys

Velocity (m/s)	Solution	C_D	k
1	1	0.030063	0.305364
2	1	0.103048	1.027315
3	8	0.218824	2.093908
4	5	0.378753	3.479759
5	1	0.569600	5.142752
6	1	0.799430	7.066403
7	1	1.013080	9.25186
8	2	1.285242	11.69200
9	1	1.610725	14.36151
10	1	1.891412	17.28877

Table H.7: Percentage difference between C_D and k obtained from ANN and Ansys for the chosen optimal solutions

Velocity (m/s)	Solution	Difference (%) C_D	Difference (%) k
1	1	5.600154	0.685723
2	1	1.269216	0.293445
3	8	4.365944	0.282147
4	5	1.645695	0.335063
5	1	4.931476	0.331581
6	1	5.896719	0.134479
7	1	3.252477	0.154132
8	2	3.154395	0.324992
9	1	3.943900	0.080110
10	1	0.015209	0.114323

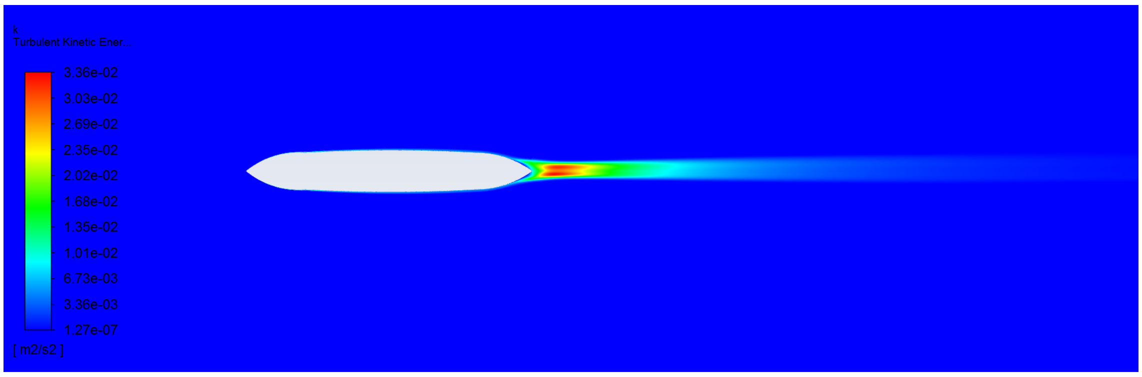


Figure H.6: Turbulent Kinetic Energy (k) contour - Full body (1 m/s)

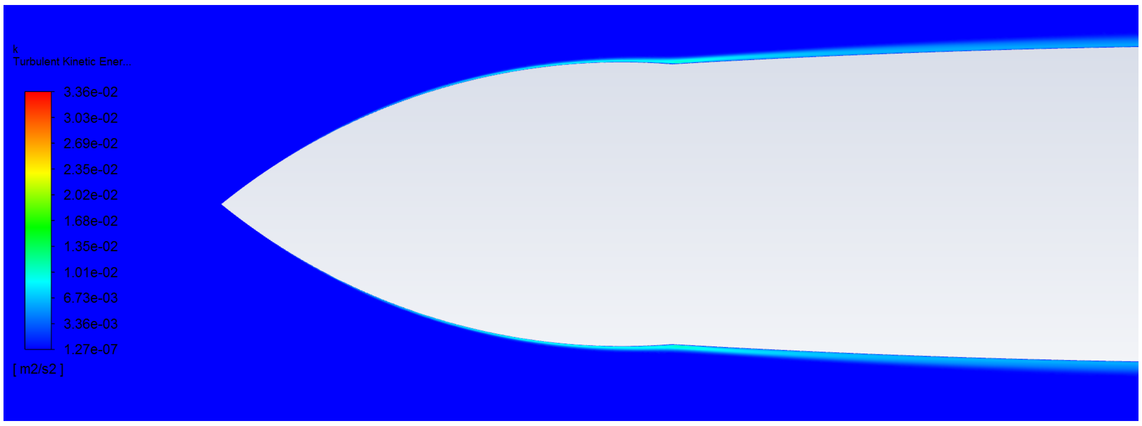


Figure H.7: Turbulent Kinetic Energy (k) contour - Front part of the body (1 m/s)

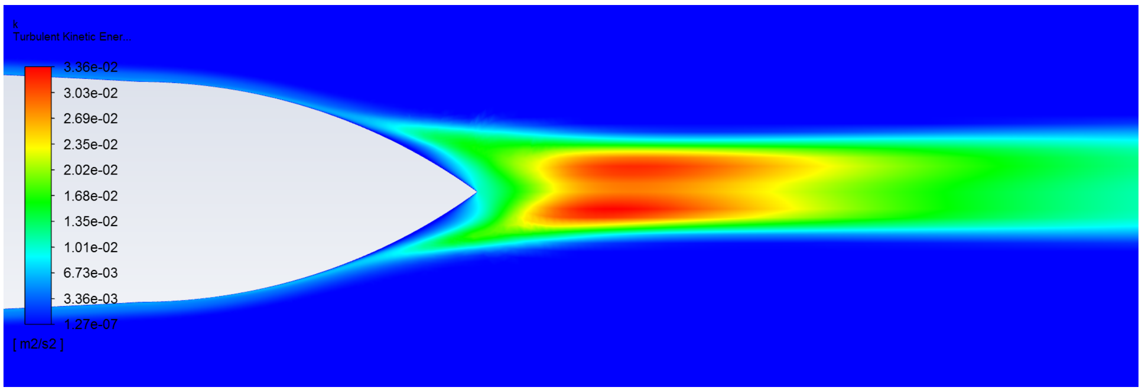


Figure H.8: Turbulent Kinetic Energy (k) contour - Rear part of the body (1 m/s)

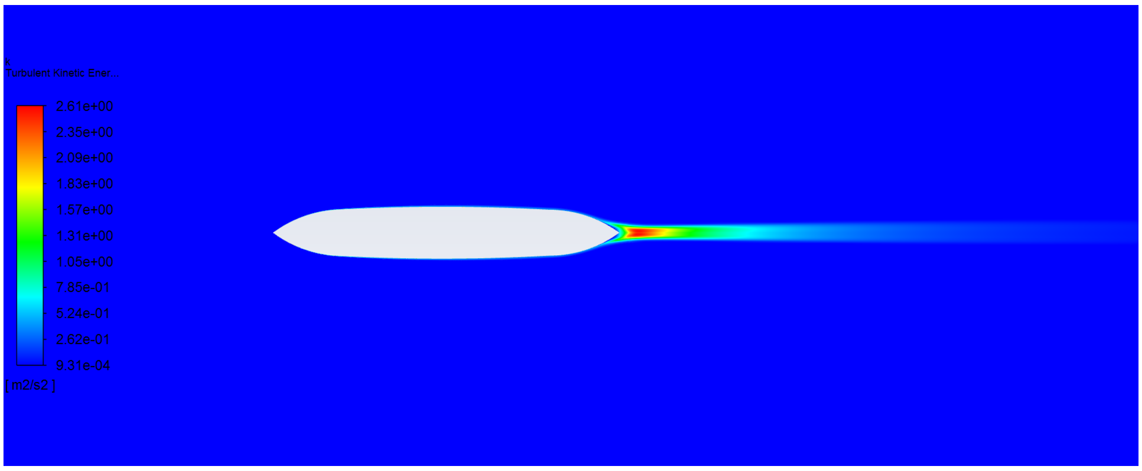


Figure H.9: Turbulent Kinetic Energy (k) contour - Full body (10 m/s)

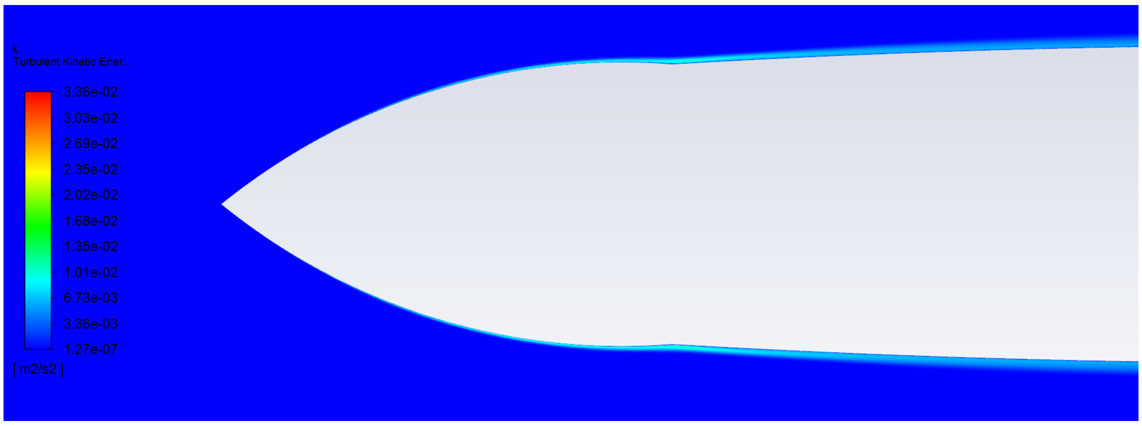


Figure H.10: Turbulent Kinetic Energy (k) contour - Front part of the body (10 m/s)

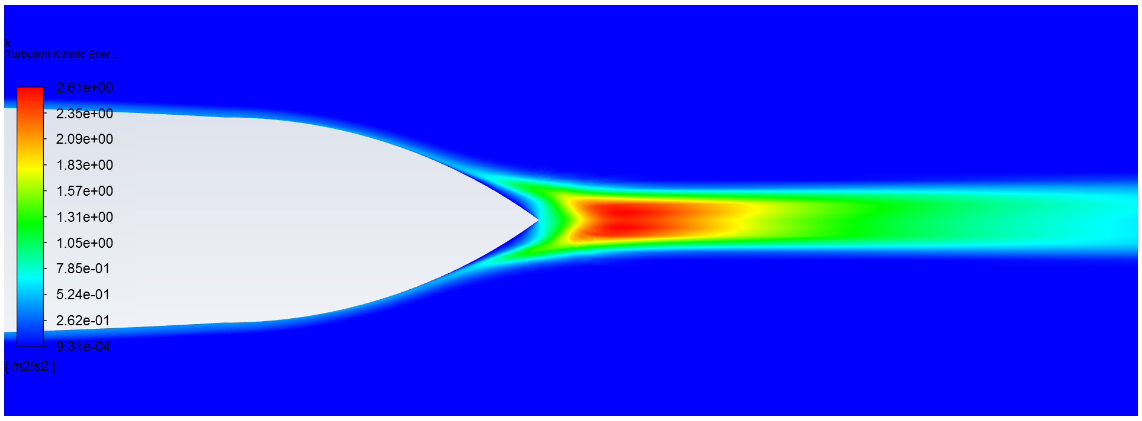


Figure H.11: Turbulent Kinetic Energy (k) contour - Rear part of the body (10 m/s)

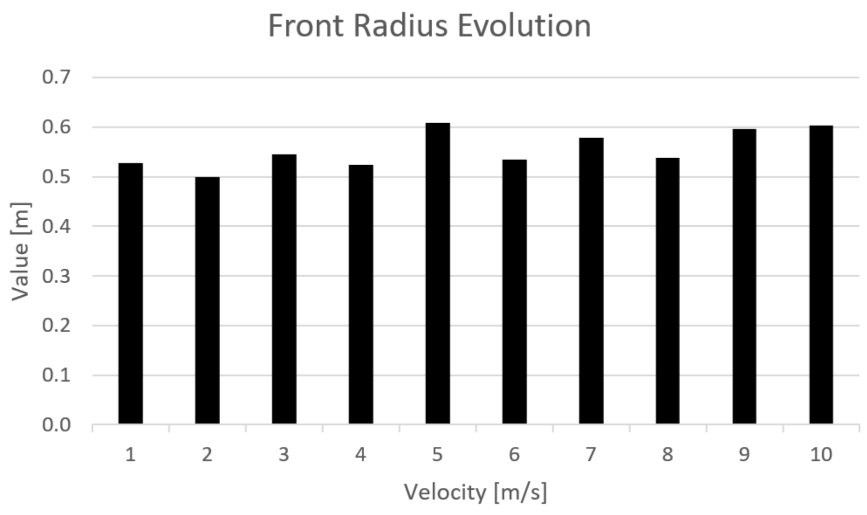


Figure H.12: Evolution of the variable value when the Flow Velocity U_{in} value changes: Front Radius (R_f)

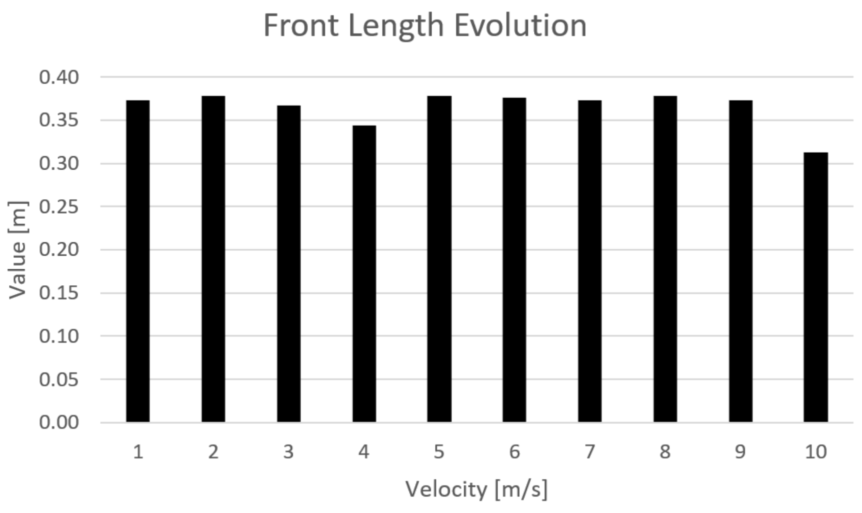


Figure H.13: Evolution of the variable value when the Flow Velocity U_{in} value changes: Front Length (L_f)

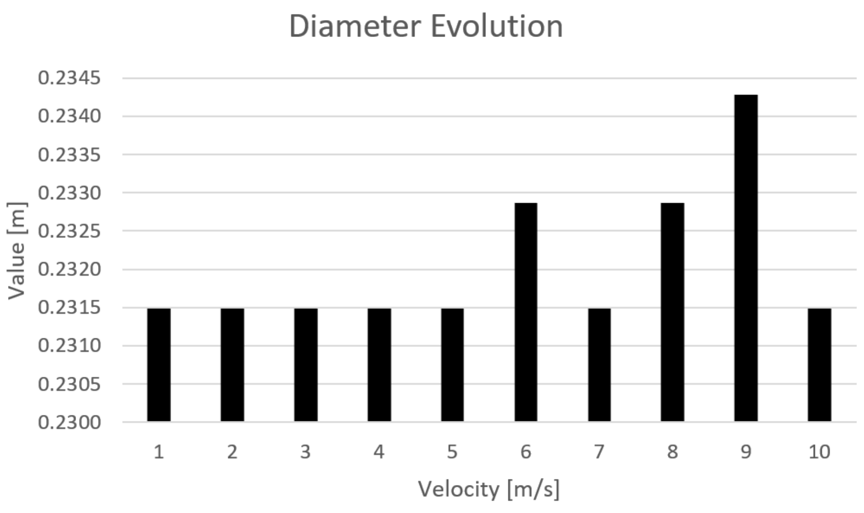


Figure H.14: Evolution of the variable value when the Flow Velocity U_{in} value changes: Diameter (D)

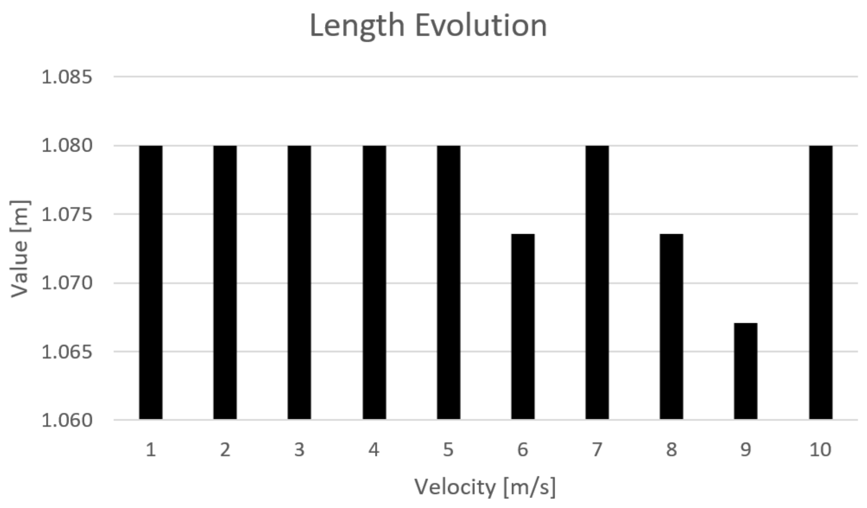


Figure H.15: Evolution of the variable value when the Flow Velocity U_{in} value changes: Length (L)

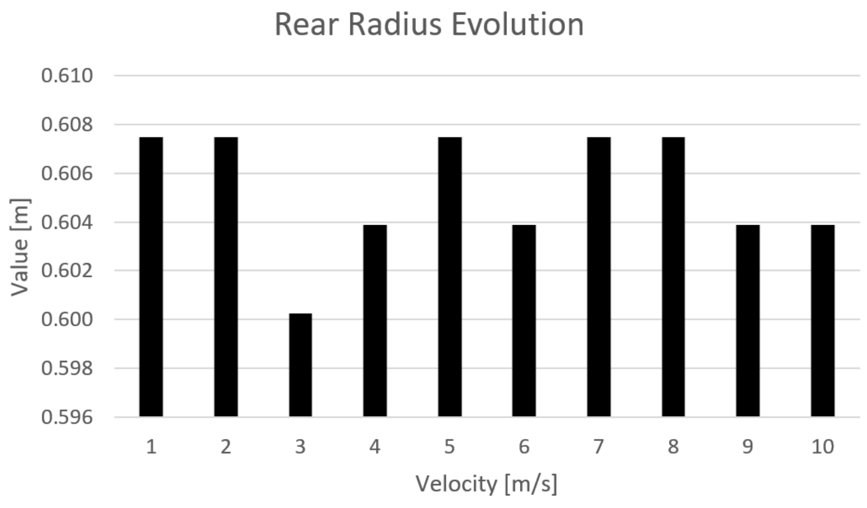


Figure H.16: Evolution of the variable value when the Flow Velocity U_{in} value changes: Rear Radius (R_r)

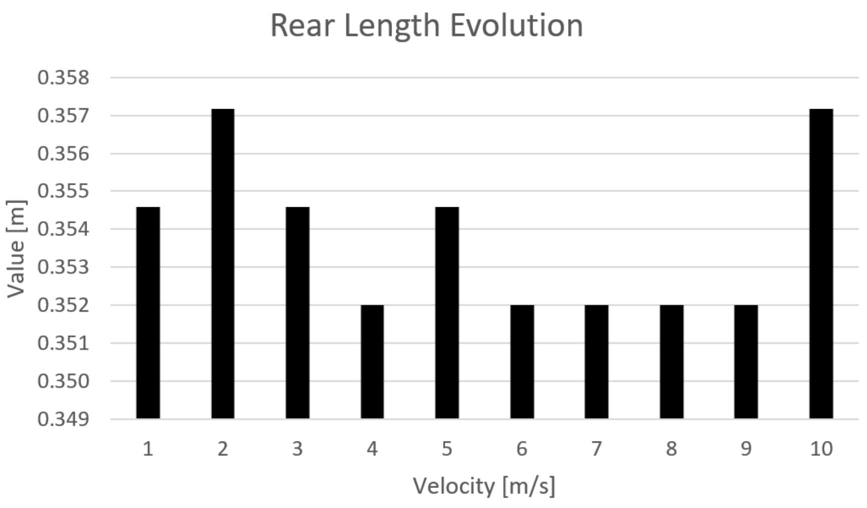


Figure H.17: Evolution of the variable value when the Flow Velocity U_{in} value changes: Rear Length (L_r)

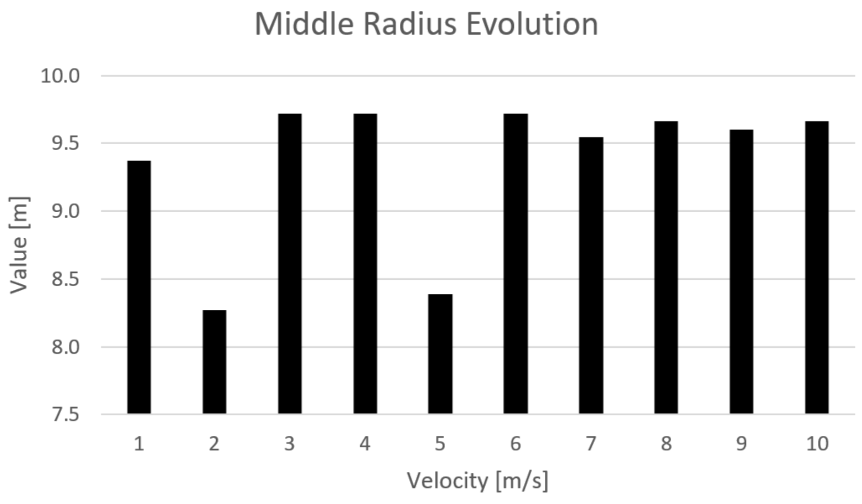


Figure H.18: Evolution of the variable value when the Flow Velocity U_{in} value changes: Middle Radius (R_m)

Bibliography

[1] A. Alvarez, A. Caiti, and R. Onken. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering*, 29 (2):418–429, 2004.

[2] ANSYS. Shear-Stress Transport (SST) k- ω Model. https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/node67.htm, 2009. [Online; accessed 9-March-2020].

[3] ANSYS. Using Flow Boundary Conditions. https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/node238.htm, 2009. [Online; accessed 11-March-2020].

[4] ANSYS. Discretization of the Momentum Equation. https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/node371.htm, 2009. [Online; accessed 12-March-2020].

[5] ANSYS. Evaluation of Gradients and Derivatives. https://www.afs.enea.it/project/neptunius/docs/fluent/html/th/node368.htm, 2009. [Online; accessed 12-March-2020].

[6] ANSYS. Choosing the Pressure-Velocity Coupling Method. https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/node785.htm, 2009. [Online; accessed 31-March-2020].

[7] ANSYS. Ansys Fluent. https://www.ansys.com/products/fluids/ansys-fluent, 2020. [Online; accessed 2-May-2020].

[8] ANSYS. Ansys Meshing Solutions. https://www.ansys.com/products/platform/ansys-meshing, 2020. [Online; accessed 2-May-2020].

[9] C. C. António and L. N. Hoffbauer. From local to global importance measures of uncertainty propagation in composite structures. *Composite Structures*, 85(3):213 – 225, 2008. ISSN 0263-8223. doi: https://doi.org/10.1016/j.compstruct.2007.10.012. URL http://www.sciencedirect.com/science/article/pii/S0263822307002553.

[10] I. Basheer and M. Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3 – 31, 2000. ISSN 0167-7012. doi: https://doi.org/10.1016/S0167-7012(00)00201-3. URL http://www.sciencedirect.com/science/article/pii/S0167701200002013. Neural Computting in Micrbiology.

[11] T. Benson. Reynolds Number. https://www.grc.nasa.gov/WWW/BGH/reynolds.html#, 2014. [Online; accessed 4-June-2020].

[12] M. Caudill. Neural networks primer, part i. *AI Expert*, 2(12):46–52, Dec. 1987. ISSN 0888-3785.

[13] J. Cheng, Q. Li, and R. cheng Xiao. A new artificial neural network-based response surface method for structural reliability analysis. *Probabilistic Engineering Mechanics*, 23(1):51 – 63, 2008. ISSN 0266-8920. doi: https://doi.org/10.1016/j.probengmech.2007.10.003. URL `http://www.sciencedirect.com/science/article/pii/S0266892007000422`.

[14] T. Chor. Turbulence: one of the great unsolved mysteries of physics. `https://www.youtube.com/watch?v=S3i6tJ4XNqA`, 2019. [Online; accessed 3-June-2020].

[15] V. Coles, R. Hood, and T. Gross. Hydrodynamic models. In S. E. Jørgensen and B. D. Fath, editors, *Encyclopedia of Ecology*, pages 1915 – 1923. Academic Press, Oxford, 2008. ISBN 978-0-08-045405-4. doi: https://doi.org/10.1016/B978-008045405-4.00894-6. URL `http://www.sciencedirect.com/science/article/pii/B9780080454054008946`.

[16] J. E. Colgate and K. M. Lynch. Mechanics and control of swimming: a review. *IEEE Journal of Oceanic Engineering*, 29(3):660–673, 2004.

[17] C. Conceição António and S. Rasheed. A displacement field approach based on fem-ann and experiments for identification of elastic properties of composites. *The International Journal of Advanced Manufacturing Technology*, 95, 04 2018. doi: 10.1007/s00170-017-1439-y.

[18] T. Crawford. Navier-Stokes Equations - Numberphile. `https://www.youtube.com/watch?v=ERBVFcutl3M&t=897s`, 2019. [Online; accessed 22-June-2020].

[19] L. DeVries, M. Kutzer, R. Richmond, and A. Bass. Design, modeling, and experimental drag characterization of a bio-inspired, shape-adapting underwater vehicle. page V05AT07A069, 08 2018. doi: 10.1115/DETC2018-85368.

[20] J. Droniou and R. Eymard. Study of the mixed finite volume method for stokes and navier-stokes equations. *Numerical Methods for Partial Differential Equations*, 25(1): 137–171, 2009. doi: 10.1002/num.20333.

[21] Eelume. Eelume Subsea intervention. `https://eelume.com/`, 2019. [Online; accessed 14-March-2020].

[22] Engineering and Technology. Realistic tuna robot created to improve next-gen underwater vehicles. `https://eandt.theiet.org/content/articles/2019/09/realistic-tuna-robot-created-to-improve-next-gen-underwater-vehicles/`, 2019. [Online; accessed 15-March-2020].

[23] K.-T. Fang, D. Lin, P. Winker, and Y. Zhang. Uniform design: Theory and application. *Technometrics*, 42:237–248, 08 2000. doi: 10.1080/00401706.2000.10486045.

[24] N. D. F. Gonçalves. Método dos volumes finitos em malhas não-estruturadas. 2007.

[25] S. Gudmundsson. Chapter 15 - aircraft drag analysis. In S. Gudmundsson, editor, *General Aviation Aircraft Design*, pages 661 – 760. Butterworth-Heinemann, Boston, 2014. ISBN 978-0-12-397308-5. doi: https://doi.org/10.1016/

B978-0-12-397308-5.00015-5. URL http://www.sciencedirect.com/science/article/pii/B9780123973085000155.

[26] J. Howard. *Fish Biology and Fisheries*. EDTECH, 2019. ISBN 9781839474484. URL https://books.google.pt/books?id=juTEDwAAQBAJ.

[27] M. Insight. Everything You Ever Wanted to Know About Autonomous Underwater Vehicle (AUV). https://www.marineinsight.com/types-of-ships/everything-you-ever-wanted-to-know-about-autonomous-underwater-vehicle-auv/, 2016. [Online; accessed 15-March-2020].

[28] C. M. Institute. Millenium Problems. https://www.claymath.org/millennium-problems, 2000. [Online; accessed 14-April-2020].

[29] A. J. *Governing Equations of Fluid Dynamics*. Springer, Berlin, Heidelberg, 2009. ISBN 978-3-540-85056-4. doi: https://doi.org/10.1007/978-3-540-85056-4_2.

[30] H. Jasak. Error analysis and estimation for the finite volume method with applications to fluid flows. *Direct*, M, 01 1996.

[31] G.-H. Y. Jiyuan Tu and C. Liu. *Computational Fluid Dynamics*. Butterworth-Heinemann, 2012. ISBN 9780080982779.

[32] J. B. Joshi, N. K. Nere, C. V. Rane, B. N. Murthy, C. S. Mathpati, A. W. Patwardhan, and V. V. Ranade. Cfd simulation of stirred tanks: Comparison of turbulence models. part i: Radial flow impellers. *The Canadian Journal of Chemical Engineering*, 89, 2011. doi: 10.1002/cjce.20446. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cjce.20446.

[33] J. R. M. Kenneth L. Wadlin, Charles L. Jr. Shuford. A theoretical and experimental investigation of the lift and drag characteristics of hydrofoils at subcritical and supercritical speeds. In *National Advisory Committee for Aeronautics*. NASA, National Advisory Committee for Aeronautics. Langley Aeronautical Lab.; Langley Field, VA, United States, 1995. URL https://ntrs.nasa.gov/search.jsp?R=19930092241.

[34] A. Kumar. Artificial neural network: In depth.

[35] G. V. Lauder and E. G. Drucker. Morphology and experimental hydrodynamics of fish fin control surfaces. *IEEE Journal of Oceanic Engineering*, 29(3):556–571, 2004.

[36] G. Li, Y. Deng, O. L. Osen, S. Bi, and H. Zhang. A bio-inspired swimming robot for marine aquaculture applications: From concept-design to simulation. In *OCEANS 2016 - Shanghai*, pages 1–7, 2016.

[37] R. Li, D. Lin, and Y. Chen. Uniform design: Design, analysis and applications. *International Journal of Materials Product Technology - INT J MATER PROD TECHNOL*, 20, 01 2004. doi: 10.1504/IJMPT.2004.003915.

[38] J. Lygouras, K. Lalakos, and P. Tsalides. Thetis: an underwater remotely operated vehicle for water pollution measurements. *Microprocessors and Microsystems*, 22(5):227 – 237, 1998. ISSN 0141-9331. doi: https://doi.org/10.1016/S0141-9331(98)00083-0. URL http://www.sciencedirect.com/science/article/pii/S0141933198000830.

[39] M. Mallick and A. Kumar. Study on drag coefficient for the flow past a cylinder. *International Journal of Civil Engineering Research*, 5:301–306, 01 2014.

[40] K. F. Man, K. S. Tang, and S. Kwong. Genetic algorithms: concepts and applications [in engineering design]. *IEEE Transactions on Industrial Electronics*, 43(5):519–534, 1996.

[41] D. S. Matthias Halwart and J. R. Arthur. Cage aquaculture: regional reviews and global overview. *Food and Agriculture Organization of the United Nations*, 2007.

[42] F. P. Meyer. Aquaculture disease and health management. *Journal of Animal Science*, 69(10):4201–4208, 10 1991. ISSN 0021-8812. doi: 10.2527/1991.69104201x. URL `https://doi.org/10.2527/1991.69104201x`.

[43] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM and Matlab*. Springer Publishing Company, Incorporated, 1st edition, 2015. ISBN 3319168738.

[44] A. J. Murphy and M. Haroutunian. Using bio-inspiration to improve capabilities of underwater vehicles. 2011.

[45] N. U. of Singapore. NUS-developed manta ray robot swims faster and operates up to 10 hours. `http://news.nus.edu.sg/press-releases/NUS-robotic-manta-ray`, 2017. [Online; accessed 15-March-2020].

[46] U. of Wisconsin-Madison. What Is MATLAB? `https://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm`, 2013. [Online; accessed 13-June-2020].

[47] C. Online. SST k-omega model. `https://www.cfd-online.com/Wiki/SST_k-omega_model`, 2011. [Online; accessed 4-March-2020].

[48] R. Paiva, C. Conceição António, and L. Silva. Optimal design of adhesive composition in footwear industry based on creep rate minimization. *The International Journal of Advanced Manufacturing Technology*, 84, 09 2015. doi: 10.1007/s00170-015-7746-2.

[49] B. Pikula, E. Mesic, and M. Hodzic. Determination of air drag coefficient of vehicle models. 10 2008.

[50] M. Šavli. Turbulence kinetic energy–tke. 2012.

[51] M. Sfakiotakis, D. M. Lane, and J. B. C. Davies. Review of fish swimming modes for aquatic locomotion. *IEEE Journal of Oceanic Engineering*, 24(2):237–252, 1999.

[52] W. Sheng. A revisit of navier-stokes equation. 12 2019.

[53] T. Tao. Finite time blowup for an averaged three-dimensional navier-stokes equation. *Journal of the American Mathematical Society*, 29, 02 2014. doi: 10.1090/jams/838.

[54] D. Valters. INTRODUCTION TO FORTRAN, THE BASICS OF THE FORTRAN PROGRAMMING LANGUAGE. `https://ourcodingclub.github.io/tutorials/fortran-intro/`, 2020. [Online; accessed 08-September-2020].

[55] R. Wernli. The present and future capabilities of deep rovs. *Marine Technology Society Journal*, 33:26–40, 01 1999. doi: 10.4031/MTSJ.33.4.4.

[56] R. B. Wynn, V. A. Huvenne, T. P. Le Bas, B. J. Murton, D. P. Connelly, B. J. Bett, H. A. Ruhl, K. J. Morris, J. Peakall, D. R. Parsons, E. J. Sumner, S. E. Darby, R. M. Dorrell, and J. E. Hunt. Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience. *Marine Geology*, 352:451 – 468, 2014. ISSN 0025-3227. doi: https://doi.org/10.1016/j.margeo.2014.03.012. URL `http://www.sciencedirect.com/science/article/pii/S0025322714000747`. 50th Anniversary Special Issue.

Faculty of Engineering of University of Porto

Morphing Autonomous Underwater Vehicle - Hydrodynamic Analysis
Dissertation
September 2020