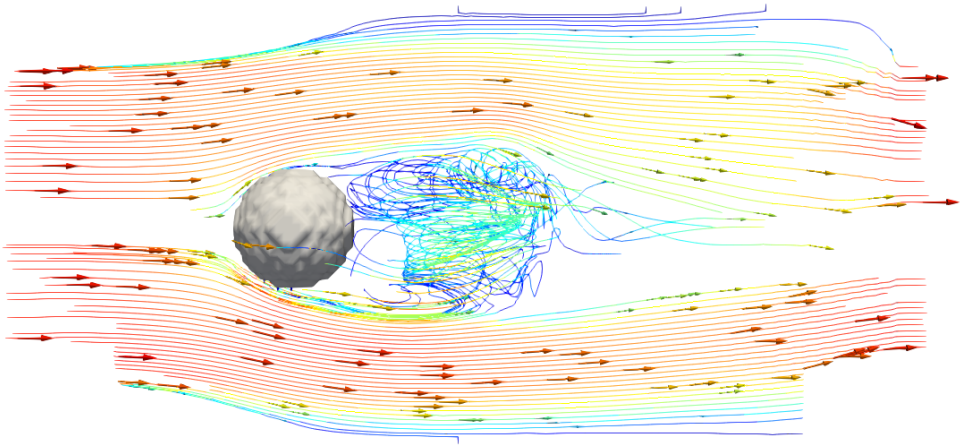
TÉCNICO LISBOA



Towards Real-Time Hydrodynamics and Fluid Simulation for Marine Robotics

Yosef Shmuel Guevara Salamanca

Thesis to obtain the Master of Science Degree in
Electrical and Computer Engineering

Supervisors:
Prof. Vincent Hugel
Prof. Rodrigo Martins de Matos Ventura

Examination Committee
Chairperson: Prof. João Luís Da Costa Campos Gonçalves
Supervisor: Prof. Rodrigo Martins de Matos Ventura
Member of the Committee: Dr. Fábio Cruz

November 2024

Statement

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisors, who, through their dedication and support, have been a fundamental pillar in this work. Despite the bureaucratic obstacles faced, their commitment and encouragement made it possible for me to advance at each stage.

I am deeply grateful to Professor Vincent Hugel for welcoming me to the COSMER laboratory at the University of Toulon during a difficult time and for guiding me through such a complex topic as fluid mechanics. Likewise, to Professor Rodrigo Ventura, for being with me from the beginning of this project, for his willingness to offer me the IST infrastructure, and for the valuable practical feedback he provided on my thesis. I also thank my master's classmates and the Erasmus Mundus MIR program for granting me this wonderful opportunity.

I would also like to thank my labmates and friends, Bilal Ghader and Christian Lasso, for their support and assistance during my time at COSMER. Without their company and collaboration, the journey would have been much more challenging. I hope that one day our paths will cross again in person.

To my beloved Luz Stefany, thank you for your guidance, trust, and unconditional love. You have been my guide and greatest support in every challenge of this project. Finally, I thank my family, who has always been there for me. I hope this achievement serves as an inspiration for their own future projects and challenges.

Abstract

This thesis explores the implementation of Physics-Informed Neural Networks (PINNs) to simulate hydrodynamic behaviors of fully submerged objects in Newtonian incompressible isothermal Fluids in confined environments that mimic a wind tunnel scenarios. The primary objective is to test and validate a (PINN)-based methodology capable of replicating real-time fluid simulations, with a focus on simulating drag force, lift force, vorticity, fluid flow, and pressure distributions. By comparing the results with traditional Computational Fluid Dynamics (CFD) simulations, this research aims to evaluate the effectiveness of (PINN) in both laminar and turbulent flow regimes. This research introduces a qualitative and quantitative analysis of hydrodynamic forces and phenomena using simplified geometric objects and typical underwater robotics shapes, checking generalization across different fluid conditions. A specific dataset was designed to train and validate the (PINN) architectures, with the results showing qualitative concordance with (CFD) baselines in certain scenarios, while identifying limitations in low Reynolds numbers flows and boundary condition handling. Additionally, the potential scalability of this (PINN) methodology makes it suitable for integration into marine robotics simulators, providing real-time hydrodynamic phenomena calculation with manageable computational requirements.

Key Words: Physics-Informed Neural Networks (PINNs), Computational Fluid Dynamics (CFD), real-time fluid simulation, hydrodynamic forces, drag and lift, vorticity, marine robotics, incompressible fluids, isothermal fluid dynamics, Pruned-UNet architecture, U-Net architecture, fluid-structure interaction, boundary conditions, turbulent flows, laminar flows.

Resumo

Esta tese explora a implementação de Redes Neurais Informadas por Física (PINNs) para simular comportamentos hidrodinâmicos de objetos totalmente submersos em fluidos newtonianos incompressíveis e isotérmicos em ambientes confinados que imitam cenários de túnel de vento. O objetivo principal é testar e validar uma metodologia baseada em PINN capaz de replicar simulações de fluido em tempo real, com foco na simulação de força de arrasto, força de sustentação, vorticidade, fluxo de fluido e distribuições de pressão. Ao comparar os resultados com simulações tradicionais de Dinâmica dos Fluidos Computacional (CFD), esta pesquisa visa avaliar a eficácia das PINNs em regimes de fluxo laminar e turbulento. A pesquisa introduz uma análise qualitativa e quantitativa das forças e fenômenos hidrodinâmicos utilizando objetos geométricos simplificados e formas típicas de robótica subaquática, verificando a generalização em diferentes condições de fluido. Um conjunto de dados específico foi criado para treinar e validar as arquiteturas PINN, com os resultados mostrando concordância qualitativa com as referências de CFD em certos cenários, ao mesmo tempo em que identifica limitações em fluxos de baixo número de Reynolds e no tratamento de condições de fronteira. Além disso, a potencial escalabilidade desta metodologia baseada em PINN a torna adequada para integração em simuladores de robótica marinha, proporcionando cálculo de fenômenos hidrodinâmicos em tempo real com requisitos computacionais razoáveis.

Palavras-chavem: Redes Neurais Informadas por Física (PINNs), Dinâmica de Fluidos Computacional (CFD), simulação de fluidos em tempo real, forças hidrodinâmicas, arrasto e sustentação, vorticidade, robótica marinha, fluidos incompressíveis, dinâmica de fluidos isotérmicos, arquitetura Pruned-UNet, arquitetura U-Net, interação fluido-estrutura, condições de contorno, fluxos turbulentos, fluxos laminares.

Contents

List of Figures

List of Tables

Acronyms

AUV Autonomous Underwater Vehicle
AI Artificial Intelligence
CNN Convolutional Neural Networks
CFD Computational Fluid Dynamics
CPU Central Processing Unit
DL Deep Learning
DFT Discrete Fourier Transform
DNS Direct Numerical Solution
DOF Degrees of Freedom
FFT Fast Fourier Transform
FLIP Fluid-Implicit-Particle
FPS Frames Per Second
GAN Generative Adversarial Networks
GNN Graph Neural Networks
GPU Graphics Processing Unit
GCNN Generative Convolutional Neural Networks
ISPH Incompressible Smoothed Particle Hydrodynamics
LES Large Eddy Simulation
LSTM Long Short Term Memory
Ma Mach Number
NN Neural Networks
RAS Reynolds Averaged Simulation
Re Reynolds number
Re$_{cr}$ Critical Reynolds number
Re$_L$ Reynolds Number for external Flows
SI International System of Units
SIMPLE Semi-Implicit Method for Pressure Linked Equations
SPH Smoothed Particle Hydrodynamics
STL Standard Triangle Language
UUV Unmanned Underwater Vehicle
UV Underwater Vehicle
ODE Ordinary Differential Equations
PDE Partial Differential Equations
PID Proportional Integral Derivative Controllers
PINN Physics Informed Neural Networks
PMM Planar Motion Mechanism
voxel Volumetric tridimentional unit

Nomenclature

Physical Constants

ε	Minimum number of molecules for continuity. (2.7×10^6)
c	Speed of sound. $(346\ m/s)$
C_{mu}	Relationship between turbulent kinetic energy and specific dispoation rate. (0.09)
g	Gravitational acceleration. $(9.8\ m/s^2)$

Physical Variables

δ	Fluid domain dimension. (m)
λ	Distance between molecules in a fluid. (m)
ϱ	Density. (kg/m^3)
$\bar{U}$	Average flow of speed. (m/s)
$\dot{m}$	Net mass flow. (kg)
ϵ	Turbulent dissipation rate. (W/m^3)
$\mathbf{k}$	Turbulent kinetic energy. (m^2/s^2)
μ	Dynamic viscosity. (Pa s)
ν	Kinematic viscosity. (m^2/s)
ω	Specific turbulence disipation rate. $(1/frecuency)$
ω_a	Angular velocity. (rad/s)
σ	Normal stress. (MPa)
τ	Shear stress. (N/m^2)
$\tau_{\mathbf{w}}$	Frictional wind stress. (N/m^2)
τ_r	Yaw torque. (Nm)
ϱ_a	Density of air. (kg/m^3)
$\vec{\omega}$	Angular velocity vector. (rad/s)
$\vec{\zeta}$	Vorticity vector. $(1/s)$
$\vec{a}$	Acceleration vector. (m/s^2)
$\vec{F}$	Vector force. (N)
$\vec{v}$	Velocity vector. (m/s)
$C(v)$	Matrix of Coriolis Centripetal term (including added mass).
F	Force. (N)
F_b	Buoyant force. (N)
F_c	Force applied to object. (N)
F_g	Gravitational force. (N)
F_s	Shear force. (N)
F_x	Thrust force. (N)
F_y	Lateral force. (N)
$F_{friction}$	Force due to the friction. (N)
$F_{pressure}$	Force due to the pressure. (N)
K	Bulk modulus of elasticity. (GPa)
M	Inertia Matrix (including added mass).
m	Mass. (kg)
$m_{element}$	Fluid element mass. (kg)
p	Pressure. (Pa)
P_m	Mechanical pressure. (Pa)
p_w	Pressure stress. (N)
Q	Volume flow rate. (m^3/s)
T	Temperature.
U	Flow speed. (m/s)
V	Volume. (m^3)
v_f	Volume of displaced water. (m^3)
U'	Flow field of fluctuations. (m/s)

Other Symbols

α_{pitch}	Vehicle proportional pitch angle.

α_{roll} Vehicle proportional roll angle.

$\eta(r,t)$ Infinite sum of waves.

$\eta_n(t)$ Wave surface displacement in Fourier space.

$\hat{n}$ Unit normal.

$\ln(\mu^{t+\Delta t})$ Logarithm of the dynamic viscosity at $t+\Delta t$.

$\ln(\rho^{t+\Delta t})$ Logarithm of the density at $t+\Delta t$.

$\mathbf{P}_m$ Vehicle directional vector.

$\mathbf{P}_w$ Wind directional vector.

$\mathbf{r}$ Displacement vector

$\mathcal{L}$ Compound loss ($L_p + L_b$).

$\mathcal{S}$ Surface of an object.

μ_T Eddy viscosity.

$\nabla \times \vec{a}^t$ Curl of the vector potential at time t.

Ω Fluid domain.

$\Omega^{t+\Delta t} \cdot \nabla \times \vec{a}^t$ Dot product of the domain and the curl at $t+\Delta t$.

$\Omega^{t+\Delta t}\dot{p}^t$ Fluid domain at $t+\Delta t$ multiplied by the time derivative of pressure at t.

$\Omega^{t+\Delta t}$ Fluid domain at $t+\Delta t$.

ω_d Wave dispersion relation.

$\partial\Omega^{t+\Delta t}$ Boundary of the fluid domain at $t+\Delta t$.

ϕ_{pitch} Vehicle pitch angle.

ϕ_{roll} Vehicle roll angle.

σ_c Current induced disturbances (constant).

τ' Reynolds stress.

τ Vector of control inputs.

τ_w Viscous stress vector or Friction Drag

τ_{wave} Vector of wave loads.

τ_{wind} Vector of wind loads.

θ Wave direction.

$\vec{a}^t$ Vector potential at time t.

$\vec{n}$ Unit vector.

$\vec{v}_d^{t+\Delta t}$ Velocity at the boundary at $t+\Delta t$.

A Area. (m^3)

C_D Drag cofficient.

C_d Drag coefficient.

$D(\omega_d, \theta)$ Directional spatial distribution.

$D(v)$ Damping matrix.

D Diameter. (m or voxels)

e_{ij} Strain rate tensor.

F_D Drag Force.

F_L Lift Force.

F_Y Side force due to the pressure.

$g(\eta)$ Vector of gravitational/buoyancy forces and moments.

g_o Vector used for pretrimming (ballast control).

$h(v_w, t)$ Water-wind-vehicle.

h Water depth.

$h_m(t)$ Fourier coefficient.

$h_m(v)$ Maximum wave amplitude.

k Angular wavenumber.

L Characteristic length. (m or voxels)

L_b Boundary loss.

L_p Momentum loss.

N Number of spatial points.

$P(k)$ Spatial spectrum.

p^t Pressure at time t.

P_L Lagrangian particle in Lagrangian description.

P_s Specific point in Eulerian description.

r Wave vector.

$S(\omega_d)$ Non-directional spatial spectrum.

v_g Group phase velocity.

v_w Wind amplitude.

CHAPTER 1

Introduction

Contents

1.1 Motivation

In engineering and scientific research, the importance of simulations and simulators is clear [1]. These virtual tools emulate real-world scenarios in a computational environment, allowing cyber-physical systems such vehicles or robots to be digitally replicated. Simulating their characteristics and functionalities, which include control theory algorithms, sensor interactions and hardware behavior. Mimicking complex environments where these digital twins will operate. This process is performed under the establishment of physical laws, rules or among other constrains established for the simulation, considering the interaction of these digital-twins with their surrounding environment, which minimizes the risks associated with experimentation in real conditions.

In robotics, simulators mostly focus on land or aerial robotics. However, in the field of marine and maritime robotics, there are just a few of them [2]. Frequently they are unable to mimic real-time physics based fluid mechanics or fluid-object interaction, which are critical aspects in areas such as control algorithms, navigation, autonomous systems and artificial intelligence. Understanding the interaction of marine vehicles with water is a major aspect of marine robotics engineering, as the multiple effects of forces and moments on these vehicles are fundamental to their design [3] and behavior. As an example, the study of how water interacts with the hull can drive the development of more efficient marine vehicles, in terms of mobility, energy consumption, cost-effectiveness, and adaptability to different weather conditions [4]. However, commonly roboticists only want to focus on control, computer vision or artificial intelligence algorithms rather than hydrodynamics.

Therefore, achieving an accurate fluid mechanics without proper fluid simulation is challenging [5]. Often, oversimplified using unrealistic physical assumptions due to the high computational cost of full 3D fluid simulation [6, 7]. This kind of simulation frequently faces challenges in proper integration with visualization tools, as well as model validation and verification [8, 6]. A scenario where hydrodynamics and fluid interactions are implicit in the environment are necessary to replicate situations beyond the ideal ones, where environmental conditions are neglected. Virtual scenarios where the different factors that can affect the behavior of a marine vehicle are necessary to develop more robust systems that allow eventual autonomy in complex situations.

In a real-time simulation, it is essential to stay within the established processing time limits of 15 to 60 frames per second (FPS) [9]. This requirement contributes to a central role in applications such as control theory, digital twins, and even video games. This implies that current fluid simulation methods used in offline simulations such CFD are not directly transferable to real-time applications, real-time fluid simulations need to be capable of handling a wide range of agent and user interactions with predefined or variable fluid conditions. However, there is a distinction between simulators focused on visual representation, such as game engines, which prioritize graphical fidelity, and those where physical accuracy is required [6].

This thesis addresses the current limitations of simulators and simulations in reproducing hydrodynamic phenomena and fluid-object interactions in real-time. To the best of the author's knowledge, and based on the state of the art at the time of writing (Nov. 2024), it is one of the first to implement artificial intelligence, specifically physics-informed neural networks (PINN), for modeling complex hydrodynamic phenomena such as flow separation, boundary layers, and vortices in real-time. These phenomena are reproduced by the implicit fluid-object interactions, rather than being directly programmed. Furthermore, the work validates these simulations through measurements, comparisons, and evaluations against reliable CFD (Computational Fluid Dynamics) results using Reynolds-averaged Navier-Stokes (RANS) equations under various fluid conditions and geometric configurations typical of marine robotics.

1.2 Goals and Research Questions

The main goal of this thesis is to evaluate and compare data-driven fluid simulations for fully submerged objects in a Newtonian incompressible isothermal fluids in confined environments, which mimics a wind tunnel scenarios, using the Physics-Informed Neural Networks (PINN) methodology proposed by Wandel et al. (2021) for real-time 3D fluid simulations based on an Eulerian description [10]. Additionally, this thesis explores the limitations of hardware and necessary simplifications in simulations due to boundary conditions and fluid domain size, while also investigating the stability and accuracy of real-time simulations versus offline CFD simulations. Furthermore, the thesis aims to analyze the hydrodynamic variables, such as drag force, lift force, pressure distribution, velocity fields, and vorticity, to ensure accurate fluid-structure interaction and flow behavior in various conditions.

The main research questions of the thesis can be summarized as follows:

1. Is it a PINN-based methodology capable to simulate fluid dynamics and hydrodynamic variables (drag force, lift force, pressure distribution, velocity, and vorticity) for fully submerged objects in real-time?
2. How does the accuracy and stability of real-time PINN simulations in capturing hydrodynamic behaviors compare to traditional offline CFD methods?
3. How scalable is the PINN approach in handling complex geometries, larger fluid domains, and different flow regimes (laminar, turbulent)?
4. What are the limitations and necessary simplifications when using PINN for fluid simulations, particularly regarding hardware, boundary conditions, fluid domain size, and the representation of hydrodynamic variables?
5. What potential does the PINN methodology hold for real-time integration into marine robotics simulators, especially in terms of generalization, hydrodynamic accuracy, and reliable fluid simulation across diverse operational conditions?

1.3 Main Contributions

Based on the research questions, this thesis aims to implement a fluid PINN-based simulation methodology capable of replicating real-time measurable hydrodynamic phenomena, by presenting the following contributions:

- A training dataset with shape simplifications has been developed to validate PINN models against CFD baselines, ensuring generalization across different domains.
- A qualitative and quantitative analysis will be conducted on the hydrodynamic phenomena such pressure, velocity field behavior and vorticity in fully submerged objects, addressing laminar and turbulent flow regimes, for a PINN-based methodology.
- A quantitative analysis will be conducted on basic hydrodynamic variables such as drag force, lift force and vorticity, to evaluate fluid-surface interactions in multiple fluid conditions.
- A comparative evaluation of PINN architectures and traditional CFD methods will be performed, evaluating computational efficiency and accuracy, with a focus on real-time simulation performance.
- The potential scalability of PINN models for integration in marine robotics simulators will be evaluated, enabling real-time fluid simulations with lower computational costs.

1.4 Thesis Outline

This thesis is divided into eight chapters, covering the main aspects of this research.

Chapter 2 - Notions on Fluid Mechanics: Introduces fundamental fluid mechanics concepts, such as density, pressure, and viscosity, along with key laws like Bernoulli's and Navier-Stokes equations for analyzing laminar and turbulent flows.

Chapter 3 - Literature Review: Fluid Simulation Approaches and Models: Reviews existing fluid simulation approaches, including ocean surface simulations, underwater modeling, and data-driven techniques like Physics-Informed Neural Networks (PINNs) to solve complex fluid equations. Different descriptions of fluid motion (Lagrangian and Eulerian) are also discussed.

Chapter 4 - Methodology Part I: PINN Model: Covers the construction of a PINN-based model, including the neural network architectures, dataset preparation, and training process.

Chapter 5 - Methodology Part II: CFD Baseline: Discusses the creation of a reference baseline using traditional computational fluid dynamics (CFD) methods, detailing software, mesh construction, and turbulence models.

Chapter 6 - Methodology Part III: Fluid Conditions and Hydrodynamic Variables: Describes the methods and procedures for calculating hydrodynamic variables like velocity, pressure, drag, lift, and vorticity to compare simulation results.

Chapter 7 - Results: Simulation Analysis and Comparison: Provides a detailed analysis of simulation results and hydrodynamic variables using different (PINN) architectures compared to the CFD baseline.

Chapter 8 - Conclusion and Further Work: Summarizes findings and discusses future research directions.

CHAPTER 2

Notions on Fluid Mechanics

Contents

Fluid mechanics is the branch of physics that studies the behavior of fluids **(liquids and gases)** both at rest **(static)** and in motion **(dynamic)**, analyzing the forces and interactions that occur within them. In the field of marine robotics, phenomena such as buoyancy, pressure, drag, lift, and vortex generation are of particular importance. These phenomena result from the internal dynamics of the fluid and its interaction with objects, being influenced by the pressure and relative velocity between the object and the fluid.

To describe these phenomena, fundamental principles such as Pascal's and Archimedes' principles are applied in static situations, while the conservation of mass, conservation of momentum, continuity equation, and Bernoulli's principle are used in dynamic cases. Along with their simplifications and special cases are employed to model approximations of these behaviors.

This chapter aims to familiarize the reader with the fundamental concepts of fluid mechanics by providing a comprehensive overview of the basic aspects governing fluid behavior, with a particular emphasis on Newtonian incompressible isothermal fluids. The first section (2.1) covers the fundamentals of fluid properties such as pressure, density, viscosity, among others. The second section (2.2) addresses key concepts in hydrostatics, including Pascal's and Archimedes' principles, and the fundamental hydrostatic equation. The third section (2.3) discusses fluid classification in motion based on Reynolds numbers and whether the fluid is compressible or incompressible. Finally, the fourth and last section (2.4) focuses on the differential modeling of incompressible-isothermal fluids, focusing on the continuity equation, the Navier-Stokes momentum equation. Concepts that will be later used during the work development, as we will explain during the following chapters.

2.1 Fluid Properties

A fluid is a substance, such as a liquid or gas, which continuously deforms under the action of a force parallel to its surface, forcing their internal layers to slide past each other, as a consequence, a fluid flows and adapts to the shape of its container due to the free movement (distance) of its molecules [11]. Under small fluctuations in heat and density, fluids exhibit the following properties [12].

2.1.1 Continuity

A fluid is considered continuous if the distance λ between the molecules is insignificant compared to the physical dimensions of the problem δ, a characteristic known as (free path). Therefore, macroscopic fluid parameters such as density, pressure, velocity, and temperature are assumed to be continuous and to vary smoothly over space. This continuum model is applicable to the majority of fluid flow conditions, as the local changes resulting from individual molecular motion are negligible, and all fluids share the following physical characteristics [13].

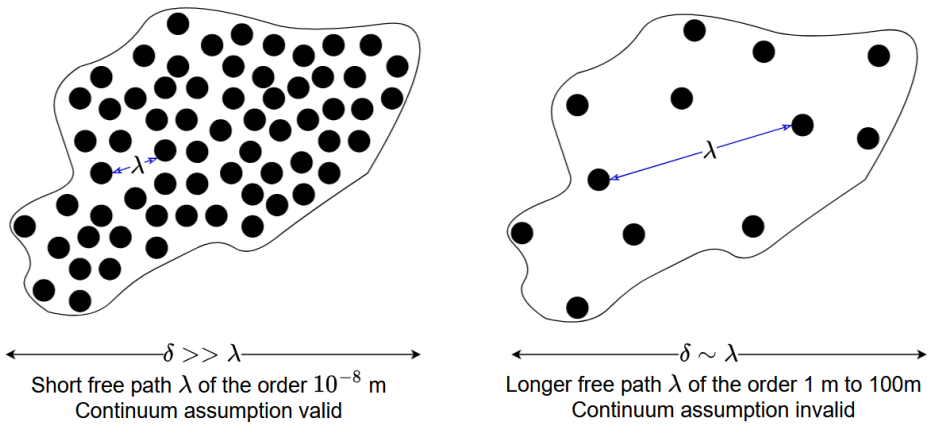


Figure 2.1: The continuum model assumption. Adapted from: [13].

2.1.2 Density

A fluid is defined as continuous over a region of interest, and this continuity is assumed based on its density, which is given by:

$$\varrho = \lim_{\Delta v \to 0} \frac{\Delta m}{\Delta V} \tag{1}$$

where, Δm in kilograms (kg), is the incremental mass variation contained in an incremental volume ΔV in cubic meters (m^3). However, as $\Delta V \to 0$, continuity is lost because the mass m varies discontinuously depending on the number of molecules. Typically, volumes smaller than $\varepsilon = 2.7 \times 10^6$ molecules (contained in a cubic millimeter of air under standard conditions 101.3 kPa and a temperature of 15°C are not considered. For general purposes, the nominal density of seawater is 1027 kg/m^3 [12].

2.1.3 Temperature

The temperature affects the fluid behavior and properties. It influences the Kinetic energy of a molecule, determining the degree of heat or cold of an object, which alters viscosity, density, and flow characteristics. As the temperature of a liquid increases, the kinetic energy of its molecules increases, causing them to move faster and overcome the intermolecular attractive forces that hold them together.

- **Viscosity Reduction:** Higher temperatures decrease liquid viscosity by increasing molecular kinetic energy, which overcomes intermolecular forces.
- **Density Decrease:** As the temperature rises, liquids expand slightly, reducing their density as the average distance between molecules increases.
- **Vapor Pressure Increase:** Vapor pressure rises exponentially with temperature. When vapor pressure equals atmospheric pressure, boiling occurs, and if local pressure drops below vapor pressure, cavitation may occur, potentially damaging surfaces.
- **Solubility Changes:** Solubility of solids in liquids generally increases with temperature, while gas solubility decreases.

The temperature is commonly expressed in scales such as Celsius (°C), Fahrenheit (°F), and Kelvin (K), Kelvin being the base unit for temperature and is defined based on absolute zero, the lowest possible temperature in the International System of Units (SI). Absolute zero is set at 0 K, where the thermal motion of atoms and molecules is minimized. Where the relation between the different temperature scales is given by:

$$K = \,^\circ\mathrm{C} + 273.15 \tag{2}$$

$$^\circ\mathrm{F} = \frac{9}{5}\,^\circ\mathrm{C} + 32 \tag{3}$$

$$^\circ\mathrm{C} = \frac{5}{9}(^\circ\mathrm{F} - 32) \tag{4}$$

2.1.4 Compressibility

All fluids compress when pressure increases, resulting in a decrease in volume and an increase in density. The amount of volume change varies among different fluids. At a constant temperature T , the **bulk modulus of elasticity** K represents the compressibility coefficient and defines the rate of change in pressure Δp relative to the fractional change in density $\Delta\varrho/\varrho$.

For water, under standard conditions, the bulk modulus is approximately 2100 MPa. To cause a 1% change in the density of water, a pressure of 21 MPa is required. Due to the extreme pressure needed to compress such fluids, they are considered incompressible [12]. The bulk modulus K is defined by the following equation:

$$K = \lim_{\Delta\varrho \to 0} \left[\frac{\Delta p}{\Delta\varrho/\varrho} \right]_T \tag{5}$$

Note: Other properties such as surface tension, specific heat, or ideal gases will not be explained, as they are not relevant to the purpose of this work.

2.1.5 Pressure

Pressure, denoted as p in Pascals (Pa) or (N/m^2), can be defined as the normal force F_n in N either on a gas or liquid, acting per unit area A in (m^2) on a surface [14], due to Pascals are a small unit, pressure is often given in kPa. Mathematically, it is expressed as:

$$p = \lim_{\Delta A \to 0} \frac{\Delta F_n}{\Delta A} \tag{6}$$

The pressure at any given point is called absolute pressure P_{abs}, measured relative to a perfect vacuum (absolute zero pressure). Being the difference between absolute and local atmospheric pressure P_{atm} , known as gage pressure P_{gage}. Pressures below atmospheric pressure are denominated vacuum pressures. Absolute, gage, and vacuum pressures are all positive values and are related as follows [15].

$$P_{gage} = P_{abs} - P_{atm} \tag{7}$$

$$P_{vacuum} = P_{atm} - P_{abs} \tag{8}$$

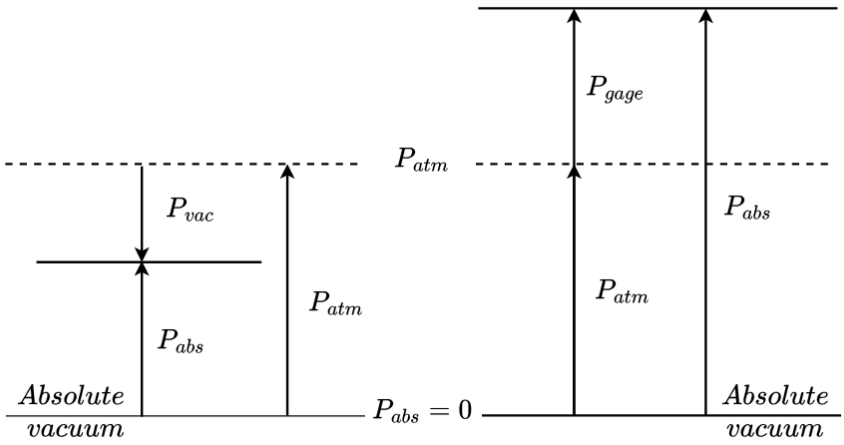


Figure 2.2: Absolute gage, and vacuum pressures. Adapted from: [15].

The pressure generated by molecules colliding with a solid surface is perpendicular to that surface. Therefore, the force direction is obtained by multiplying the pressure by a minus unit vector $\vec{n}$, which is normal to the surface, that points outward from the surface, while the force due to pressure points inward. Since the pressure acts normal to the surface, the resulting force ΔF is given by:

$$\Delta F = -p\vec{n}\, dA. \tag{9}$$

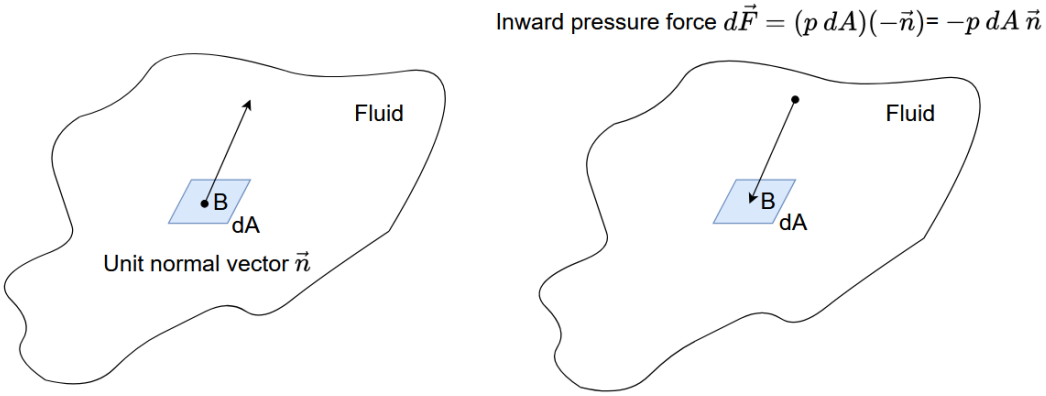


Figure 2.3: Pressure as a Force. Adapted from: [13].

2.1.6 Viscosity

Viscosity can be defined as the internal resistance that fluid molecules present to their movement, that is, their resistance to flow. This property can be interpreted as "how thick" or "how sticky" a fluid is. The higher the viscosity, the greater the resistance to flow, and vice versa. This viscosity is known as dynamic viscosity and is represented by the Greek letter μ and measure in $(Pa \cdot s)$ [12].

Temperature T has a significant impact on the viscosity of both gases and liquids. As temperature increases, the viscosity of liquids typically decreases due to the reduction in bonding forces between molecules, making them flow more easily. In contrast, the viscosity of gases generally increases with temperature because of increased intermolecular momentum transfer, which raises resistance to deformation, as can be seen in the Figure 2.4.

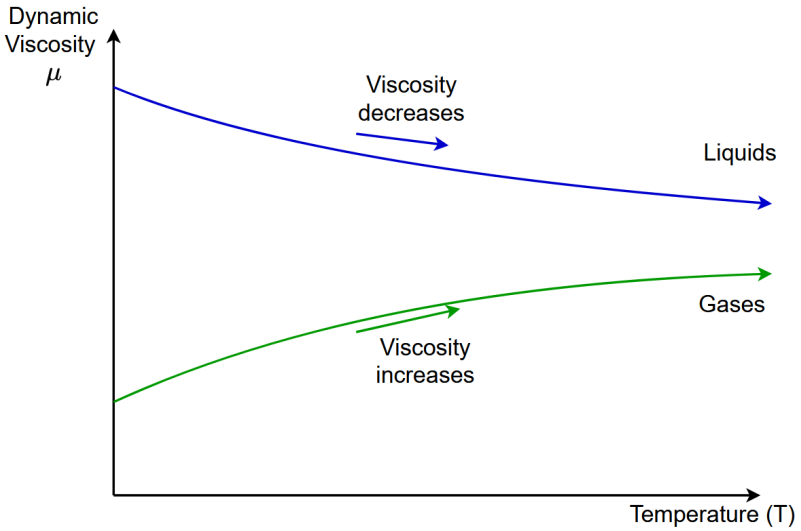


Figure 2.4: Temperature effects on the viscosity of liquids and gases. Adapted from: [13].

This resistance is related to shear forces. When a force is applied to a fluid, it causes the layers of the fluid to slide over each other. The higher the viscosity, the greater the shear force required to achieve a certain rate of flow. When different layers of a fluid move relative to each other, viscosity resists this motion, which manifests as shear forces within the fluid [15].

Visualizing Viscosity

A common way to visualize this phenomenon is through an experiment where a fluid is contained between two solid plates: one stationary plate and another moving at a velocity U máx, separated by a distance h. Viscosity causes the fluid to adhere to the surfaces, resulting in a higher fluid velocity near the moving plate and lower velocity as it approaches the stationary plate. This phenomenon is known as the no-slip condition, as shown in Figure 2.5.

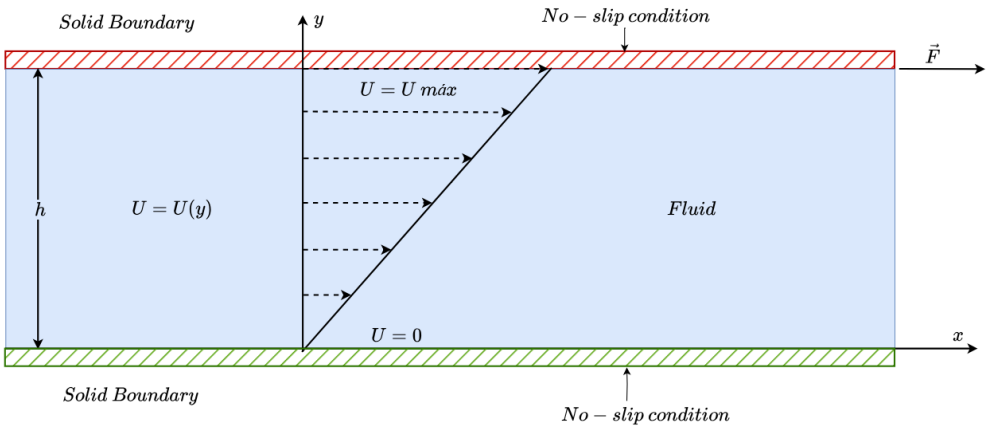


Figure 2.5: The flow between two parallel plates.
The lower is stationary, while the upper moves at a velocity of $U = U$ máx. Adapted from: [11].

The relationship between the shear stress τ (N/m^2) and the shear force F_s (N) applied to the upper plate is given by the following equation [11]:

$$\tau = \frac{F_s}{A} \ (N/m^2) \tag{10}$$

This shear stress is proportional to the velocity gradient, where the upper layer of the fluid moves at the same speed as the moving surface, and the lower layer remains at zero velocity. This allows viscosity to be defined in relation to the shear stress through the following equation [12]:

$$\tau = \mu \frac{dU}{dy} \tag{11}$$

This is true for Newtonian fluids, such as water, air, and oil. This rate of deformation for different fluids and materials can be seen in Figure 2.6.

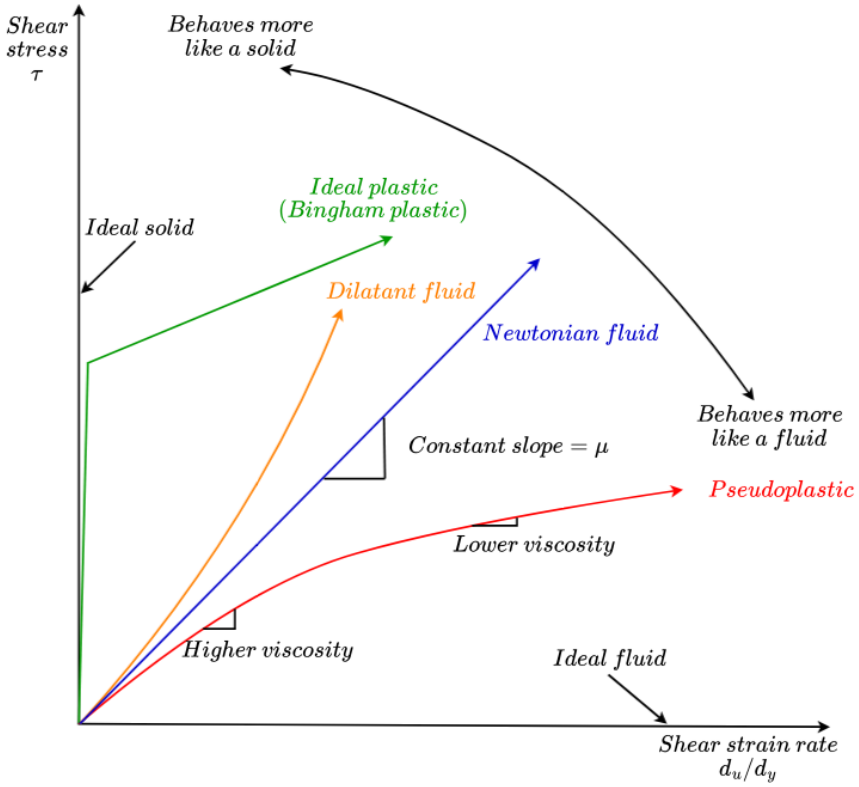


Figure 2.6: Strain rate for solid and liquids. Adapted from: [13].

The comparison between viscous forces and inertial forces, which cause fluid acceleration, is important. Since viscous forces are proportional to μ and inertial forces to ϱ, the ratio μ/ϱ is commonly used to solve fluid dynamics problems. This ratio, known as kinematic viscosity, is denoted by υ [13]:

$$\upsilon = \frac{\mu}{\varrho} \ \ (m^2/s) \tag{12}$$

2.2 Hydrostatics

Hydrostatics is the branch of fluid mechanics that studies fluids at rest. It mainly deals with how forces, particularly gravity, act on a fluid and how the fluid responds to these forces when it is in equilibrium. The fundamental principles include:

2.2.1 Pressure at a Point in a Resting Fluid (Pascal's Principle)

Pressure is the force per unit area that a fluid exerts on any surface. In a resting fluid, pressure is the same in all directions at a point, meaning it acts isotropically. Mathematically, the pressure at a given point is independent of the direction in which it is measured.

2.2.2 Variation of Pressure with Depth (Fundamental Hydrostatic Equation)

A key principle is that pressure in a fluid increases with depth due to the weight of the fluid above. As the Figure 2.7 shows and is described by the equation 13.

$$p = p_0 + \varrho\, g\, h \tag{13}$$

Where:

p : Pressure at a depth h (Pascals), (Pa or $\mathrm{kg}/(\mathrm{m}\cdot\mathrm{s}^2)$),
p_0 : Pressure at the surface (Pa),
g : Acceleration due to gravity ($\mathrm{m/s}^2$),
h : Depth below the fluid surface (m).

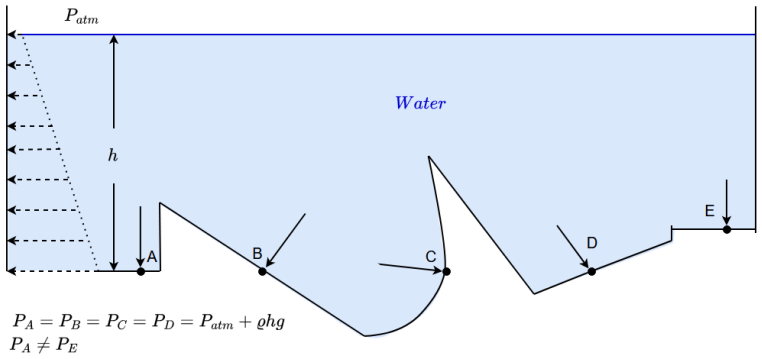
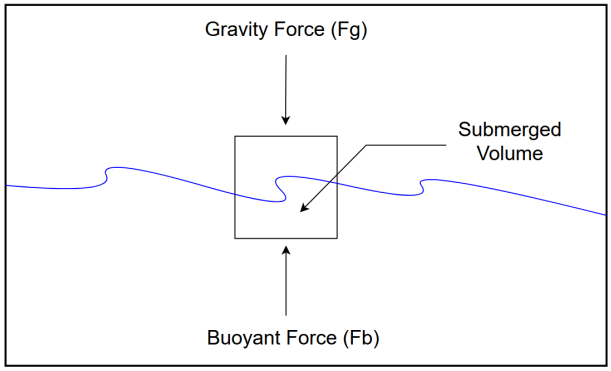


Figure 2.7: Variation of Pressure with Depth. Adapted from: [15].

2.2.3 The Buoyant force (Archimedes' Principle)

A body fully or partially submerged in a fluid experiences an upward buoyant force F_b is equal to the weight of the displaced fluid by the submerged portion of the hull. Mathematically, the magnitude of the buoyant force vector F_b where g is the gravitational constant, ϱ represents the water density, v_f is the volume of displaced water, and F_g is equal to F_b in the opposite direction, is expressed in the equation 14 and shown in the Figure 2.8 [16].



$$\|F_b\| = \varrho\, v_f\, g, \tag{14}$$

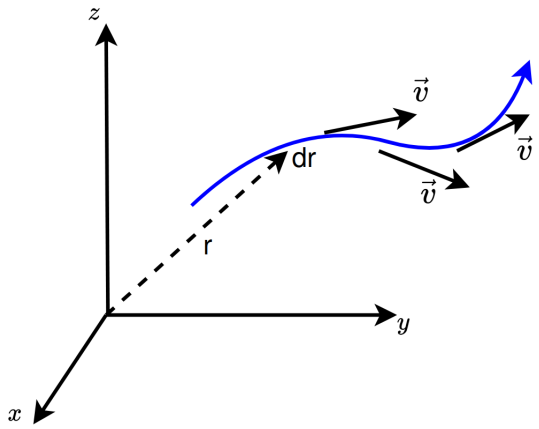
Figure 2.8: The Buoyant force. Adapted from: [16].

2.3 Fundamentals of Fluid Flow and Flow Classification

Pathlines, streaklines, and streamlines are concepts in fluid dynamics used to visualize and analyze flow behavior. A **pathline** represents the trajectory of an individual fluid particle over time, while a **streakline** traces the current positions of particles that originated from a specific point. A **streamline** indicates the instantaneous direction of the flow at each point, with the velocity vector always tangent to it, ensuring no fluid crosses the streamline.

$$\vec{v} \times dr = 0 \tag{15}$$

In steady-state flow, pathlines, streaklines, and streamlines coincide. A **streamtube** is defined by streamlines, allowing fluid to flow through without crossing its boundaries [12, 14]. In a streamline, the velocity V and the acceleration a vectors with components (u, v, w) in (x, y, z) are:



$$\vec{v} = u\hat{i} + v\hat{j} + w\hat{k} \tag{16}$$

$$\vec{a} = u\frac{\partial \vec{v}}{\partial x} + v\frac{\partial \vec{v}}{\partial y} + w\frac{\partial \vec{v}}{\partial z} + \frac{\partial \vec{v}}{\partial t} \tag{17}$$

Figure 2.9: Streamline visualization. Adapted from: [12].

2.3.1 External Versus Internal Flows

A fluid is classified as **internal** when it moves within a fully enclosed channel, such as flow through a pipe [15]. In this case, the velocity at the pipe boundaries is zero, and viscosity and friction become significant, resulting in **viscous flow** [12]. On the other hand, a fluid is classified as **external** when it moves over a surface, such as around an aircraft wing [15]. For fluids like air and water as external flows, viscosity is often neglected in analytical terms, making them **inviscid flow**, where viscous forces are negligible compared to inertial or pressure forces [12]. The movement of liquids in a duct is known as **open-channel flow** when the duct is only partially filled, resulting in a free surface [15].

2.3.2 Compressible Versus Incompressible Flows

A fluid is classified as **incompressible** if its density variation can be neglected, which is typically the case for liquids under normal conditions between $0 - 25°C$ and atmospheric pressure. Some gas flows can also be considered incompressible if their density changes by less than 5%. This classification largely depends on the speed of the fluid U (m/s) relative to the speed of sound c (approximately 346 m/s) or if the Mach Number (Ma) is less than 0.3.

$$Ma = \frac{U}{c} = \frac{Speed\,of\,the\,fluid}{Speed\,of\,sound} \tag{18}$$

In other words, under flow conditions that involve only slight changes in velocity or pressure, gases can be treated as incompressible. However, it is important to note that this approximation is valid only in specific contexts and does not apply universally to all gas flows. For higher Ma numbers, where significant changes in density occur, gases are generally considered **compressible** [15, 12], due to the increased separation of gas molecules [13].

2.3.3 Steady Versus Unsteady Flows

As in other engineering contexts, **steady** means there is no temporal change. In this context, **steady flow** refers to a flow where macroscopic properties remain constant over time at a specific point in the fluid field. As a result, a turbulent flow can be considered statistically steady [13].

In contrast, **unsteady flow** exhibits changes in these properties over time. Unsteady flow can include both transient and periodic flows. **Transient flow** refers to conditions where fluid properties are changing over time, typically during transitions between different states, such as when a valve closes or when there are sudden changes in system conditions [15]. On the other hand, **periodic flow** is characterized by oscillations around a mean value and is often observed in systems with cyclic conditions, such as pumps or turbines operating under varying loads.

It is also worth noting that while steady flow may imply a lack of temporal change, it does not necessarily mean that the system is in equilibrium; it simply indicates that properties do not change with time at any given point.

2.3.4 Laminar and Turbulent Flows

When classifying a fluid based on its motion, it can be categorized as laminar or turbulent. In **laminar flow**, the fluid moves smoothly in parallel layers without mixing, they are typically present in fluids with high viscosity and low velocity. As the fluid velocity increases, it eventually transitions to **turbulent flow**, where the motion becomes chaotic [12, 14, 15].

A flow that alternates between laminar and turbulent is known as **transitional flow**, occurring when laminar flow gradually becomes turbulent. This process is not instantaneous but happens over time or distance [12, 14, 15].

In **turbulent flow**, fluid layers mix, generating internal shear stresses that create eddies, vortices, and chaotic motion. These flows are characterized by random variations in velocity and pressure over time and space. Despite their irregularity, turbulent flows can be described using statistical averages, allowing for analysis of their macroscopic properties. Turbulent flows tend to occur at higher velocities and lower viscosities; however, they can also arise in higher-viscosity fluids under certain conditions, particularly when the Reynolds number exceeds a critical threshold [12, 14, 15].

Figure 2.10: Fluid flow transitioning from smooth laminar flow to a turbulent mixed flow. Adapted from: [13]

2.3.5 Reynolds number in Internal Flows

The transition from laminar to turbulent flow can be described using the Re, which depends on factors such as geometry, flow velocity, temperature, fluid type, and, in turbulent regimes, surface roughness. It is represented by the ratio of inertial forces, such the density ϱ and the average flow speed, U_{avg} to viscous forces μ or ν [15, 12]. In a circular pipe of diameter D, this ratio is represented by:

$$Re = \frac{\text{Inertial Forces}}{\text{Viscous Forces}} = \frac{\overline{U} D}{\nu} = \frac{\varrho \, \overline{U} \, D}{\mu} \tag{19}$$

The dimensionless Re is the key parameter for determining the flow regime in pipes. The Re at which flow becomes turbulent is called the Critical Reynolds number (Re_{cr}) [15, 12]. In pipe flow regimes, the values are:

$$\begin{aligned} \text{Laminar Flow:} &\quad Re < 2300 \\ \text{Transitional Flow:} &\quad 2300 < Re_{cr} < 4000 \\ \text{Turbulent Flow:} &\quad Re > 4000 \end{aligned}$$

Note: That surface roughness influences flow behavior primarily in the turbulent regime, where it can significantly impact drag and flow characteristics.

2.3.6 Boundary layer and Reynolds number in External Flows

In most cases, the viscous effects of a fluid are confined to a **boundary layer** δ, which is the layer of fluid in direct contact with a surface, this layer grows along the surface. As a result, shear stresses are generated, which cause the fluid to slow down. Excessive growth of this boundary layer may lead to flow separation after reaching the **critical distance** x_{cr} and the **transition region**, resulting in a loss of lift and an increase in drag. The boundary layer is thicker in turbulent flow compared to laminar flow [13].

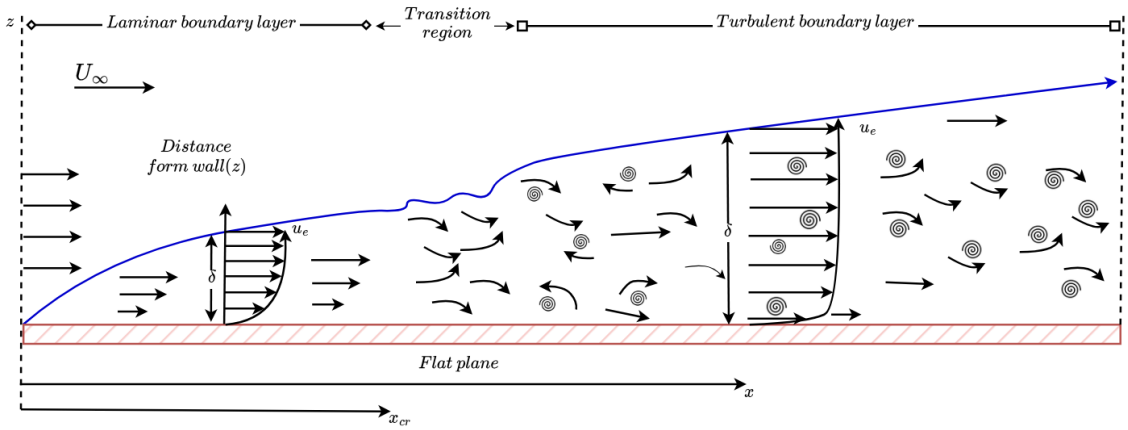


Figure 2.11: Boundary layer δ over a flat plate. Adapted from: [13].

The boundary layer can be described in terms of the Reynolds Number for external Flows (Re_L), which is related to the **characteristic length** L of the surface over which the fluid flows. This characteristic length may differ between the upper and lower surfaces of a body, taking the stagnation point (where the velocity is zero and the pressure is highest). As a result, the Re_L is given by:

$$Re_L = \frac{\varrho \, U \, L}{\mu} = \frac{U \, L}{\nu} \tag{20}$$

By convention, using the NACA0012 airfoil as a reference [1], for $Re_L > 10^5$, the flow naturally becomes turbulent [13]. Additionally, velocities of $U > 100\,\text{km/h}$ over lengths $L > 4\,\text{m}$ are often considered in practice [14].

[1] http://airfoiltools.com/airfoil/details?airfoil=n0012-il

2.4 Modeling of a Fluid in motion

This section describes the general governing equations valid for three-dimensional, unsteady flows, applicable to all types of flows (**compressible** or **incompressible**, **viscous** or **inviscid**). The motion of any fluid is governed by three fundamental principles: **conservation of mass**, **Newton's second law**, and **conservation of energy**, typically expressed through **integral** or **Partial Differential Equations (PDE)**.

Fluid dynamics focuses on the internal and external interactions of a moving fluid under the influence of forces. In the 18th century, **Daniel Bernoulli** introduced the concept of **energy conservation** in fluids, while **Leonhard Euler** developed simplified equations that neglected **viscosity**. Later, in the 19th century, **Claude-Louis Navier** and **George Gabriel Stokes** formulated the **Navier-Stokes equations**, which account for viscosity and are crucial for analyzing complex behaviors such as **turbulence**.

2.4.1 Bernoulli Equation

The Bernoulli equation describes the conservation of energy in a flowing fluid. It states that for an incompressible, non-viscous fluid, the total mechanical energy along a streamline remains constant. It is an approximate relation between pressure, velocity, and elevation, is reasonable in certain regions of many practical flows. Mathematically, it can be expressed as:

$$p + \frac{1}{2}\varrho\, U^2 + \varrho g z = \text{constant (conserved pressure)} \tag{21}$$

where p represents the fluid pressure, ϱ is the fluid density, v is the flow velocity, g is the acceleration due to gravity, and z is the height above a reference level. This equation represents the trade-off between pressure energy, kinetic energy, and potential energy in a fluid system. In an incompressible flow and that no mechanical work is introduced into or taken out of the fluid system, the equation can be rewritten as:

$$p_1 + \frac{1}{2}\varrho\, U_1^2 + \varrho g z_1 = p_2 + \frac{1}{2}\varrho\, U_2^2 + \varrho g z_2 \tag{22}$$

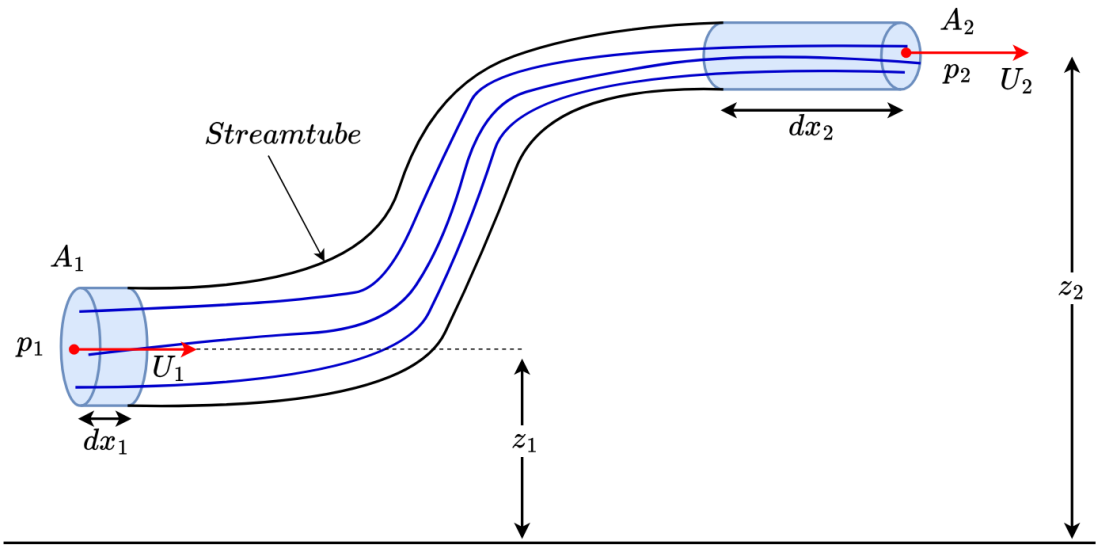


Figure 2.12: Streamtube Flow Model for Deriving the Bernoulli Equation. Adapted from: [13].

2.4.2 Continuity equation

The principle of mass conservation states that mass cannot be created or destroyed [13]. Therefore, the net mass flow $\dot{m}$ entering a control volume V must equal the mass flow leaving it, represented by the change in fluid element mass, $m_{element}$ [15].

$$\sum \dot{m}_{\text{in}} - \sum \dot{m}_{\text{out}} = \frac{\partial}{\partial t} m_{\text{element}} \tag{23}$$

This equation assumes no fluid accumulation within the control volume. For example, if we consider a pipe with varying diameters at the inlet and outlet, the volume flow rate Q remains constant [13].

$$Q_1 = Q_2 \tag{24}$$

Thus,

$$U_1 A_1 = U_2 A_2 \tag{25}$$

The differential form

In the differential form, the volume of a fluid element is:

$$d\vec{v} = dx\, dy\, dz \tag{26}$$

Thus, the mass of the fluid element is:

$$\frac{\partial}{\partial t} m_{\text{element}} = \varrho\, d\vec{v} = \varrho\, dxdydz \tag{27}$$

In Cartesian coordinates, the infinitesimal volume is represented as the Figure 2.13.

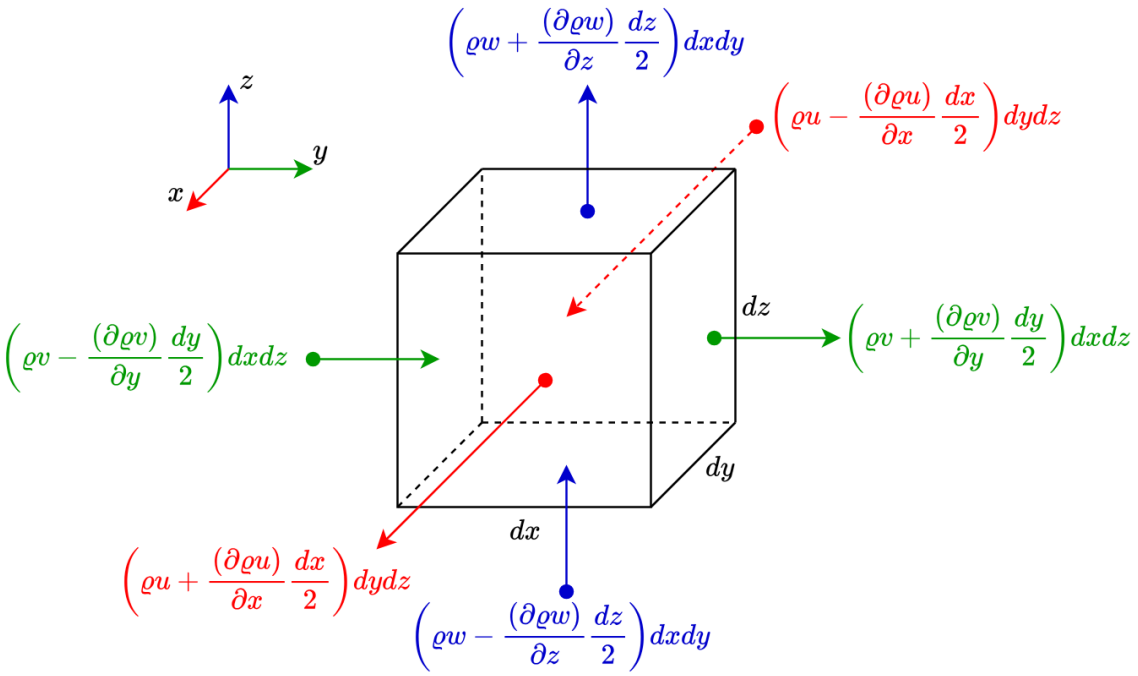


Figure 2.13: Mass inflow or outflow through each face of the differential control volume. Adapted from: [15] .

The inflow and outflow of mass through each face of the control volume can be approximated as [15]:

$$\sum \dot{m}_{\text{in}} \approx \left(\varrho u - \frac{\partial(\varrho u)}{\partial x}\frac{dx}{2}\right) dy\,dz + \left(\varrho v - \frac{\partial(\varrho v)}{\partial y}\frac{dy}{2}\right) dx\,dz + \left(\varrho w - \frac{\partial(\varrho w)}{\partial z}\frac{dz}{2}\right) dx\,dy \tag{28}$$

$$\sum \dot{m}_{\text{out}} \approx \left(\varrho u + \frac{\partial(\varrho u)}{\partial x}\frac{dx}{2}\right) dy\,dz + \left(\varrho v + \frac{\partial(\varrho v)}{\partial y}\frac{dy}{2}\right) dx\,dz + \left(\varrho w + \frac{\partial(\varrho w)}{\partial z}\frac{dz}{2}\right) dx\,dy \tag{29}$$

The **continuity equation** in Cartesian coordinates is given by:

$$\frac{\partial \varrho}{\partial t} + \frac{\partial(\varrho u)}{\partial x} + \frac{\partial(\varrho v)}{\partial y} + \frac{\partial(\varrho w)}{\partial z} = 0 \tag{30}$$

Divergence Theorem

The divergence theorem allows the transformation of a volume integral of the divergence ∇ of a vector field (e.g., velocity $\vec{v}$) into a surface integral over the boundary of the volume [15]. Thus, the continuity equation in vector form becomes:

$$\frac{\partial \varrho}{\partial t} + \nabla \cdot (\varrho \vec{v}) = \frac{\partial \varrho}{\partial t} + \vec{v} \cdot \nabla \varrho + \varrho \nabla \cdot \vec{v} = 0 \tag{31}$$

Dividing by the density ϱ, we obtain:

$$\frac{1}{\varrho}\frac{D\varrho}{Dt} + \nabla \cdot \vec{v} = 0 \tag{32}$$

If the density variations are negligible compared to the velocity gradients $\nabla \cdot \vec{v}$:

$$\varrho^{-1}\frac{D\varrho}{Dt} \approx 0 \tag{33}$$

The flow is considered **incompressible**, and we simplify the equation to [15, 12, 13]:

$$\nabla \cdot \vec{v} = 0 \tag{34}$$

2.4.3 Linear Momentum equation

The momentum equation describes the conservation of movement (momentum) in a fluid, can be written as:

$$\varrho\left(\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v}\right) = \varrho\frac{D\vec{v}}{Dt} = \varrho\vec{g} + \nabla \cdot \sigma_{ij} \tag{35}$$

Where:

ϱ : Fluid density,
$\vec{V}$: Velocity vector,
g : Gravity,
$\nabla \cdot \sigma_{ij}$: Viscous forces within the fluid

This equation follows from Newton's second law, where momentum changes equal the sum of forces. In fluids, these forces include surface forces (pressure gradients and viscosity) and body forces (gravity). The vector form of the momentum equation yields three scalar equations, aiding in determining velocity and pressure fields [13].

The momentum equation also accounts for the stress components acting on a fluid element. At any point in the fluid, there are nine stress components defined by the matrix τ_{ij}, where σ are the **normal stresses** acting perpendicular to surfaces and the are the **shear stresses** τ acting tangential to a surface, forming the **stress tensor** [12, 13]:

$$\tau_{ij} = \begin{pmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{pmatrix} \tag{36}$$

Dividing by the volume at Δ, the equation simplifies to:

$$\varrho \frac{Du}{Dt} = \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} + \varrho g_x \tag{37}$$

$$\varrho \frac{Dv}{Dt} = \frac{\partial \tau_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} + \varrho g_y \tag{38}$$

$$\varrho \frac{Dw}{Dt} = \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} + \frac{\partial \sigma_{zz}}{\partial z} + \varrho g_z \tag{39}$$

By taking moments about the center of the element, we find:

$$\tau_{yx} = \tau_{xy}, \quad \tau_{yz} = \tau_{zy}, \quad \tau_{xz} = \tau_{zx} \tag{5.3.5}$$

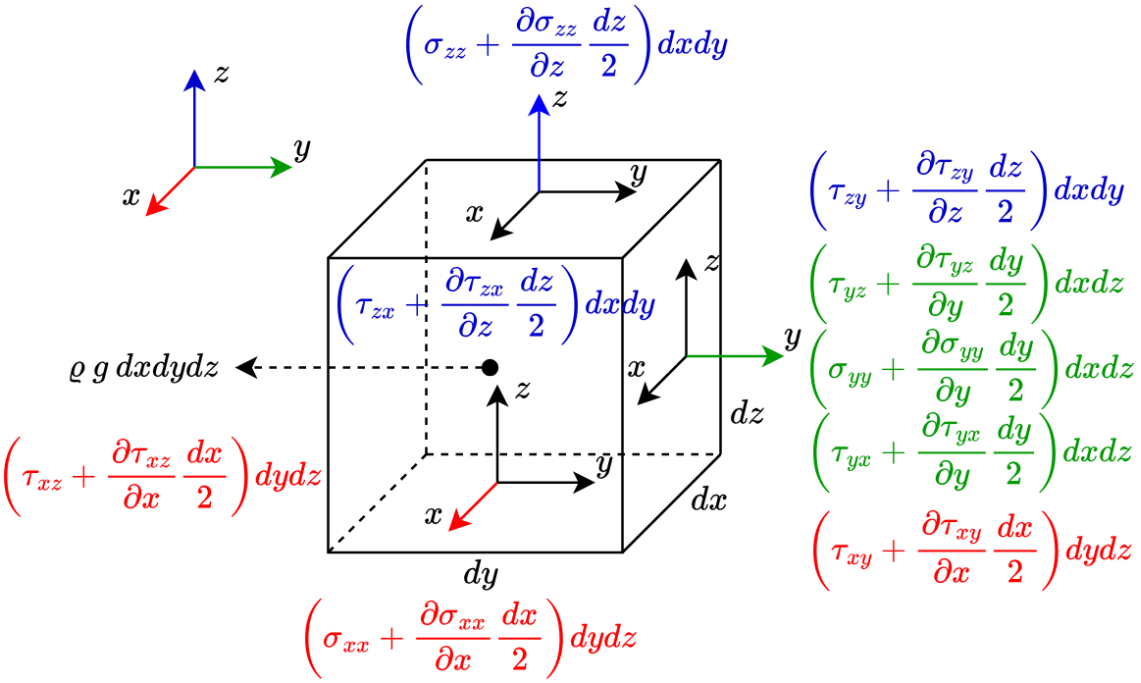


Figure 2.14: Forces acting on an infinitesimal fluid particle. Adapted from: [12] .

Note: Given the focus of this work, the differential energy equation will not be addressed. This decision is made because the study primarily focuses incompressible isotermical fluids, in which the conservation of momentum and mass are important, where thermal effects and energy transfer are negligible, as we will see in the section 2.4.4.

2.4.4 Navier–Stokes momentum equation

The Navier-Stokes equations are used to describe the motion of compressible and incompressible fluids, such as water or gases. Their complexity arises from the nonlinear **advection term**, making them difficult to solve analytically, leading to the use of **numerical methods** or approximations. These equations, along with the **continuity** and **energy equations**, are partial differential equations that describe fluid motion based on parameters such as **pressure**, **velocity**, and **viscosity**.

To derive the Navier-Stokes equations, we utilize constitutive equations. These equations relate the components of the stress tensor to the velocity and pressure fields within the fluid. When a fluid is in motion, pressure still acts inwardly on the fluid elements, but viscous stresses also become significant. To account for both pressure and viscous forces, the stress tensor σ_{ij} is composed of two parts: the diagonal pressure matrix and the viscous stress tensor τ_{ij}, which represents internal friction due to the fluid's viscosity [12, 15].

$$\sigma_{ij} = \begin{bmatrix} -P & 0 & 0 \\ 0 & -P & 0 \\ 0 & 0 & -P \end{bmatrix} + \begin{bmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \tau_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \tau_{zz} \end{bmatrix} \tag{40}$$

In this equation, P represents the pressure acting inwardly and τ_{ij} captures the viscous stresses, with each τ_{ij} term representing a component of the internal frictional forces acting in the fluid. However, this equation does not apply to incompressible fluids, as it does not account for the effects of temperature on pressure or thermodynamic pressure. In incompressible fluids, the concept of **mechanical pressure** or **mean pressure** P_m or just p is used, which is defined as the mean normal stress [15]:

$$P_m = -\frac{1}{3}(\sigma_{xx} + \sigma_{yy} + \sigma_{zz}) \tag{41}$$

Incompressible-Isothermal Flow description

This type of fluid is classified as a **Newtonian fluid**, defined as one in which the shear stress is linearly proportional to the shear strain rate. By assuming an isothermal flow, where local temperature variations are small or nonexistent, we can eliminate the need for a differential energy equation since the temperature of the fluid remains constant. Therefore, for an incompressible and isothermal fluid, the following properties are assumed [15]:

- ϱ = constant (density)
- μ = constant (dynamic viscosity)
- ϵ = constant (other relevant property, if needed)

Here, the viscous stress tensor τ_{ij} is expressed in terms of the strain rate tensor e_{ij}, which describes the rate of deformation of fluid elements. The strain rate tensor is directly related to the velocity gradients within the fluid and is given by:

$$\tau_{ij} = 2\mu e_{ij} \tag{42}$$

and e_{ij}

$$e_{ij} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) & \frac{\partial v}{\partial y} & \frac{1}{2}\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z}\right) & \frac{1}{2}\left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right) & \frac{\partial w}{\partial z} \end{bmatrix} \tag{43}$$

Applying this definition of the viscous stress tensor to the linear momentum equation and the Laplace operator, we obtain the Navier-Stokes momentum equation, defined as:

$$\varrho\left[\frac{\partial\vec{v}}{\partial t}+(\vec{v}\cdot\nabla)\vec{v}\right]=-\nabla p+\varrho g+\mu\nabla^2\vec{v} \tag{1}$$

Where:

ϱ : Fluid density,
$\vec{V}$: Velocity vector,
p : Pressure,
g : Gravity,
μ : Dynamic viscosity,
$\nabla^2\vec{V}$: Diffusive term representing viscosity.

The Continuity and Navier–Stokes Equation for Newtonian Incompressible, Isothermal Fluids

Therefore, the momentum and continuity equations for a Newtonian Incompressible, Isothermal fluid in Cartesian coordinates are defined as follows [12, 15]:

Incompressible continuity equation:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \tag{44}$$

x-component of the incompressible Navier–Stokes equation:

$$\varrho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=-\frac{\partial P}{\partial x}+\varrho g_x+\mu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right) \tag{45}$$

y-component of the incompressible Navier–Stokes equation:

$$\varrho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=-\frac{\partial P}{\partial y}+\varrho g_y+\mu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right) \tag{46}$$

z-component of the incompressible Navier–Stokes equation:

$$\varrho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)=-\frac{\partial P}{\partial z}+\varrho g_z+\mu\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right) \tag{47}$$

CHAPTER 3

Literature Review: Fluid Simulation Approaches and Models

Contents

In marine robotics, the notions seen in the chapter 2 are used in tasks such as reducing the resistance caused by water as a vehicle moves through it, to improve the efficiency of the power needed for the vehicle displacement [17]. They are also applied to maintain operational integrity during operations at different depths, aiming to preserve atmospheric pressure for the enclosed volume and many other tasks [18].

One of the applications of these approximations is their integration into digital twin simulations in virtual environments [2]. These simulations allow real-time implementation of physical phenomena on marine vehicles, recreating scenarios close to reality with minimal risk. This facilitates the design and integration of control systems, the implementation of sensors, or the adjustment of environmental variables, such as viscosity, without compromising the safety of the vehicle, even though these models do not fully reproduce real-world dynamics.

However, most simulators use pseudo forces or omit physical effects due to their complexity, the computational requirement, and the challenges to calculate them in real-time. In marine robotics, the options are limited as can be seen in table 3.1, and in the case of open-source simulators, access to the source code is often unavailable, documentation is insufficient, or many of these projects have been abandoned [19].

Simulator	Hydrodynamics	Hydrostatics	Thruster	Fins	Pressure Sensor	GPS	DVL	Realistic Rendering	Contact Dynamics	Wind	Waves	Water Currents
UWSim	✓	✗	✓	✗	✓	✓	✓	✓, osgOcean	✓	✗	✓	✗
UUV	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✗
Stonefish	✓	✓	✓	✗	✓	✓	✓	✓, custom	✓	✓	✓	✓
URSim	✓	✓	✓	✗	✓	✗	✗	✓, Unity	✗	✗	✓	✗
USVSim	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓

Table 3.1: Feature comparison of popular simulators used for marine robotics. Source: [2].

This chapter covers various approaches used to simulate fluids and hydrodynamic variables, both in real-time and non-real-time computational processing. These approaches range from methods for large-scale water simulations, such the surface of the ocean, to hybrid techniques that capture small details like splashes. Additionally, it includes methods for simulating underwater vehicles and hydrodynamic forces by applying geometric and numerical techniques, as well as data-driven fluid simulations.

The first section (3.1) focuses on simulating the ocean surface in open seas, including buoyancy forces and the effects of ocean currents and wind. The second section (3.2) explores geometrical approaches for simulating fully submerged vehicles and the use of Fossen equations. The third section (3.3) describes numerical models for fluid motion using Lagrangian and Eulerian approaches. The fourth section (3.4) discusses hybrid methods for simulating oceans, rivers, and small-scale effects such as splashes. Finally, the fifth section (3.5) presents current fluid simulation methods based on data-driven approaches, with an emphasis on applications of Physics-Informed Neural Networks (PINN).

3.1 Ocean Surface simulation

This type of simulation emphasizes wave dynamics in both shallow waters and the open ocean. Shallow water simulations capture wave movements near the coast, where the depth significantly affects wave dynamics, including non-sinusoidal behavior in some cases. On the other hand, in open ocean simulations, the fast Fourier transform is used to decompose small amplitude sinusoidal wave components in motion in the ocean, following the principles of Airy wave theory, that describes the propagation of small amplitude gravity waves on the surface of a homogeneous fluid, such as water. It assumes the fluid is incompressible, inviscid, and irrotational, with a uniform mean depth. [20].

Waves appear across the entire ocean surface, ranging from tsunamis generated by seismic movements on the seafloor to tidal waves (tides), which are caused by the gravitational forces of the Moon and the Sun, and wind-generated waves are the most commonly used for simulating the ocean surface in marine robotic simulators and video games. This type of wave originates from atmospheric disturbances, such as wind, storms, and gusts. Surface waves are formed through frictional stress between the wind and the water surface due to the difference in velocities between the two fluids [21]. This frictional wind stress τ_w (force per unit area) is defined by the equation 48 [22].

$$\tau_{\mathbf{w}} = \varrho_a C_d U^2 \quad (N/m^2) \tag{48}$$

Where:

$\tau_{\mathbf{w}}$: Is the frictional wind stress

ϱ_a : Is the density of air,

C_d : Is the drag coefficient, which depends on wind speed and surface conditions,

U : Is the wind speed at a reference height above the water.

As wind moves across open oceans and coastal regions, it generates waves. Contrary to common belief, waves do not significantly transport water; instead, they primarily transmit energy through water, often in a circular motion. Although waves can traverse entire ocean basins without significant water displacement, there is a small net forward movement of water, known as *Stokes drift*, which occurs near the surface. The diameter of this circular motion decreases with depth in open water, while in shallow water, waves take on an elliptical motion, as illustrated in Figure 3.1.

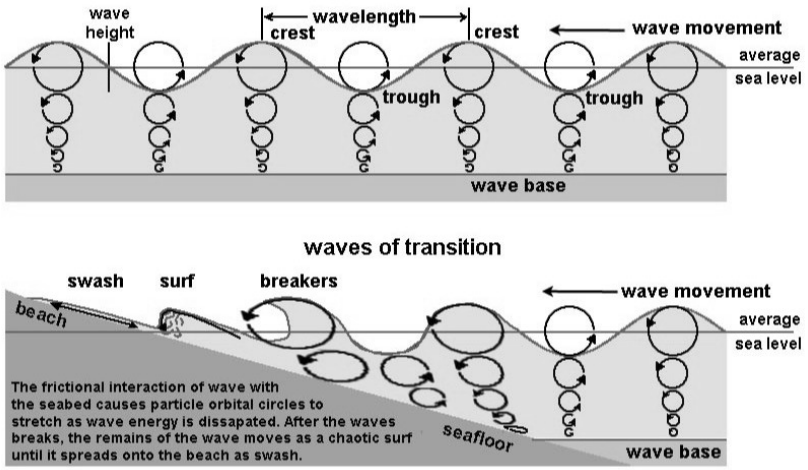


Figure 3.1: Wave Circular motion in open and shallow waters. Source: [21].

3.1.1 Wave Simulation Process

Trochoidal or Gerstner wave

The circular motion of a wave, as illustrated in Figure 3.1, represents a simplified version of wave behavior, commonly known as **trochoidal** or **Gerstner waves**. These waves are characterized by both horizontal and vertical components:

$$x(a,b,t) = a + \frac{e^{kb}}{k}\sin(k(a+\omega_{\mathbf{a}}t)) \tag{49}$$

$$y(a,b,t) = b - \frac{e^{kb}}{k}\cos(k(a+\omega_{\mathbf{a}}t)) \tag{50}$$

where a and b are the coordinates of the center of the circular path, $\omega_{\mathbf{a}}$ is the angular velocity, which determines how fast a complete rotation is made around the circumference, and k is the wave number. The wave number k measures how many wavelengths fit in a unit of distance, typically in radians per unit length. While this model provides a stylized representation, it only offers an idealized approximation of the ocean's surface the implementation of these waves can be found in the tutorial made by *Jasper Flick* [1] , and a more detailed explanation of them is available in the paper *Simulating Ocean Water* by Tessendorf [23].

However, to achieve realistic wave simulations, it is necessary to understand two main approaches: **wave dynamics**, which focuses on the description of individual waves, and the **spectral models**, which uses a stochastic methods to characterize wave fields in specific oceanic regions, where the surface of the ocean is represented as a superposition of many individual waves with different frequencies, directions and random amplitudes [24]. The spectral approach considers variables such as wind direction and strength, water depth, and coastline shape to represent the distribution of wave energy over different frequencies and directions.

[1] https://catlikecoding.com/unity/tutorials/flow/waves/

Ocean as an infinite sum of waves (Wave dynamics)

A 2D wave dynamic can be described as a sinusoidal wave η with z direction as surface elevation. Since both the angular velocity ω_d and the wave direction θ depend on the wave number k, each individual wave can be fully described by the parameters (k_i, a_i, ϵ_i), where, t is the time, a_i is the amplitude, and ϵ_i is the phase. As result, the surface of the ocean can then be modeled as the sum of an infinite number of these 2D waves, as the equation [25]:

$$\eta(r,t) = \sum_{i=0}^{\infty} a_i \cos(k_i \cdot r - \omega_{\mathbf{a}i} t + \epsilon_i) \tag{51}$$

Where:

$$\begin{aligned} r &: (x, y)\ (Wave\ vector) \\ k_i &: (k_i \cos\theta_i, k_i \sin\theta_i) \\ k_i &: (k_{ix}, k_{iy}) \end{aligned}$$

However, representing the ocean surface using a sum of sinusoidal functions is computationally expensive. To make this process more efficient, Fast Fourier Transform (FFT) can be applied, expressing the cosines as complex exponential. In this way, the displacement of the surface, represented as a sum of waves, can be transformed into a more manageable form if the coordinates and wave numbers k are arranged on a regular grid as can be seen in the Figure 3.2. and represented using the Discrete Fourier Transform (DFT) when:

$$\begin{aligned} &1.\text{Number of spatial points} = N \\ &2.\text{Coordinates } x = \frac{m}{N}L, \quad \text{wave numbers } k = \frac{2\pi n}{L} \\ &3. m, n \in \left[-\frac{N}{2}, \frac{N}{2}\right] \end{aligned}$$

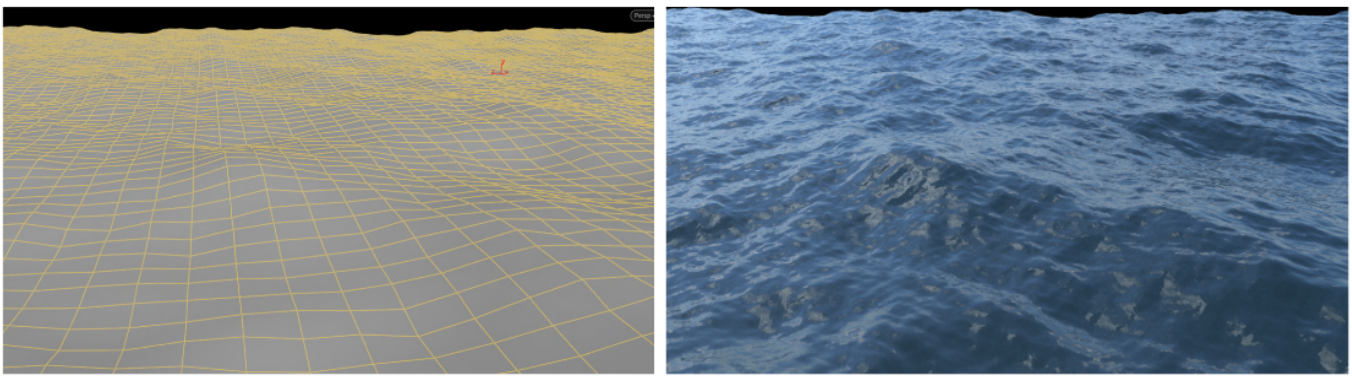


Figure 3.2: Visualization of the ocean surface (left) and the corresponding rendered image (right). Source: [26].

As a result of applying the Fourier transform to equation 51, we obtain the surface displacement in Fourier space:

$$\eta_n(t) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}} h_m(t) e^{2\pi i \frac{m}{N} n} \tag{52}$$

In this expression, $h_m(t)$ are the Fourier coefficients, representing the amplitude and phase of each wave component, while the exponential term introduces periodicity in space.

Ocean as a spectrum model

As we have discussed, waves are characterized by their wave vectors, wavenumbers, and frequencies. In a spectral model, these parameters are related through a dispersion relation ω_d. The disturbance force driving the waves is the wind, while the restoring forces include gravity g and water depth h.

In the open ocean, wave amplitudes follow random patterns [27], which are described by the non-directional spectrum $S(\omega_d)$. This spectrum represents the analysis and modeling of time series data of wave height. Additionally, the directional distribution $D(\omega_d, \theta)$ describes the tendency of waves to follow the wind direction, depending on the angle between the wave vector and the wind θ, as well as the wave frequency. These factors and the group phase velocity v_g, which is the speed as the entire wave packet moves, defines the spatial spectrum $P(k)$.

$$P(k) = S(\omega_d)\, D(k, \omega_d) \frac{v_g(\omega_d)}{2} \tag{53}$$

Ocean waves include wind-driven waves and swell, with the latter forming far from their origin. Both types of waves are modeled using distinct spectral functions, to simulate their real-world behavior, with adjustments made to account for factors such as water depth and wave direction. Table 3.2 presents some of the most common dispersion relations and group velocities implemented [5].

	Dispersion relation	Group velocity
Deep Water	$\omega_d^2 = gk$	$v_g = 0.5\,(\omega_d/k)$
Shallow Water	$\omega_d^2 = gk\,\tanh(kh)$	$v_g = 0.5\,(\omega_d/k)\left(1 + \frac{kh(1-\tanh^2(kh))}{\tanh(kh)}\right)$
Capillary Waves	$\omega_d^2 = gk\left(1 + (k\ell_c)^2\right)$	$v_g = 0.5\,(\omega_d/k)\left(1 + \frac{2(k\ell_c)^2}{1+(k\ell_c)^2}\right)$
Combined	$\omega_d^2 = gk\,\tanh(kh)\left(1 + (k\ell_c)^2\right)$	$v_g = 0.5\,(\omega_d/k)\left(1 + \frac{kh(1-\tanh^2(kh))}{\tanh(kh)} + \frac{2(k\ell_c)^2}{1+(k\ell_c)^2}\right)$

Table 3.2: Dispersion and group velocity expressions. Source: [5].

In the case of **shallow waters**, models account for the influence of the seabed on wave propagation. Since the water depth is comparable to the wavelength, waves interact significantly with the bottom, leading to non-sinusoidal behavior. As waves approach the shore, they slow down and increase in height. Simulations of this nature require methods capable of capturing these effects, such as finite difference models combined with convolution methods, which discretize variables along both the seabed and the water surface, an implementation of the spectral method pipeline is shown by Ivan Pensionerov [2]. Recently, in 2023, an approach was implemented that resolves and combines open ocean and shallow water waves into a heightfield field wave model capable of simulating complex interactions between the two approaches [28].

As discussed in this section, simulating the complexities of natural ocean phenomena and achieving accurate representations is a challenging task. The ocean's highly dynamic behavior encompasses a wide range of conditions, from calm seas to turbulent waters, and from small ripples to large breaking waves at the shore. These movements are influenced by various phenomena occurring across multiple scales [29]. Next, we will explore some implementations for simulating the forces exerted by the water surface on objects in contact with it.

3.1.2 Object interactions with water and hydrostatic forces

In general, to achieve real-time behavior, rigid bodies are used to simulate vehicles or objects above or below the water surface. These rigid bodies are represented as meshes with no real interaction with the water. In real-time open ocean simulations, Fourier-spectral-based methods lack the ability to handle interactions with moving obstacles, while optimized finite-difference or finite-element methods, which can accommodate complex environmental interactions, remain computationally expensive [30]. Therefore, mathematical methods and simplifications are applied to approximate forces such as buoyancy, wind, and ocean currents [16].

For example, although **buoyant force** is often simulated with reasonable accuracy, other parameters such as the specific shape of the object or its mass distribution are often not fully considered. Many simulations use bounding boxes or simplified meshes to approximate the shape of the object, which can introduce inaccuracies. Additionally, the actual waterline, where the object should be submerged based on its mass and geometry, is often not precisely accounted for.

[2] https://www.youtube.com/watch?v=kGEqaX4Y4bQs/

Wind effects

The movement and orientation of a rigid object on the water's surface is influenced by the propeller or by external forces, such as wind, water currents, and waves. Given the absence of sails in most modern marine vehicles, the effects of wind can be neglected, except when considering their impact on the vehicle's hull. Lateral winds cause oscillating roll torque, whereas frontal winds result in oscillating pitch torque [16].

The pitch angle ϕ_{pitch} and roll angle ϕ_{roll} of a surface vehicle can be determined based on the alignment with the wind directional vector $\mathbf{p}_w$, as well as vehicle dynamic factors such as the pitch and roll coefficients α_{pitch} and α_{roll} [16], which account for acceleration and velocity in both degrees of freedom.

$$\phi_{\text{pitch}}\left(\mathbf{p}_m^y, \mathbf{p}_w, \nu_w, t\right) = \left(\mathbf{p}_m^y \cdot \mathbf{p}_w\right) h(\nu_w, t)\alpha_{\text{pitch}} \tag{54}$$

$$\phi_{\text{roll}}\left(\mathbf{p}_m^x, \mathbf{p}_w, \nu_w, t\right) = \left(\mathbf{p}_m^x \cdot \mathbf{p}_w\right) h(\nu_w, t)\alpha_{\text{roll}} \tag{55}$$

Water currents effects

Water currents can exert linear and rotational forces on surface vehicles, such as yaw torque (τ_r). The yaw torque is computed as:

$$\tau_r = \mathbf{r} \times F_c \tag{56}$$

where F_c is the force applied to the object, and $\mathbf{r}$ is the displacement vector from the center of mass to the point of force application, and the yaw rotation can be defined by:

$$F_x(\mathbf{p}_x^m, \mathbf{p}_w) = (\mathbf{p}_x^m \cdot \mathbf{p}_w)||\mathbf{p}_w|| \tag{57}$$

$$F_y(\mathbf{p}_y^m, \mathbf{p}_w) = (\mathbf{p}_y^m \cdot \mathbf{p}_w)||\mathbf{p}_w|| \tag{58}$$

where $||\mathbf{p}_w||$, F_x, F_y is the magnitude of the wind force, thrust force, lateral force, respectively. The interaction depends on the wind and drag force direction relative to the vehicle orientation.

Other approaches

Other methods approximate non-hydrostatic forces on 3D models using Proportional Integral Derivative Controllers (PID) to adjust force, dampening, and mass distribution on the ship [31]. Some approaches rely on experimental data, stochastic wind wave descriptions, and numerical models to simulate interactions between submerged bodies and irregular water surfaces under various wind and wave conditions, as shown in Figure 3.3 [32]. However, these methods struggle with non-linear effects like breaking waves or splashes. Fourier-based approaches break down liquid motion using amplitude functions to handle local interactions in static environments [30].

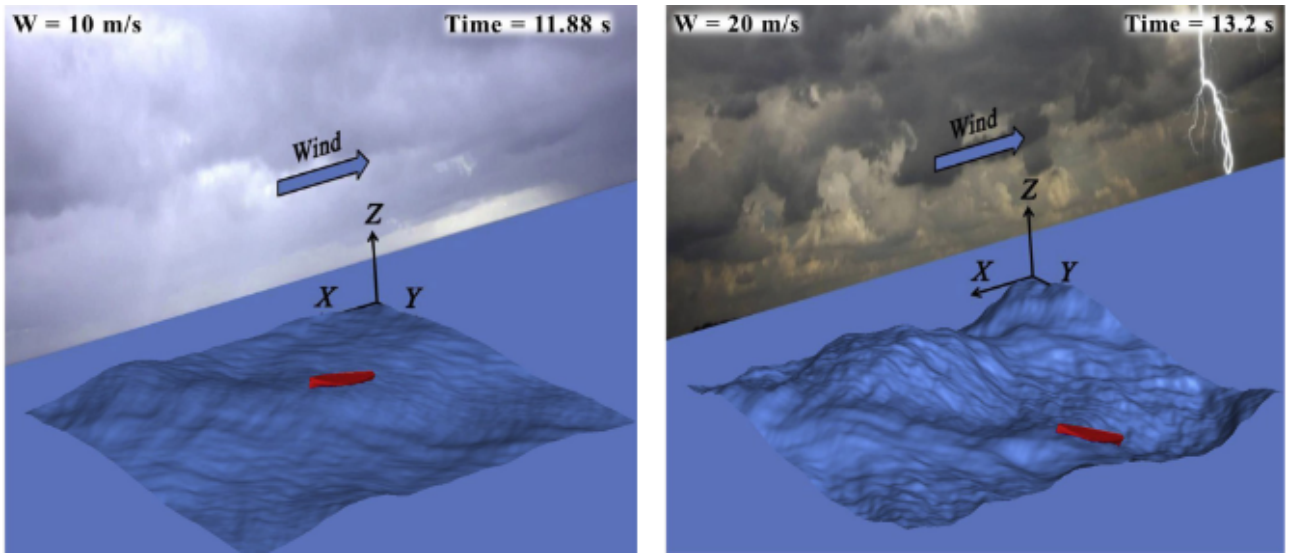


Figure 3.3: A. Zheleznyakova algorithm approach results under different wind conditions. Source: [32].

3.2 Underwater Ocean simulation

Underwater, the effects of the fluid significantly influence the entire vehicle. For example, when an underwater vehicle accelerates, it displaces a volume of water, resulting in added mass that exerts pressure on the hull, perpendicular to the surface [33], and requires additional energy to move [34]. The fluid resistance to this motion is determined by the pressure distribution and flow separation along the hull [35]. For fully submerged objects, the shape of the object and fluid distribution are studied using two main approaches: **geometric approach** and **numerical methods**, which will be detailed below.

3.2.1 Geometrical-Based Methods and Fossen Equations

A typical UV consists of three main components: the hull, diving planes or fins, and the propeller [3]. The shape of the vehicle influences how water flows around it and is often optimized using geometric methods, such as the Myring profile equations [36]. Basic geometries or slender body theory are used to simplify the shape of the vehicle in marine robotics. These approximations help to understand the forces applied to the vehicle by the surrounding fluid medium and can be broken down into added mass, lift, drag, and other forces [3].

The geometrical simplifications are mainly used for the calculation of the added mass. When a body moves or accelerates in a fluid, the surrounding fluid moves adding a resistance, acting as an additional or "added mass" that increases the force required for acceleration. A geometrical simplification for the hull of a typical Torpedo shaped UV is the represented in the Figure 3.4.

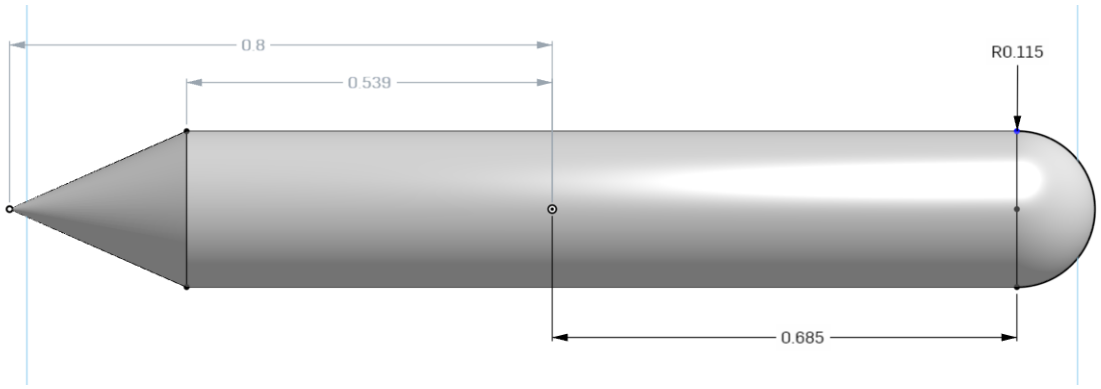


Figure 3.4: Geometrical simplification for the hull of a typical Torpedo shaped UV in meters.

Other approximations employ ellipsoids, with the major axis equal to the vertical length of the vehicle and the minor axis equal to the width of the vehicle [3, 37], as shown in Figure 3.5.

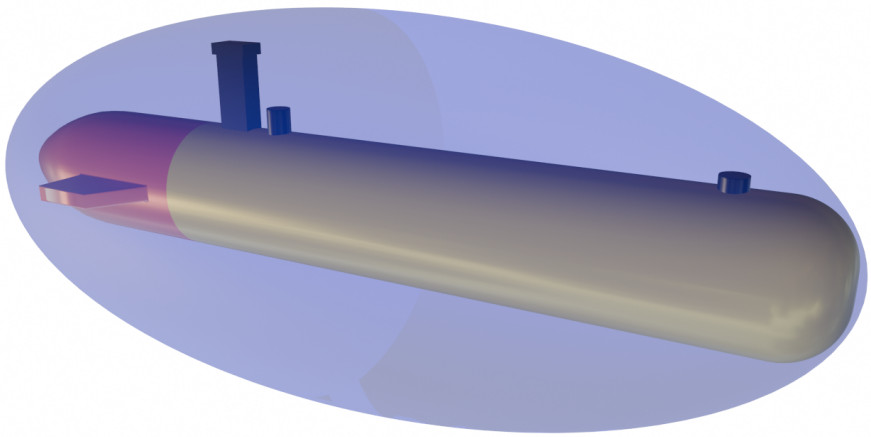


Figure 3.5: Ellipsoidal simplification for a typical Torpedo shaped UV.

These combined approaches are used to calculate hydrodynamic variables, minimizing computational requirements. They can be applied to parametrize vehicle dynamics across different displacement directions or Degrees of Freedom (DOF) as is represented in the Figure 5.1.

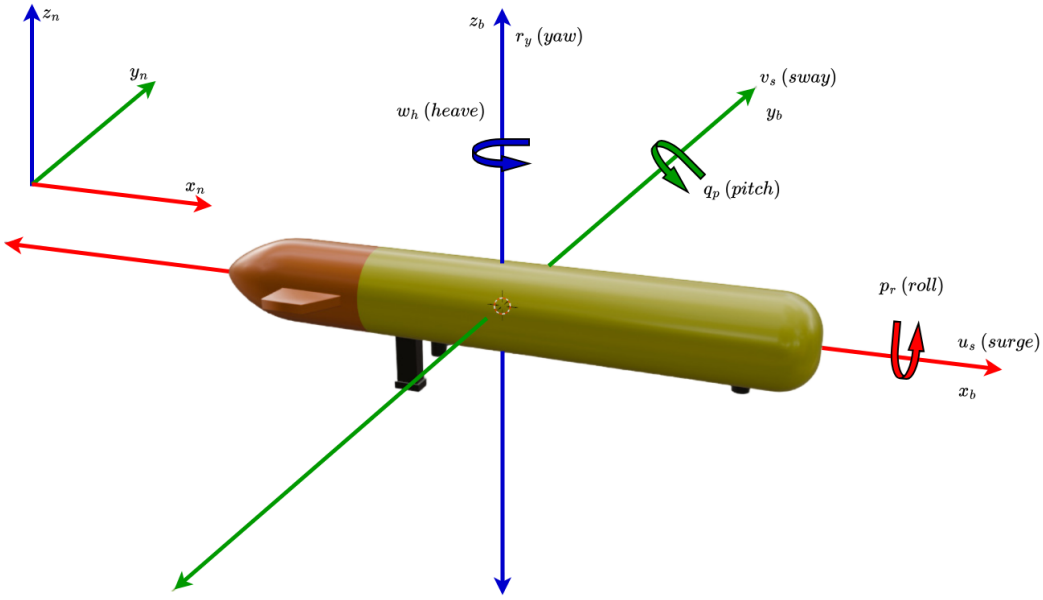


Figure 3.6: Degrees of Freedom (DOF) in an Underwater Vehicle UV. Adapted from: [38].

Often using variations of the Fossen's equations (Equation 59). This approach is not limited to the vehicle's equilibrium region and uses local symmetry of components, even if the entire vehicle, including diving planes, is not symmetric. Added mass and inertia are often determined using experimental methods like the Planar Motion Mechanism (PMM), rotating arm, and circular motion tests [39].

$$M\dot{v} + C(v)v + D(v)v + g(\eta) + g_o = \tau + \tau_{\text{wind}} + \tau_{\text{wave}} + \sigma_c \tag{59}$$

Where:

M : Inertia Matrix (including added mass)
$C(v)$: Matrix of Coriolis Centripetal term (Including added mass)
$D(v)$: Damping matrix
$g(\eta)$: Vector of gravitational/bouyancy forces and moments
g_o : Vector used for pretrimming (ballast control)
τ : Vector of control inputs
τ_{wind} : Vector of wind loads
τ_{wave} : Vector of wave loads
σ_c : Current induced disturbances (constant)

However, this model implies the following assumptions and limitations [40]:

1. Address the objects as fully submerged, ensuring constant added mass coefficients.
2. Usually neglects waves, currents, and disturbances.
3. Assumes an inviscid, frictionless fluid with no circulation.
4. Vehicle's rotational symmetry around the x-axis with identical cross-sections in the XY and XZ planes.
5. Treats vehicle components in isolation, ignoring interactions.

3.3 Perspectives on Fluid Motion

A fluid in motion can be described from two different perspectives. One of them, **Lagrangian description**, that approaches it as the movement of individual particles over time, while the **Eulerian description** focuses on the behavior of a specific region of the fluid as it evolves. Both models assume a continuous fluid, composed of small infinitesimal regions [14]. If the fluid is incompressible, its volume does not change in magnitude but may deform [12].

3.3.1 Lagrangian Description

Lagrangian methods describe the behavior of a small volume of fluid represented by an individual particle P_L, with Cartesian coordinates x_o, y_o, z_o over time t, which moves according to the forces acting upon it. The velocities corresponding to each coordinate u, v, w are the first partial derivatives of each position component with respect to time, while the accelerations a_x, a_y, a_w are the second partial derivatives [14]. In the Lagrangian description, many particles can be tracked, and their influence on one another can be observed. The accuracy of this method depends on the number of particles that are followed [12].

$$P_L = P_L(x_0, y_0, z_0, t), \quad \vec{v} = \frac{\partial P_L}{\partial t}, \quad \vec{a} = \frac{\partial \vec{u}}{\partial t} \tag{60}$$

In these methods, the trajectory of each fluid particle is tracked based on its initial position and the applied forces, allowing a detailed representation of a fluid behavior, especially in cases of large deformations or complex movements.

Smoothed Particle Hydrodynamics

The most common computational approach to Lagrangian methods is through Smoothed Particle Hydrodynamics (SPH), initially developed for astrophysics [41]. In this method, each particle is assigned properties like density, pressure, and velocity, which are smoothed using interpolation functions (kernels) that account for nearby particles. Fluid dynamics equations, such as Navier-Stokes, are solved locally for each particle using solvers.SPH is particularly effective for representing fluid surfaces and volumes at small scales [42, 43].

However, in marine robotics, SPH techniques have only been implemented recently due to the high computational requirements needed to simulate large numbers of particles, which limits their real-time applications but improves the accuracy [7]. In 2022, the **SPlisHSPlasH** SPH library [3] was used to simulate fluid interactions with rigid bodies in Gazebo [4], an open-source robotics simulator. It was applied to simulate autonomous swimming with soft robotics and swimming robots, as shown in Figure 3.7, with improvements in neighborhood search algorithms and pressure solvers [44].

In 2023, Incompressible Smoothed Particle Hydrodynamics (ISPH) was used to simulate fully submerged vehicles in shallow waters, successfully reproducing buoyancy and added mass on the same order of magnitude as the geometric reference model. However, calculating other hydrodynamic variables required adjustments in the number, diameter, and density of particles, moving the results away from real-time computation [45], as shown in Figure 3.8.

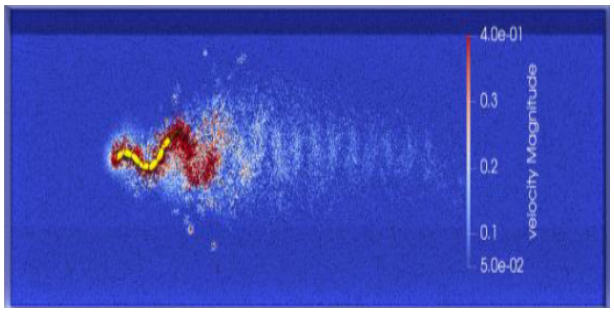


Figure 3.7: Simulation of an amphibot robot a tank of particles. Source: [44].

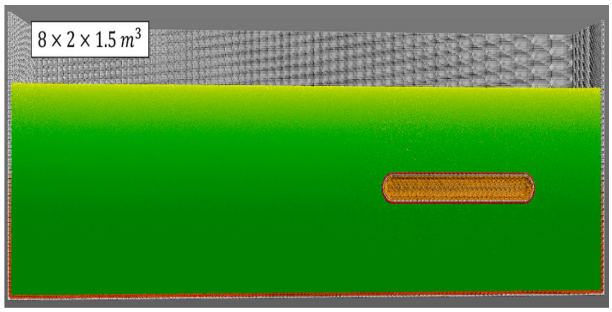


Figure 3.8: Underwater vehicle inside a tank of particles. Source: [45].

[3] https://splishsplash.readthedocs.io/en/latest/about.html
[4] https://gazebosim.org/home

3.3.2 Eulerian Description

The Eulerian description does not track the path of a specific fluid particle; instead, it focuses on how fluid properties behave at a specific point P_s within a control volume over time. Under this approach, key fluid properties such as the density ϱ, the velocity vector field $\vec{u}$, the acceleration vector field $\vec{a}$, and the scalar pressure field p are analyzed as functions of both position and time [46].

$$P_s = (x, y, z), \quad \vec{v} = \vec{v}(P_s, t), \quad \vec{a} = \vec{a}(P_s, t), \quad p = p(P_s, t), \quad \varrho = \varrho(P_s, t) \tag{61}$$

Eulerian methods generally rely on meshes, which divide the control volume into discrete cells, where fluid velocity and pressure are represented as fields within each cell. This description is often easier to apply in practical simulations, though both Eulerian and Lagrangian approaches should ideally lead to the same conclusions [15, 12]. One of the most common applications of this method is in CFD, where fluid flow is analyzed over time using fixed spatial grids to compute the fluid properties at each point.

Computational fluid dynamics (CFD) Method.

This method is the computational approach for solving fluid flow problems by Eulerian description. The governing Flow equations, which typically include the Navier-Stokes equations, are solved using numerical techniques. These equations, which describe the conservation of mass, momentum, and energy, are partial differential equations (PDEs).

This method employs discretization methods to convert these PDEs into a system of algebraic equations that can be solved numerically. One of the most widely used discretization methods in CFD is the finite volume method (FVM). In this approach, the computational domain is divided into small, discrete control volumes or cells through a process called meshing as is shown in the Figure 3.9. The mesh can be structured (regular) or unstructured (irregular), depending on the complexity of the geometry and the desired level of accuracy.

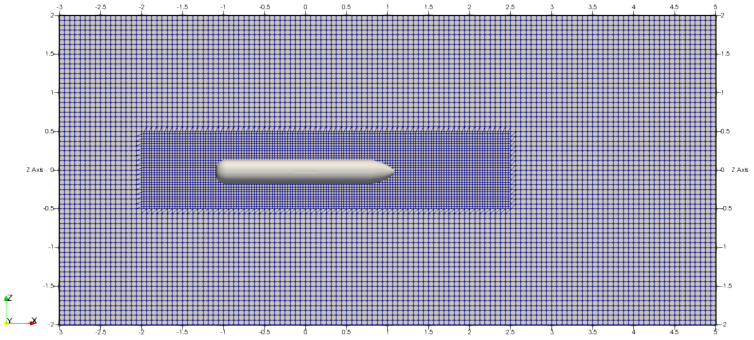


Figure 3.9: CFD mesh around a torpedo-shaped UUV used in this work.

Within each control volume, the governing equations are integrated, and the resulting algebraic equations are solved iteratively. This process calculates fluid properties, such as velocity, pressure, temperature, and density, at specific points in the computational domain. By discretizing both space and time, CFD provides numerical solutions that are analogous to real-world wind tunnel experiments [47], allowing for the simulation of fluid flow phenomena in a wide range of applications, including aerospace engineering, automotive design, process engineering, and environmental studies.

In marine robotics, CFD is used to calculate the hydrodynamic characteristics of underwater vehicles, such as the influence of the angle of attack on drag and lift coefficients. These results are utilized to optimize vehicle design in virtual environments that closely mimic real-world conditions. Examples include studying the dependency of steady-state velocity to improve power performance in underwater vehicles [36], or analyzing drag coefficients with and without rudders, as the Figure 3.10 shows.

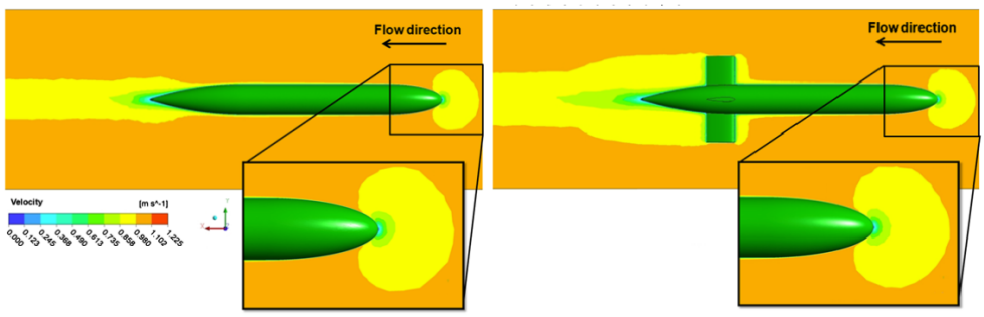


Figure 3.10: Velocity field around an underwater vehicle with and without rudders. Source: [4].

3.4 Hybrid Approaches

As the name suggests, hybrid approaches combine Lagrangian, Eulerian, and spectral techniques to achieve realism and real-time simulation, often with the help of Graphics Processing Unit (GPU). In 1986, the popular Fluid-Implicit-Particle (FLIP) method was introduced, nowadays is widely used in video games and movies, in this method, a 2D fluid flow is represented using particle-in-cell technique on adaptive grids, minimizing numerical dissipation and handling large data variations for small volumes of fluid [48]. In 2001, a method which uses grid-based spectral methods for open ocean waves, incorporating numerical calculations and artificial damping for subtle details [24].

In 2004, a 2D GPU implementation of the Navier-Stokes equations for incompressible flow was developed [49], followed in 2005 by a Lagrangian simulation of viscoelastic fluids reaching 10 Frames Per Second (FPS) particles [50]. In the same year, a shallow-water nonlinear wave simulation 30 times faster than the Central Processing Unit (CPU) was achieved [51]. In 2006, calculations for mass and buoyant forces were achieved for 50 floating objects with a rate of 16 FPS[52]. Within one year, a model reduction method for precomputed fluid simulations was projected them into a low-dimensional solution space [53].

By 2007, water surface simulations were split into two components: 3D low-frequency fluid flow and 2D high-frequency surface waves [54], and a full 3D GPU simulation achieved 120-180 FPS on a 64x64x128 grid [55]. During the same year, simulations of waves and interactions with floating objects were successfully achieved by simplifying the wave representation, transforming wave particles into water surface heights [5] [6].

In 2010, the Height Field Fluid method became popular for simulating large bodies of water, like oceans and lakes, specially because they were able to reproduce foam and splashes. This method used shallow water solvers to account for topography, water depth variations, simulating currents, breaking waves, and waterfalls, and representing foam and spray as particles [6]. GPU parallelization made this approach widely adopted in video games [8]. Examples of the implementation of these two last approaches were added as footnotes.

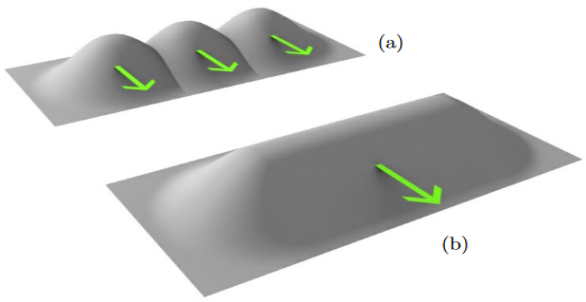


Figure 3.11: (a) Individual wave particles (b) Wavefront formed by these wave particles. Source: [6].

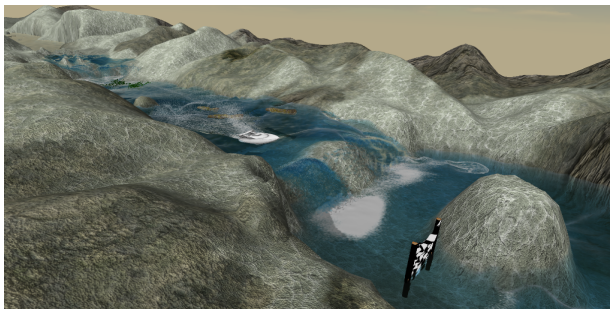


Figure 3.12: Small waterfalls,foam, spray, splashes, small-scale waves, and rigid body water interaction. Source: [8].

[5] https://www.youtube.com/watch?v=qR09XP-S8wM
[6] https://www.youtube.com/watch?v=bojdpqi2l_o

3.5 Data-driven Fluid Dynamics

With the increasing availability of large volumes of data from numerical simulations, new strategies based on data-driven applications have been implemented using processing and compression techniques. From the exploration of turbulent flows [56] to wall turbulence and control for energy generation [57]. These strategies were first proposed to analyze local turbulences statistically in incompressible viscous fluids at high Reynolds numbers in 1940 [58]. In the 1960s, Ingo Rechenberg and Hans-Paul Schwefel tested the aerodynamic drag of five corrugated plates by randomly varying their angles using a Galton board, adjusting each variation based on the success of previous configurations [59].

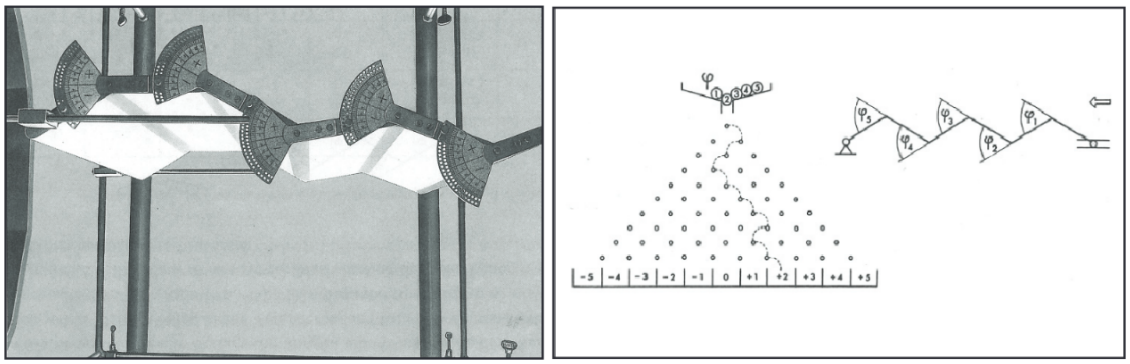


Figure 3.13: Drag reduction experiments used the Galton board as a random number generator. Source: [59].

In 1980, Artificial Intelligence (AI) began to be applied in aerodynamic design with the help of CFD software [60]. In 1994, neural networks were used to approximate solutions to partial differential equations, and studies demonstrated their potential in solving both ordinary and partial differential equations [61]. Three years later, they were used to solve initial and boundary value problems by splitting the solution into two parts: one satisfying the boundary conditions, and the other involving a feed-forward network trained to satisfy the differential equation [62]. Similar approaches were used in the modeling of distributed nonlinear systems using neural networks [63].

As mentioned earlier in section 3.3, the interactive process for CFD involves a high computational cost and is not suitable for real-time applications. Therefore, in recent years, Deep Learning (DL) solutions have been implemented to solve the differential equations that describe fluid motion in the context of surrogate fluid models or other physical systems [64], keeping the accuracy and speeding up the calculations. In 2016, they were used to describe the temporal dynamics of wave packets in turbulent jets, replacing nonlinear models with linear ones for control purposes [65].

In 2018, DL was applied to aircraft airfoils under varying Reynolds numbers [66]. That same year, Neural Networks (NN) were used in fluid mechanics, in the optimization of combustion for coal-fired boilers [67]. Long Short Term Memory (LSTM) networks were also applied to high-dimensional chaotic systems, achieving better results than Gaussian processes in short-term predictions [68]. Also, Generative Convolutional Neural Networks (GCNN) were successfully used to efficiently synthesize fluid simulations, reconstructing velocity fields up to 700 times faster and compressing data 1300 times more than traditional simulations [69]. In that same year, Generative Adversarial Networks (GAN) were implemented to generate high-resolution, temporally coherent fluid simulations from low-resolution data, using a temporal discriminator to maintain smooth transitions [70].

In 2019, frameworks were developed to generate turbulence simulations without sacrificing performance in other flow metrics, aiming to improve computation times while maintaining a high level of accuracy [71, 72]. These frameworks were typically applied to the Navier-Stokes equations under variations of the average Reynolds number [73]. The same year, the use of GANs were applied to generate turbulent flow inlet conditions, achieving flow fields statistically similar to those obtained through Direct Numerical Solution (DNS) across different Reynolds numbers, without the need for additional simulations [73]. In 2020, a physics-informed framework was proposed using deep autoregressive networks with physical constraints to model PDEs, as an alternative to traditional numerical methods without requiring large training datasets, while quantifying the uncertainty in predictions [74].

Also in 2020, different methods applying Graph Neural Networks (GNN) to solve surrogate models appeared. They approximate differential operators in mesh space to model the internal dynamics of physical systems, and in Euclidean world-space to estimate external dynamics like contact and collision. These methods predict the dynamics of a wide range of physical systems, including cloth, structural mechanics, and fluid dynamics, running significantly faster than traditional numerical methods [75].

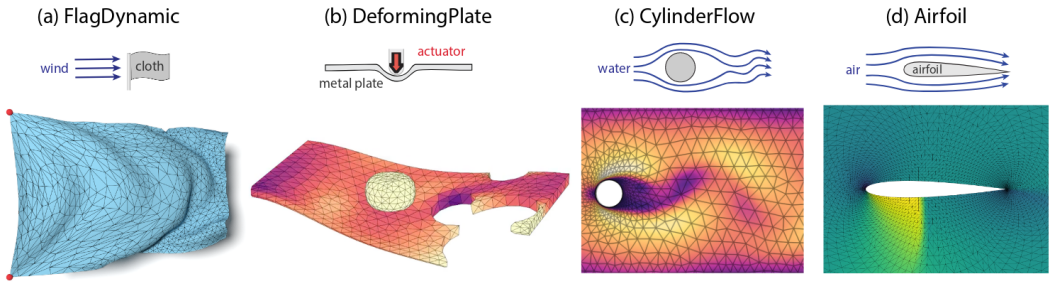


Figure 3.14: Graph Mesh Neural Network on multiple physical systems. Source: [75].

In 2024, a GNN was applied in industrial geometries focused on external aerodynamics for shape optimization, extracting fluid properties from neighboring graphs with low memory consumption, even as the number of layers increases. The methodology was tested on a parametric Navier-Stokes problem, where the parameters control the surface shape of a motorbike [76].

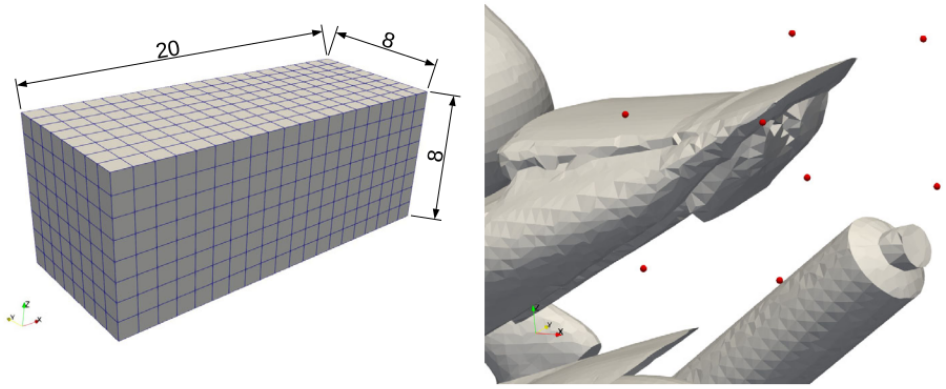


Figure 3.15: Graph Neural Network domain for a motorbike and deformation points for dataset generation. Source: [76].

The progress and research in this field have been achieved thanks to the differentiable nature of physical equations, which has enabled the use of machine learning methods, including gradient propagation throughout the network architecture [77, 10]. However, despite these advances, these methods generally require large amounts of data for training and often do not generalize well to scenarios not included in the training set, limiting their implementation in situations involving real-time applications. As a result, many of these frameworks have begun implementing physical constraints through PINNs, and feeding back input variables in interactive processes [78, 10, 79], which we will discuss later.

3.5.1 Physics Informed Neural Networks (PINN)

In the first formal introduction of PINN, in the two parts of the paper **Physics Informed Deep Learning**, Maziar Raissi, Paris Perdikaris, and George Em Karniadakis formulated a framework for data-driven or solving and discovering partial differential equations using data. Depending on the structure and type of data available, they proposed two types of algorithms: continuous-time and discrete-time models. These neural networks are trained to solve supervised learning tasks while respecting the laws of physics described by general nonlinear partial differential equations, benefiting from domain knowledge in the form of physical models [72]. Depending on whether the available data is scattered in space-time or arranged in fixed temporal snapshots, these two main classes of algorithms are introduced [80].

In essence, PINN combine physics with data, integrating classical numerical techniques and deep learning methods. They are typically applied to PDE or Ordinary Differential Equations (ODE), where a regularizer is added to the loss function based on physical constraints, often in the form of a governing equation [77, 81, 82]. The most common way to create a regularizer is by using the integrated squared residual of the ODE or the PDE. This involves moving all terms to the left-hand side of the equation, squaring the result, and integrating it over the domain. As a result, the loss function takes the following general form [81]:

$$Loss = Data\ loss + Physics\ regularizer\ (Virtual\ loss) \tag{62}$$

Loss : True measure fields from the predicted physics law
Data loss : Predicted output of the physical variable
Physics regularizer : Computes input-output partial derivatives

In fluid mechanics, these methods typically prefer the Eulerian perspective based on mesh descriptions, using CNN. This deep learning method for the Navier-Stokes equations can figure out velocity and pressure fields from simple flow visualizations, without knowledge about shape or boundary conditions [83]. However, as was mentioned previously, these methods trained on pre-generated numerical solutions and do not generalize well to new domains that were not in the training set [73]. The goal of these neural networks is to predict flow field variables from input data such as x, y, z, t, to forecast velocity components u, v, w and pressure p, by learning the mapping from the fluid's position, as shown in the following image [82].

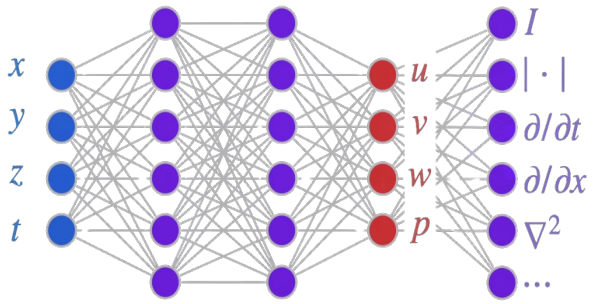


Figure 3.16: PINN general architecture. Source: [82].

PINNs aim to develop a loss function that incorporates known physics using a small dataset, mainly trained on virtual particles instead of large amounts of data [82]. This allows physics to be integrated in an easy and intuitive way through the use of automatic differentiation or backpropagation. In this specific example of the Figure 3.16, we may state that the loss function is provided by:

$$Loss = \underbrace{\sum}_{\text{Data loss}} \|\hat{u}(x_j) - u(x_j)\|_2^2 + \underbrace{\sum}_{\text{Virtual loss}} \|\mathcal{N}(u(x), x_j, t)\|_2^2 \tag{63}$$

Both types of loss only suggest that the physics will be satisfied, but generally cannot fully meet the physical conditions. This may or may not be critical depending on the application. A small, non-zero physics loss is better than none, in other words, the physics laws are not enforced directly; they are promoted through the loss function.

As a result, PINNs may struggle to learn complex physical phenomena due to limited expressiveness and challenges in optimizing the loss landscape. However, these issues can be addressed with the implementation of different architecture or constrained optimization techniques like curriculum regularization or sequential learning, which significantly reduce errors [84]. Length and time scales can lead to conflicting loss functions during training, unbalancing the data and physics loss residuals, this behavior can be addressed by empirically adjusting the loss weights and align them with physically valid solutions [85].

Without requiring previous knowledge of geometry or boundary conditions, these deep learning method for the Navier-Stokes equations infers velocity and pressure fields from basic flow visualizations (such smoke or dye). In areas where direct measurements are impractical, it makes precise predictions in complex problems like blood flow in arteries by utilizing physical rules [83].

CHAPTER 4

Methodology Part I: PINN Model Construction

Contents

This work implemented a PINN-based methodology to simulate hydrodynamic patterns and forces such as drag, lift, and vortices on fully submerged objects, including basic geometric shapes, hydrofoil profiles, and an UVs in Newtonian incompressible isothermal fluids. Following the approach proposed by Wandel et al. (2021) [10], this technique trains a 3D Volumetric Convolutional Neural Network to replicate fluid dynamics using physical regularization built on the continuity and momentum Navier-Stokes equations for both laminar and turbulent flows.

By incorporating physical principles directly into the neural network, this PINNs methodology it is able of real-time fluid simulations without requiring large amounts of prior training data, making them a potential tool for fluid simulation even in non previously seen scenarios. The results are compared to traditional CFD simulations to validate the quantitative and qualitative accuracy of the PINN predictions and ensure consistency between both methods. The first section (4.1), the main aspect of the PINN's mechanism, numerical calculations and the Loss function, the second part will cover volumetric CNN used (4.2), the third section (4.3) the dataset construction, the last section the training procedure(4.4).

4.1 Building the PINN model

As discussed in Chapter 2, the Navier-Stokes equations approximate fluid behavior. In our case, we assume the fluid is incompressible, as is typical with water. We use an Eulerian description, observing the fluid at fixed points in space rather than following individual particles. The fluid is represented by a velocity vector field $\vec{U}$ and a scalar pressure field p over a domain Ω, and its behavior over time is governed by the continuity and momentum equations, as outlined in Section 2.4.

4.1.1 Boundary conditions

Every fluid can be described within a finite volume, and the boundaries of these spaces are defined by boundary conditions. In our case, both the fluid boundaries and the surface of the fully submerged object will have **no-slip** boundary conditions (**Dirichlet boundary condition**) [86]. This condition states that the fluid velocity at the surface of the submerged object is equal to the velocity of the submerged object. In our case, meaning that the fluid velocity at the surface of the object is equal to 0, due we will replicate an external flow over a stationary body.

4.1.2 Ensuring Incompressibility via Vector Potential Decomposition

Any continuous vector field can be decomposed into a curl-free part and a divergence-free part, according to the Helmholtz Theorem [87].

$$\vec{v} = \nabla q + \nabla \times \vec{a} \tag{64}$$

Where ∇q is a curl-free potential field, meaning that the fluid does not tend to rotate around a point, connoting an absence of vortices generated by the field itself. This type of flow is known as irrotational, although it is only valid in regions without vorticity. The fluid will only develop vortices (rotation) when it encounters an obstacle or some type of discontinuity in the boundary conditions. In this case, the following holds:

$$\nabla \times (\nabla q) = 0 \tag{65}$$

When $\nabla \times \vec{a}$ is divergence-free, it means that the velocity field ensures that there are no changes in the local volume of the fluid, i.e., the fluid is incompressible.

$$\nabla \cdot (\nabla \times \vec{a}) = 0 \tag{66}$$

A common approach is to solve the Poisson equation, which guarantees that the projection will not diverge. Instead, we employ a direct prediction of the vector potential to minimize computational demands. This process automatically meets the necessary conditions within the domain, even if it means a possible reduction in accuracy.

$$\vec{v} = \nabla \times \vec{a} \tag{67}$$

4.1.3 Numerical Modeling of Fluid Dynamics with MAC Grids

Being an Eulerian description, one of the most common ways to represent it is using MAC (Marker-And-Cell) cells, which is a way to discretized fluid domain divided into cubical cells of the same size. This way, a grid can be created where pressure values are stored at the center of each cell, while the velocity components are stored perpendicular to the cell faces, and the components of the vector potential can be stored on the cell edges (Figure 4.1).

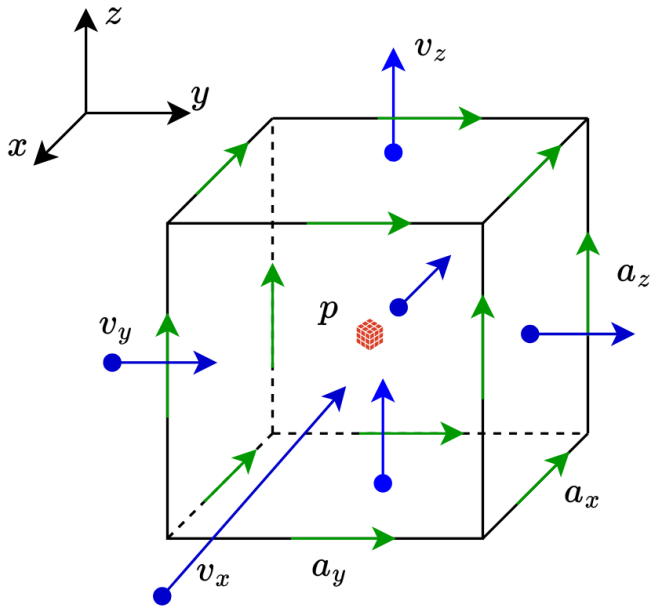


Figure 4.1: Pressure (Red cube), Velocity (blue arrows), vector potential (green arrows) in a MAC Cell. Adapted from: [10].

To compute the velocity field $\vec{v} = \nabla \times \vec{a}$ from the vector potential $\vec{a}$ on a 3D MAC grid, the curl is calculated as follows:

$$\begin{bmatrix} (v_x)_{i,j,k} \\ (v_y)_{i,j,k} \\ (v_z)_{i,j,k} \end{bmatrix} = \begin{bmatrix} (a_z)_{i,j+1,k} - (a_z)_{i,j,k} - \{(a_y)_{i,j,k+1} - (a_y)_{i,j,k}\} \\ (a_x)_{i,j,k+1} - (a_x)_{i,j,k} - \{(a_z)_{i+1,j,k} - (a_z)_{i,j,k}\} \\ (a_y)_{i+1,j,k} - (a_y)_{i,j,k} - \{(a_x)_{i,j+1,k} - (a_x)_{i,j,k}\} \end{bmatrix} \tag{68}$$

The divergence of the velocity field $\vec{v}$ can be computed as follows:

$$\begin{aligned} \nabla \cdot \vec{v}_{i,j,k} &= 0 \\ (v_x)_{i+1,j,k} - (v_x)_{i,j,k} + (v_y)_{i,j+1,k} - (v_y)_{i,j,k} + (v_z)_{i,j,k+1} - (v_z)_{i,j,k} &= 0 \end{aligned} \tag{69}$$

To model the diffusion of the fluid velocity, the Laplace operator Δ is used. This operator employs a 27-point stencil, meaning each node is surrounded by 26 neighboring nodes in the (x, y, z) directions, forming a cube. This approach provides a precise and isotropic estimate (equal in all directions), without favoring any particular direction. The momentum equation, which describes the conservation of mass, has the following form in terms of the time derivative of velocity:

$$\varrho \left(\frac{\vec{v}^{\,t+dt} - \vec{v}^{\,t}}{dt} + \left(\vec{v}^{\,t'} \cdot \nabla \right) \vec{v}^{\,t'} \right) = -\nabla p^{\,t+dt} + \mu \Delta \vec{v}^{\,t'} + \vec{f} \tag{70}$$

To compute the velocity field $\vec{v} = \nabla \times \vec{a}$ from the vector potential $\vec{a}$ on a 3D MAC grid, the curl is calculated explicitly, while we employ an implicit-explicit (IMEX) scheme for the treatment of diffusion terms to ensure numerical stability. This method is a combination of both implicit and explicit methods. Here, the velocity at the intermediate time step, $\vec{v}^{\,t'}$, is the average of the current and future velocities:

$$\vec{v}^{\,t'} = \frac{\vec{v}^{\,t} + \vec{v}^{\,t+dt}}{2} \tag{71}$$

Note: The meaning of all the variables can be found in the Nomenclature section .

4.1.4 Fluid Model

The model, F built, takes as input the **vector potential** $\vec{a}^t$ and the **pressure field** p^t at a given **time** t and predicts their future state at a time $t+dt$. This prediction depends on the **boundary conditions** $\vec{v}_d^{\,t+dt}$, **fluid parameters** like viscosity (μ) and density (ϱ), and the spatial domain (Ω) at the time $t+dt$. Mathematically, the model is described as:

$$(\vec{a}, p)^{t+dt} = F[(\vec{a}, p)^t, \Omega^{t+dt}, \vec{v}_d^{\,t+dt}, \mu^{t+dt}, \varrho^{t+dt}] \tag{72}$$

The model gathers **features** of the fluid via the model inputs, viscosity, and density logarithms through the time. By continuously applying F to the initial state, the fluid simulation progresses to simulate the future state over numerous time steps. Feature representations are:

$$\text{Features} = \left[p^t, \vec{a}^t, \nabla \times \vec{a}^t, \Omega^{t+\Delta t}, \partial\Omega^{t+\Delta t}, \Omega^{t+\Delta t} \cdot \nabla \times \vec{a}^t, \Omega^{t+\Delta t}\dot{p}^t, \partial\Omega^{t+\Delta t}, \vec{v}_d^{\,t+\Delta t}, \ln(\mu^{t+\Delta t}), \ln(\rho^{t+\Delta t})\right] \tag{73}$$

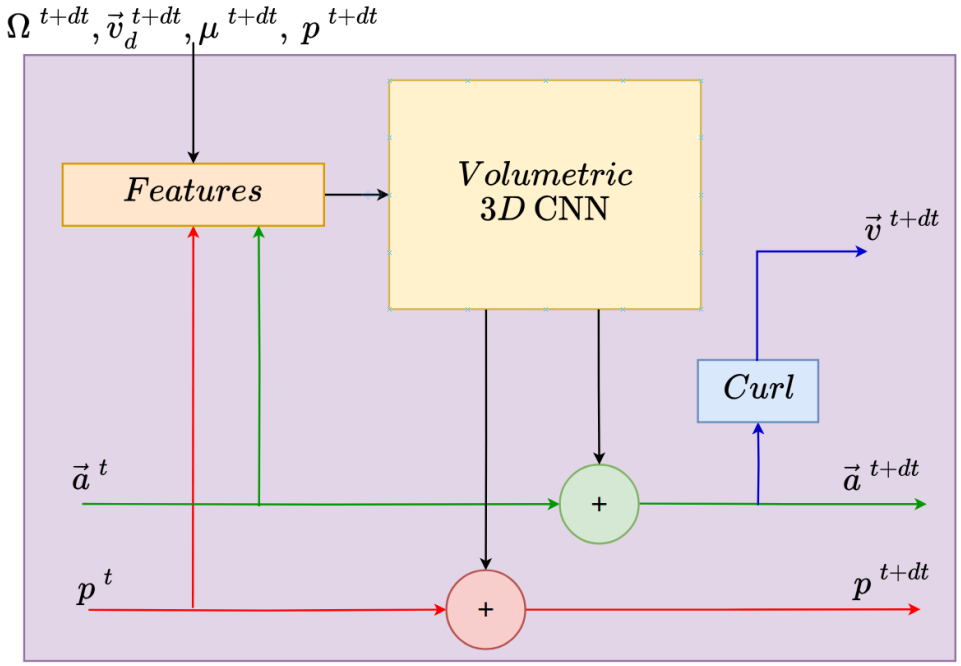


Figure 4.2: PINN structure for a fluid model. Adapted from: [10].

4.1.5 Physics Informed Loss Function

Our loss function is composed of two terms. The first term enforces the momentum equation 2.4.3, continuity 2.4.2, and incompressibility condition. Incompressibility is ensured via the vector potential formulation 4.1.2 and is based on the residuals of the Navier-Stokes equations. The following term for **momentum loss** L_p is derived from that.

$$L_p = \left\| \varrho \left(\frac{\vec{v}^{\,t+dt} - \vec{v}^{\,t}}{dt} + \left(\vec{v}^{\,t'} \cdot \nabla\right) \vec{v}^{\,t'} \right) + \nabla p^{t+dt} - \mu \Delta \vec{v}^{\,t'} - \vec{f} \right\|^2 \text{ in } \Omega \tag{74}$$

The second loss term ensures compliance with the no-slip boundary conditions, which is enforced through the **boundary loss** L_b term:

$$L_b = \left\| \vec{v}^{\,t+dt} - \vec{v}_d^{\,t+dt} \right\|^2 \text{ on } \partial\Omega \tag{75}$$

The Compound loss function is obtained by combining the momentum and boundary loss terms, weighted by hyperparameters α and β:

$$\mathcal{L} = \alpha L_p + \beta L_b \tag{76}$$

In this work, the coefficients $\alpha = 1$ and $\beta = 20$ were chosen to prioritize the boundary loss term L_b over the momentum loss L_p, reducing the tendency of the fluid flow to penetrate the boundaries.

4.2 Volumetric Convolutional Neural Networks

This section discusses two variants of 3D volumetric CNN used for feature extraction in the context of solving fluid dynamics problems using neural surrogate models:

- **U-Net architecture**: A standard 3D U-Net that includes several convolutional and pooling layers. This architecture is widely recognized for its effectiveness in capturing spatial hierarchies and is utilized for more accurate simulations of fluid dynamics, as demonstrated in various studies where CNNs significantly improved prediction accuracy and speed compared to traditional methods [88].

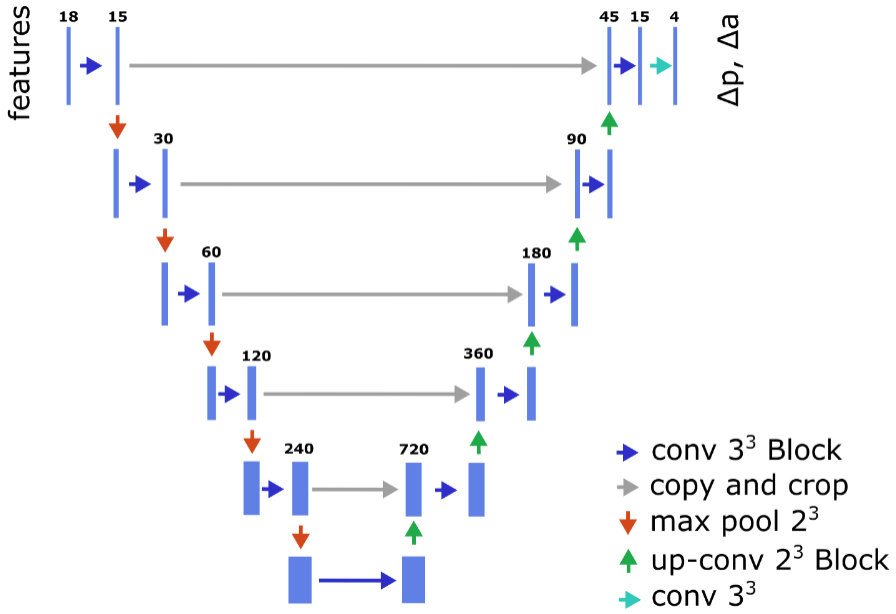


Figure 4.3: U-Net CNN architecture. Source: [10].

- **Pruned U-Net**: A simplified version of the U-Net, where some layers and pooling stages are removed. This variant, while less accurate, offers enhanced computational efficiency, making it suitable for real-time applications where speed is crucial. Research indicates that such pruned architectures can still maintain reasonable performance levels while reducing computational costs.

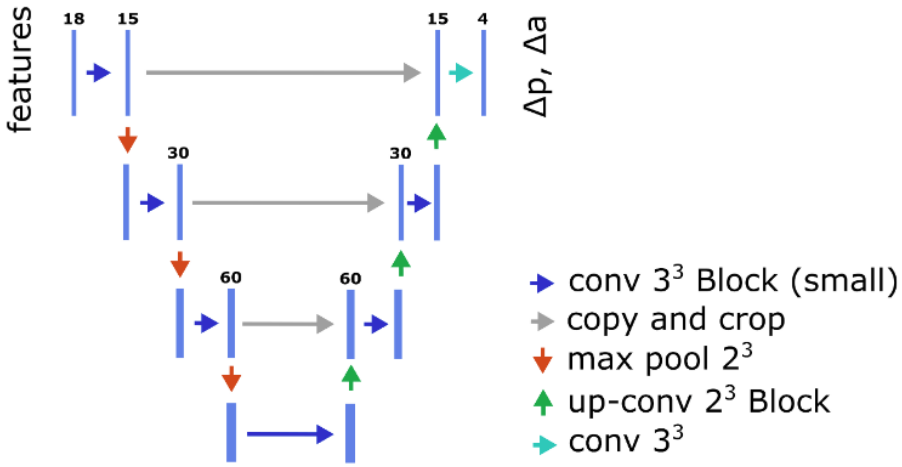


Figure 4.4: Pruned-UNet CNN architecture. Source: [10].

The goal of implementing these architectures is that the depth of the U-Net allows it to represent better complex features, such as surface discontinuities or vortex shedding, while the compact structure of the Pruned-UNet allows for faster inference times.

4.3 Building the Dataset

We used the volumetric CNN domain from [88, 10] with a resolution of $128 \times 64 \times 64$ voxels. The domain was divided along the x-axis into eight sections, each 16 voxels long, where each Volumetric tridimentional unit (voxel) represents 0.0625 m (6.25 cm), so 16 voxels equals 1 meter. For the purposes of this work, we will follow the approach made by Zhi-Hua Liu et al. (2010) [89].

As a result, taking as reference the largest test shape, the UV, the characteristic length L is 32 voxels (2 meters). Distances to the **inlet wall** and **outlet wall** were set to L and $2L$, respectively, with L as the distance to the lateral, top, and bottom walls. This ensures proper flow development, as explained in Section 5.2, and aligns with the CFD baseline.

Cube Figure

Given that the shapes handled by our PINN architecture are simplifications or abstractions formed by voxels, it is unable to accurately represent curves, angles, and surface contours without losing detail. As a result, the PINN only allows us to work with low-resolution objects. Consequently, it becomes necessary to focus on primitive structures composed entirely of cubes (voxels), where the figure with the highest fidelity in its representation is the cube. For our work, the cube will serve as a benchmark to evaluate the convergence of results, calculation speed, flow behavior, and hydrodynamic variables. Thus, if we take the side of our cube as $S = 1\,\text{m}$ for the CFD simulation, the side of the cube in our PINN architecture will be represented by $S = 16$ voxels, as can be seen in the Figure 4.5.

Flat plane

The magnitude of both pressure drag and friction drag depends on the geometry of the object and the flow direction. For a flat plane with its wide side perpendicular to the flow, flow separation occurs easily due to the blunt body shape. As a result, the pressure drag is large, while the friction drag is almost negligible, since the shear stresses are not aligned with the drag direction. When the flat plane is oriented at 90 degrees to the flow, the drag coefficient is not significantly affected by whether the flow is laminar or turbulent.

If the height of the flat plane is aligned with the flow, the pressure drag decreases, while the friction drag increases. However, the drag coefficient decreases as the Reynolds number increases, although it begins to rise again after the transition to turbulent flow. In our case, the dimensions of the flat plane for the PINN are $w, h, d = (16, 3, 16)$ voxels, and for the CFD simulation, the corresponding dimensions are $w, h, d = (1, 0.1875, 1)$ meters, as can be seen in the Figure 4.6.

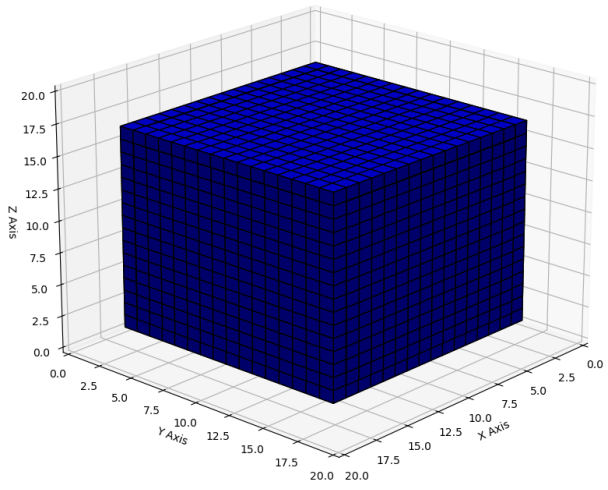


Figure 4.5: Voxelized Cube.

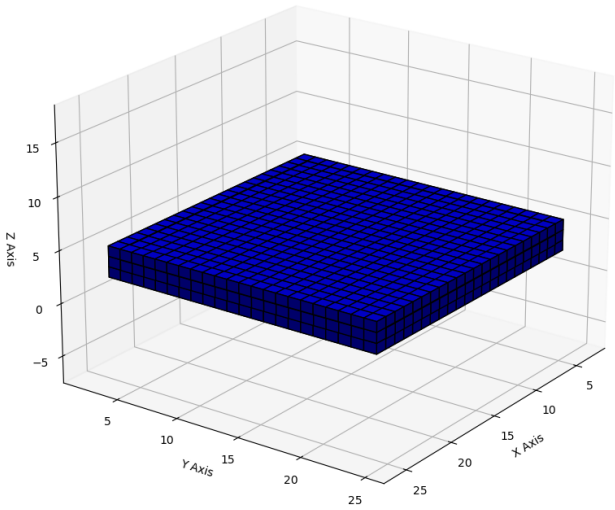


Figure 4.6: Voxelized Flat Plane.

4.3.1 Complex Shape building

Although tools like Blender, free platforms such as VAMtools [1], or the algorithm proposed by Wandel et al. (2020) for the architecture of our PINN [10] include methods for voxelizing complex figures in Standard Triangle Language (STL), these tools often generate non-uniform voxel sizes or overly simplify the overall shape. This makes their use in our PINN architecture impractical, as important features of the figure, such as curvatures, are often lost.

For this reason, it is necessary to manually build the figures. A voxelized figure is built layer by layer, where each layer contains information about the contour of the figure. Each layer is represented by a 2D binary matrix [0,1], where zero indicates the absence of a voxel, and one indicates the presence of a voxel.

To construct these figures, a larger minimum diameter (i.e., a higher number of voxels for the smallest curvature) enhances detail representation. Through experimentation, we determined that the minimum size necessary to represent a discernible curvature with minimal detail is $D = 5$ voxels or 31.25 cm, as shown in Figure 4.7.

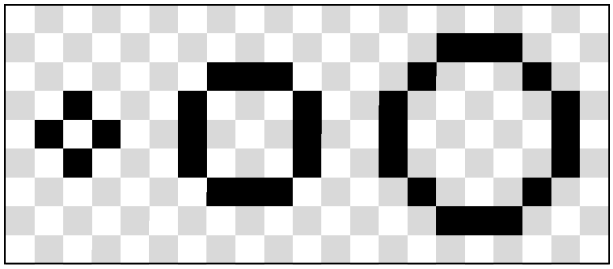


Figure 4.7: Visualization of circumferences of D 3, 5, and 7 voxels respectively.

To assemble the shape of the UV, we implemented a digital art tool Pixelart [2], which allows for the visualization of each layer of the figure. As explained earlier, the minimum diameter for a shape is $D = 5$ voxels. This diameter is used as a reference to scale the model in the PINN, where 22 voxels are equivalent to 1 meter. This creates a conversion ratio, as shown in Figure 4.8, which follows a similar shape configuration to the Sparrus II torpedo-shaped UUV[3].

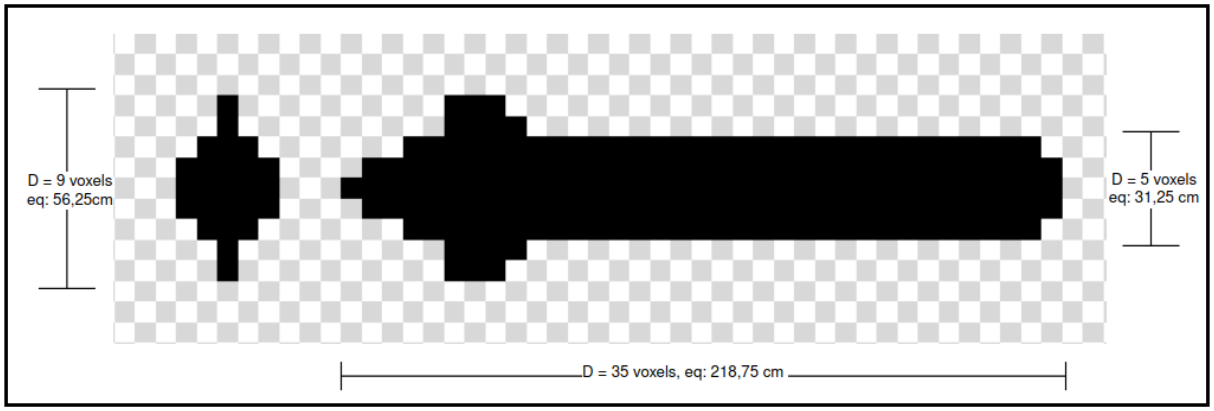


Figure 4.8: UV voxelized layer, showing the front and top views respectively.

Note: The repository containing all voxelized shapes and scripts can be found at the link.[4].

[1] https://vamtoolbox.readthedocs.io/en/latest/_docs/examples/voxelizestl.html
[2] https://www.pixilart.com
[3] https://iquarobotics.com/sparus-ii-auv
[4] https://github.com/YosefGuevara012/underwater_CFD_data/tree/master/training_files

Sphere Figure

The importance of the sphere lies in its ability to allow us to compare consistent behaviors regarding the drag coefficient C_D. For a sphere, C_D behaves linearly when the Reynolds number is less than one, meaning that no flow separation occurs, and it is defined by:

$$C_D = \frac{24}{Re} \tag{77}$$

This behavior is described by **Stokes' law**, which provides an analytical solution for spheres when $\mathrm{Re} < 1$. In this regime, the drag force F_D is given by:

$$F_D = 3\pi\mu V D \tag{78}$$

Although it is not possible to generate a perfectly smooth surface using voxels, approximations of smooth surfaces can be made if the figure is large enough to represent them adequately. In the case of a sphere, we used the software [5], which allows the breakdown of each layer into voxels. The diameter of our sphere was set at $D = 16$ voxels, equivalent to 1 meter.

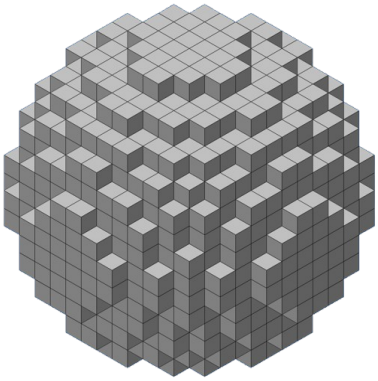


Figure 4.9: Sphere of D = 16 voxels.

Figure 4.10: Sphere of D = 1 m.

NACA 0018 Profile Shape

Although submarine fins **(diving planes)** are not required to generate lift or any other specific hydrodynamic reason. In marine robotics, symmetrical hydrofoil profiles are commonly employed, likely due to their ease of **manufacture**. The **hydrofoil** or transversal section of the **fin** are typically based in their airfoil counterparts. In underwater vehicles, they are implemented to help the submarine move at different depths or maintain neutral buoyancy. However, it is known that due to the reduction in pressure caused by the increase in velocity 2.4.1, a phase change in the fluid can occur, leading to cavitation. This effect typically begins at the intersection between the fin and the body of the vehicle [90].

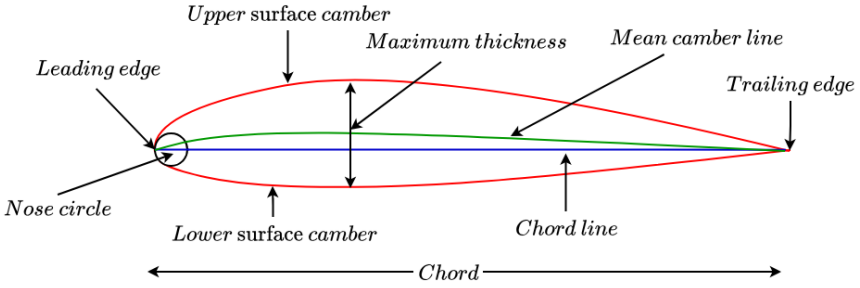


Figure 4.11: Elements of a hydrofoil. Adapted from: [13].

For this work, a symmetric profile was chosen based on the NACA 0018 airfoil, with a chord length of 136,4 m or 22 voxels and a span of 1 meter or 16 voxels, to allow the curvature approximation. This choice was partly motivated by previous experiments conducted on the same profile [91, 92]. The airfoil was designed using the Airfoil Tools platform [6] and the Airfoil Tools add-on for Blender [7].

[5] https://www.plotz.co.uk/plotz-model.php?model=Sphere
[6] http://airfoiltools.com/airfoil/details?airfoil=naca0018-il
[7] https://github.com/Fritkot99/AirfoilToolsBeta

Figure 4.12: NACA0018 side view.

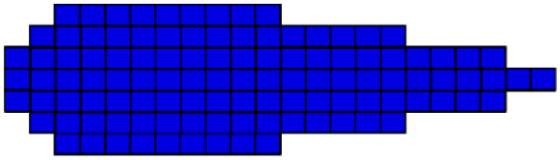


Figure 4.13: Voxelized NACA0018 side view.

Since the hydrofoil profile used is a NACA 0018 airfoil, the camber distance is equal to 18% of the chord line. For a chord length of 22 voxels, this results in a camber distance of 3.96 voxels.

Torpedo-Shaped UV

Although the sphere is known for its optimal pressure and added mass distribution over a hull, the torpedo shape is one of the most widely used hydrodynamic forms. This geometry is common in many types of UV, such as torpedoes, gliders, UUV , Autonomous Underwater Vehicle (AUV), and submarines, due to its ability to minimize drag and provide stability during movement through water. In our study, we utilize a torpedo shape UV, with a hull diameter with the characteristics explained in Figure 4.14, to look closer to a glider structure distribution.

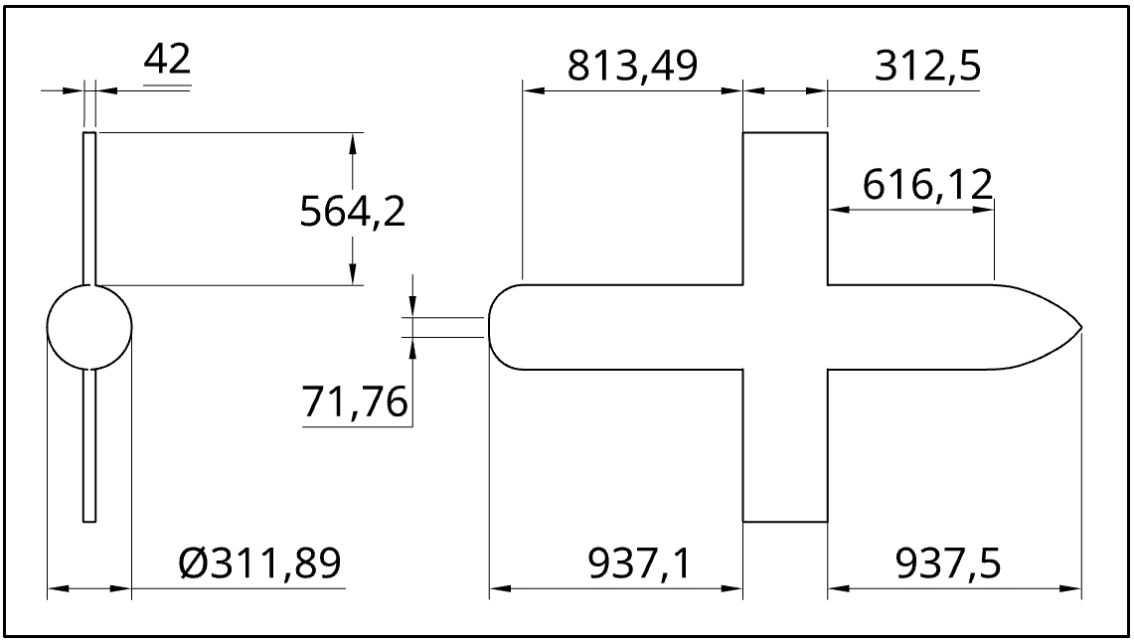


Figure 4.14: Torpedo-Shaped UV measurements in (mm).

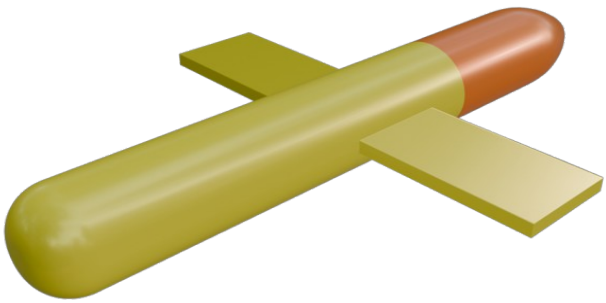


Figure 4.15: Torpedo-shaped UV.

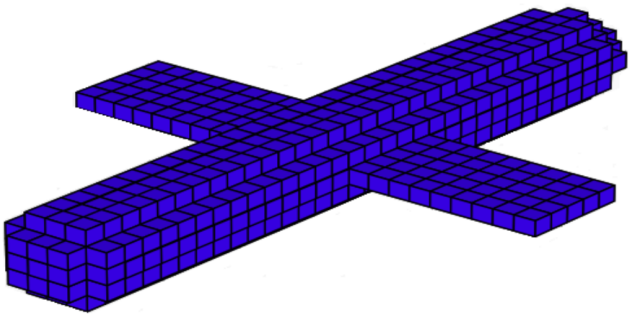


Figure 4.16: Voxelized torpedo-shaped UV.

4.4 Training process and characteristics

The training process was conducted on both the **pruned U-Net** and **U-Net** architectures, following the method described by Wandel et al. (2020) [10]. The model was trained for **1000 epochs** with a **batch size of 10** and a **learning rate of 0.0005**, where the **training pool** assigns a unique, random viscosity (μ) and density (ϱ) to each environment. A batch in compse of a series of environments and submerged objects, each with its corresponding vector potentials ($\vec{a}$), pressure field (p), viscosity (μ) and densities (ϱ) . To balance stability and accuracy during the simulation, we employed the **IMEX (implicit-explicit)** integration scheme. This scheme is particularly suitable for stiff systems, where the implicit part is used for the stiff terms to ensure stability, while the explicit part is applied to non-stiff terms to maintain computational efficiency. After extracting features, the CNN output is *mean-normalized* to fix any drifting offsets in the vector potential $\vec{a}$ and pressure p. This adjusted output is then added to the previous state to predict the fluid's future state.

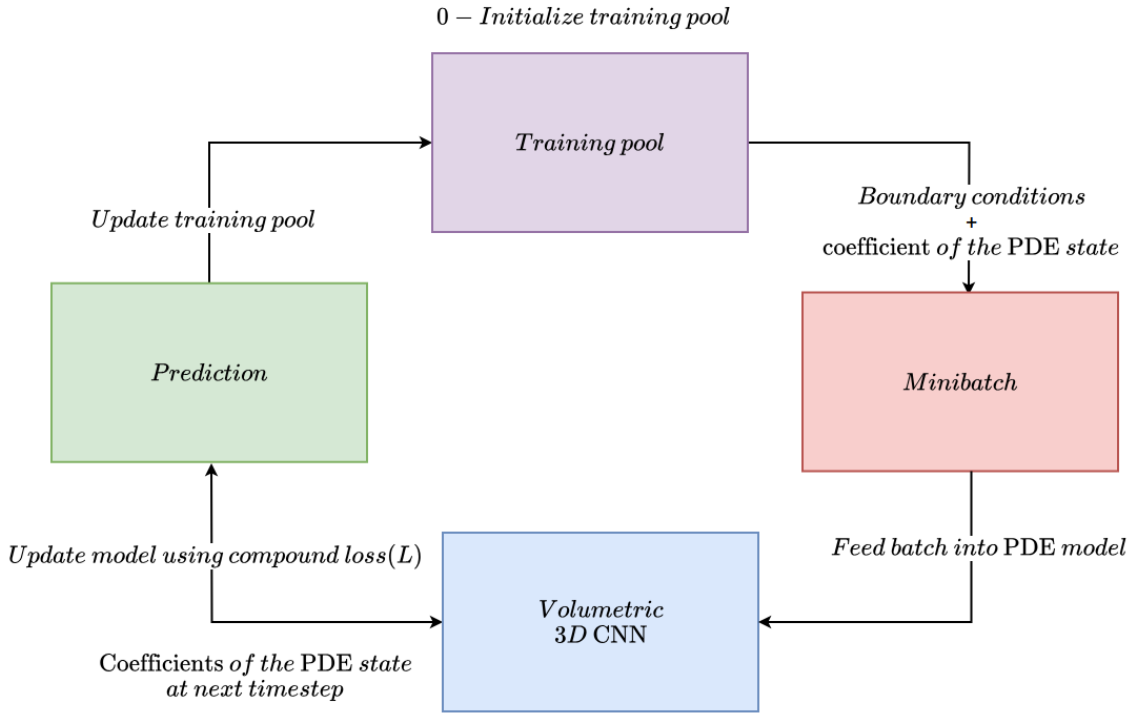


Figure 4.17: PINN methodology training cycle. Adapted from: [79].

The training dataset consisted of various shapes and fluid phenomena, generated in the methodology described in Section 4.3. This included shapes such as a **cube**, a **hydrofoil**, a **flat plane**, and a torpedo-shaped **UV**, all simulated in surge (x-direction) and heave (z-direction) relative to fluid motion. The dataset was defined on a **128x64x64** grid, with viscosity (μ) and density (ϱ) values randomly sampled within the ranges [0.01, 8] $Pa \cdot s$ and [0.1, 15] kg/m^3, respectively, to simulate varying fluid dynamics. Each simulation averaged 5000 time steps, with a time step size $\mathbf{dt} = \mathbf{4}$ representing the state of the system at a future time point or how far forward in time the simulation progresses during one iteration. The general training cycle for the PINN is illustrated in Figure 4.17.

The training process does not evaluate any specific convergence value, as no validation reference is provided during training, besides the compound loss function $\mathcal{L}$ value, mentioned in 7.1. Consequently, no stopping criterion is implemented, given the absence of an external indicator to halt training before completing all epochs. In this setup, the model learns from its interaction with the training data, adjusting its parameters to minimize the loss function.

Note: The repository containing all scripts for the PINN methodology can be found at the link.[8].

[8] https://github.com/YosefGuevara012/Teaching_Incompressible_Fluid_Dynamics_to_3D_CNNs

CHAPTER 5

Methodology Part II: CFD Baseline Construction

Contents

The CFD simulations serve as a proven reference for solving the incompressible Navier-Stokes equations under various fluid conditions, providing a reliable numerical solution for building a valid baseline. The significant computational costs of these simulations require careful strategies to balance accuracy and resource efficiency. In this thesis, OpenFOAM was used to perform the CFD baseline simulations, as detailed in Section 5.1. The fluid domain setup and boundary conditions, which ensure the proper development of the flow, are discussed in Section 5.2. Lastly, the turbulence model used, essential for simulating more complex flow behaviors, is outlined in Section 5.3.

5.1 Software and Tools for CFD Simulation

To compare the results of our PINN architectures, a baseline was built using CFD simulations in **OpenFOAM 8.0**[1]. OpenFOAM is a widely recognized tool in fluid dynamics simulations. We simulated multiple scenarios under different fluid conditions, varying viscosity (μ) and density (ϱ), to model laminar, transitional and turbulent flows using typical marine shapes, such as spheres, cubes, and torpedo shape UV, under a fixed flow speed u .

To ensure consistency, both the PINN and CFD simulations were conducted with no-slip boundary conditions on the object surfaces and fluid domain boundaries, simulating the hydrodynamic behaviors inside a fluid domain constrained by barriers, similar to a wind tunnel. The geometries used were designed in **Blender**[2] and exported in STL format. For this baseline, OpenFOAM and the **Paraview 5.6.0**[3] visualizer were used on **Ubuntu 20.04**[4]. The shapes used were generated in Blender and exported in the STL, which ensures compatibility with various simulation environments.

Note: The repository containing all scripts for the CFD simulations be found at the link.[5].

5.2 Mesh Building

5.2.1 Domain Size

The domain size in CFD simulations is especially important because it directly influences the accuracy of the results. The domain must be large enough to allow the flow to stabilize and develop properly before reaching the boundaries. The boundary effects may be altered if the domain is too small, leading to unwanted interactions between the flow and the domain walls. This is particularly important in turbulent flows, where fluctuations can extend beyond the immediate region around the object.

In this work, we adopt the approach of Zhi-Hua Liu et al. (2010) [89], similar to how we set up the PINN domain. The characteristic length L for the torpedo-shaped UV is rounded to $L = 2\,\text{m}$, representing the maximum object length used in the simulations. The distance between the object and the inlet is L, while the distance to the outlet is $2L$, with the object placed L from the lateral and upper boundaries. As a result, the fluid domain spans an area of $8 \times 4 \times 4\,\text{m}$.

In OpenFOAM, the fluid domain size is generated using the **blockMesh** utility, which creates one or more 3D hexahedral blocks (in our case, cubes). The script was set to produce a maximum of **524,288 hexahedral blocks**, corresponding to a $128 \times 64 \times 64$ grid. This grid is consistent with the PINN architecture, which has the same number of voxels by default.

5.2.2 Surface Features

In CFD simulations, complex geometries are often simplified to speed up computational times, especially during the early stages of prototyping. However, this should be done with caution, as important details of the shape of the object that could be useful for flow analysis could be lost. In some cases, the symmetry of the problem can be exploited to reduce the domain size, significantly decreasing the computational time and resources required.

[1] https://openfoam.org/version/8/
[2] https://www.blender.org/
[3] https://www.paraview.org/download/
[4] https://ubuntu.com/
[5] https://github.com/YosefGuevara012/underwater_CFD_data/tree/master/OpenFoam8_CFD_simulations

For purposes of this research, the object mesh structure is captured using the **surfaceFeatures** utility. The command was set to identify features when the angle is less than 150°. Although our objects do not have spaces or holes on their surface, we have decided to maintain **openEdges** to allow flexibility in case of future needs that require such characteristics.

5.2.3 Mesh Refinement

The **snappyHexMesh** utility was configured to maintain a minimal level of detail in certain object contours, avoiding excessive subdivision around the object or within the fluid domain. A variable sub-grid was constructed around the object. This sub-grid typically extends 0.5 meters in all directions, depending on the orientation of the object. For example, the possible movement directions of a UV (Figure 5.1), such as longitudinal motion (movement along the z_b-axis) or surge motion (movement along the x_b-axis), can influence the required mesh refinement[6]. Each box in the sub-grid was divided just once to avoid excessive refinement, which could interfere with comparisons between the simulations.

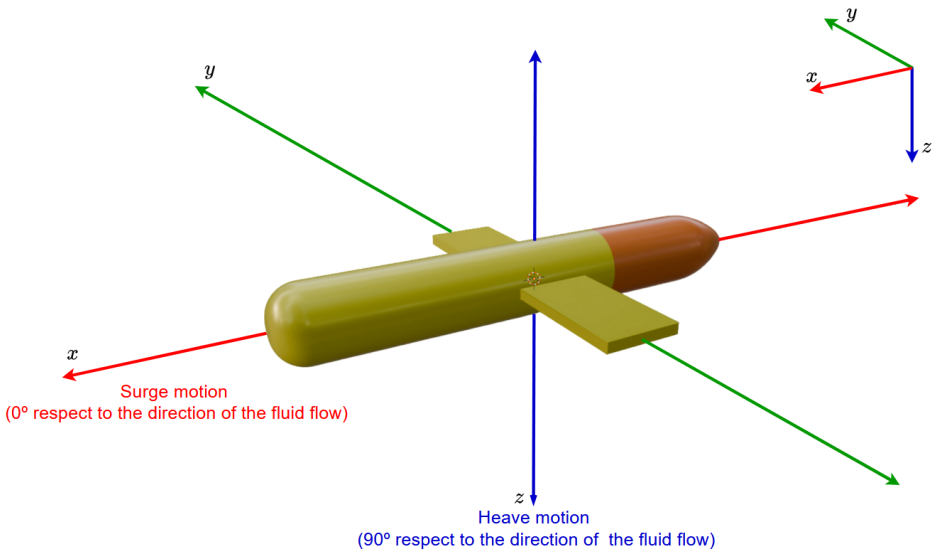


Figure 5.1: Degrees of Freedom (DOF) in an Underwater Vehicle UV. Adapted from: [38].

A result of the mesh building (domain size, surface features, mesh refinement) process is exemplified in the Figure 5.2.

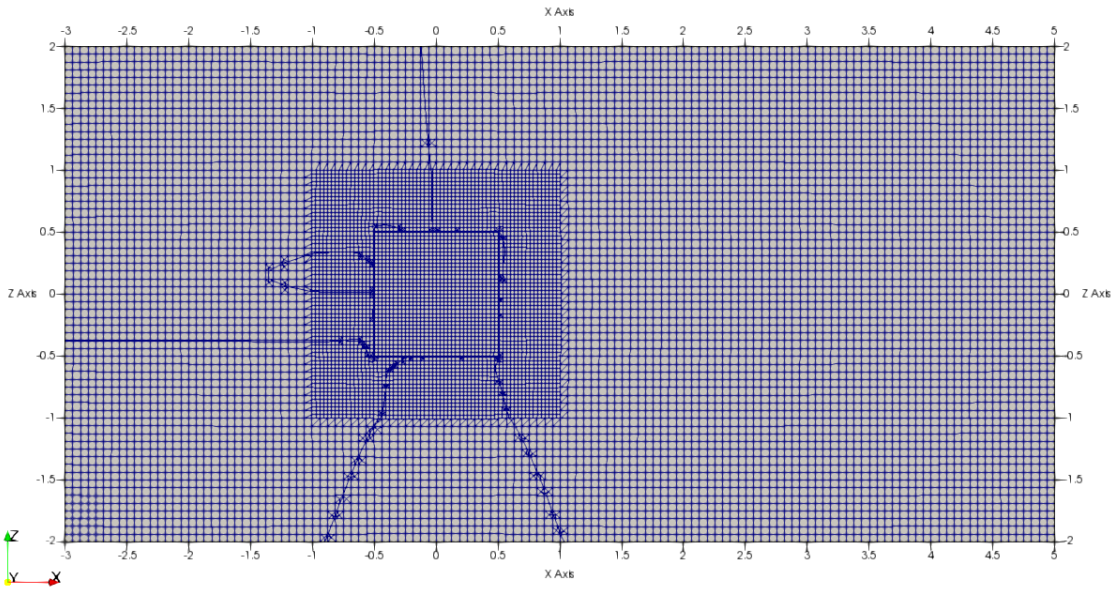


Figure 5.2: Cube Mesh representation for the CFD baseline.

[6]**Note:** In the case of an UV, the heave motion along the z-axis points from the bottom of the submarine towards the seabed.

5.3 Turbulence Model

Although a DNS or Large Eddy Simulation (LES) can accurately determine variables such as lift and drag, these methods are computationally expensive, even for small fluid domains. To reduce computational costs, **turbulence models** are used as an alternative to solving the Navier-Stokes equations at all scales and times. For purposes of building our CFD baseline, we will use the Reynolds Averaged Simulation (RAS) as a framework of equations for turbulence models [93], applying specifically the $k-\omega\,SST$ model [94] for incompressible flows.

5.3.1 Reynolds-Averaged Simulation (RAS)

Also known as Reynolds-averaged Navier-Stokes (RANS), this framework resolves the average field variables, avoiding resolving small fluid fluctuations, focusing on the broader characteristics of the flow. This approach considers the same flow multiple times under the same arbitrary initial conditions, rather than considering an average over a time interval [93].

Field Decomposition:

For that purpose, the flow speed U, is decomposed into an averaged field speed $\overline{U}$ and a field of fluctuations U', according to the following expression:

$$U(t) = \overline{U} + U'(t). \tag{79}$$

5.3.2 Momentum Conservation in RAS

For momentum conservation, the averaged equation takes the form:

$$\frac{\partial \varrho \overline{U}}{\partial t} + \nabla \cdot (\varrho \overline{U} U) = -\nabla \overline{p} + \nabla \cdot (\tau + \tau'), \tag{80}$$

where τ' represents the Reynolds stresses, which are crucial for capturing the effects of turbulences within the simulation.

Reynolds Stresses:

Derived from velocity fluctuations, and represents the effects of turbulence in RAS. These stresses are defined as:

$$\tau' = -\varrho \overline{U'U'}, \tag{81}$$

These stresses model the momentum and energy transfer caused by turbulences. Without a model for them, the averaged flow equations would be incomplete and unsolvable.

5.3.3 The k $-\omega$ SST Turbulence

In this research, this specific approach under the RAS Framework of equations 80. This model introduces transport equations for turbulent kinetic energy ($\mathbf{k}$) $[m^2/s^2]$ and the turbulence dissipation rate (ϵ) $[m^2/s^3]$, under a specific turbulence dissipation rate ω [1/frequency], allowing us to solve for the eddy viscosity μ_T and C_μ is a constant equal to 0.09 [93, 95].

$$\omega = \frac{\epsilon}{C_\mu \mathbf{k}} \tag{82}$$

Transport Equation for k:

The transport equation for turbulent kinetic energy is given by:

$$\frac{\partial(\varrho \mathbf{k})}{\partial t} + \nabla \cdot (\varrho U \mathbf{k}) = \nabla \cdot \left(\left(\mu + \frac{\mu_t}{\sigma_k} \right) \nabla \mathbf{k} \right) + P_k - \varrho \varepsilon \tag{83}$$

where P_k represents the production of turbulent kinetic energy, and ϵ is the dissipation rate of $\mathbf{k}$.

Transport Equation for ω:

The transport equation for ϵ is defined as:

$$\frac{\partial(\varrho\omega)}{\partial t} + \nabla \cdot (\varrho U\omega) = \nabla \cdot \left(\left(\mu + \frac{\mu_t}{\sigma_\omega}\right)\nabla\omega\right) + \gamma\frac{P_k}{\nu_t} - \beta\varrho\omega^2 \tag{84}$$

where β is an empirical coefficient that varies depending on the specific $k-\omega$ model used.

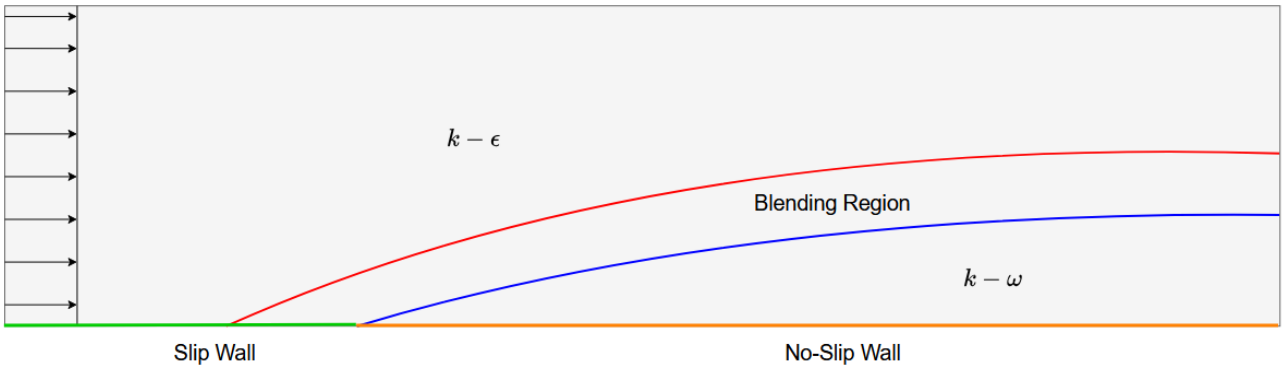


Figure 5.3: The $k-\omega$ SST model blending region between the $k-\epsilon$ and $k-\omega$ turbulence models. Adapted from: [95].

In this model, the **transport equation for** ϵ includes an additional term:

$$+2\frac{\varrho\sigma\omega^2}{\omega}\nabla\mathbf{k} : \nabla\omega \tag{85}$$

The $\mathbf{k}-\epsilon$ model tends to perform well in the free-stream region, while the $\mathbf{k}-\omega$ model performs better in the boundary layer region close to the wall. Because this model blends $\mathbf{k}-\epsilon$ and $\mathbf{k}-\omega$ approaches, this additional term is introduced. By multiplying this by $(1-F_1)$, we can smoothly blend between $\mathbf{k}-\omega$ and $\mathbf{k}-\epsilon$ models:

$$2(1-F_1)\frac{\varrho\omega^2}{\omega}\nabla\mathbf{k} : \nabla\omega \tag{86}$$

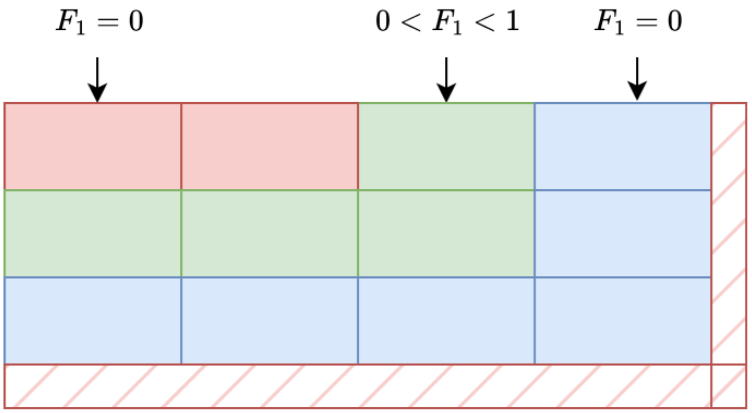


Figure 5.4: The blending function F_1. Adapted from:[95].

$$F_1 = 0, \text{ the model is } \mathbf{k}-\epsilon, \text{ Away from the wall}$$
$$F_1 = 1, \text{ the model is } \mathbf{k}-\omega, \text{ near the wall}$$

For our work, the values are set as $\mathbf{k} = 0.24$ (kinetic energy) and $\omega = 1.78$(specific turbulence dissipation rate), which are typical values for low viscosity fluids such water [95].

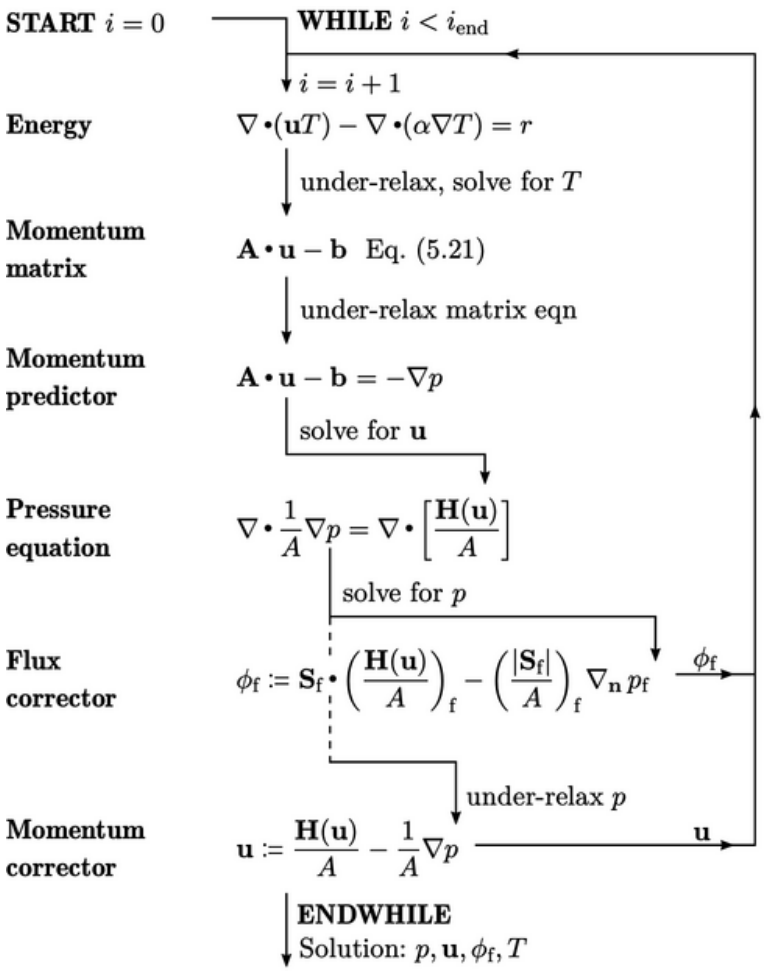
5.3.4 The SIMPLE Algorithm

In the thesis, we applied the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) algorithm decouples the pressure and velocity fields and solves them iteratively [96, 97, 93] towards a converged solution for the $\mathbf{k}-\omega$ **SST** turbulence. This algorithm is primarily used in steady flow simulations, where the flow variables do not change with time.

It operates as an iterative sequence, where each step **i** involves constructing a matrix equation for the energy equation, which is then relaxed, which involves adjusting the values by applying a factor α to stabilize convergence and control the influence of each iteration. After relaxation, the temperature T is solved. The temperature equation is updated based on the equation of state, and then a new equation is constructed, which includes all terms of the momentum equation except the pressure gradient [97, 93].

The steps of the algorithm are detailed below:

Algorithm 1 Solving system for p, u, ϕ_f, T



Note: Algorithm image taken from the book Notes on Computational Fluid Dynamics: General Principles [93].

Note: The SIMPLE algorithm is suitable for representing incompressible Newtonian fluids in a steady state. However, rather than solving the thermal energy equation, the model captures turbulence effects through the energy and dissipation factors represented by k and ω, respectively. In this work, as previously discussed in 2.4.4, thermal effects are neglected due to the incompressible and isothermal characteristics of the fluids considered.

CHAPTER 6

Methodology Part III: Fluid Conditions and Hydrodynamic Variables

Contents

The relationship between the velocity vector $\vec{v}$ and the pressure distribution P has a fundamental role in the analysis of hydrodynamic forces acting on submerged bodies in a fluid, they are essential for understanding the behavior of marine vehicles. These forces determine the stability, efficiency, and maneuverability of objects in fluid environments, and their calculation is crucial for optimizing the design, control, and operation of any UV.

In this section, we describe the methods and procedures to calculate the used to evaluation patterns and hydrodynamic variables to compare the results of our simulations. This description will help us to characterize the fluid behavior and effects of a fully submerged object in a Newtonian incompressible isothermal fluid within a confined environment that replicates typical conditions found in wind tunnel experiments under different fluid conditions, section 6.1. The major hydrodynamic variables: the **Fluid velocity field**, **pressure distribution**, **drag force**, **lift force**, and **vorticity**, in sections (6.2,6.2,6.3.2,6.3.3,6.4) respectively. Finally, the last section 6.5 shows the calculation of these variables on CFD and PINN simulations using Paraview software.

6.1 Fluid Simulation Conditions

Since the units in the PINN are based on voxels, the parameters from the CFD simulations were adjusted accordingly. In this case, 1 meter corresponds to 16 voxels, meaning that 1 voxel represents 6.25 cm. Under this assumption, the initial velocity field will be $U = 1$ m/s and the initial pressure will be 0 Pa.

One of the key aspects in fluid dynamics is the transition between laminar and turbulent flow. Laminar flow is characterized by smooth, orderly fluid motion, while turbulent flow exhibits chaotic and disordered fluctuations. The primary criterion for determining this transition is the Reynolds number, as discussed in 2.3.4, specific for the PINN based methodology, which varies based on changes in viscosity and density to simulate different flow regimes under specific fluid conditions.

Variable	**Symbol**	**Reynolds Number (Re)**		
		Laminar: 0.64	**Transitional: 80**	**Turbulent: 800**
Dynamic Viscosity	$\mu\ (Pa \cdot s)$	5	0.2	1
Density	$\varrho\ (kg/m^3)$	0.2	1	0.5
Kinematic Viscosity	$\nu\ (m^2)$	25	0.2	0.02
Characteristic Length	L (m)	16 Voxel	16 Voxel	16 Voxel

Table 6.1: Fluid variable values for different Reynolds numbers.

6.2 Flow Patterns and Fluid Velocity Field

Flow velocity describes how the fluid moves at different points within the simulation domain. This variable is important for analyzing transitions between flow regimes, such as laminar and turbulent flow, and for detecting flow separation around submerged bodies. Flow separation occurs when the boundary layer of fluid around an object loses adhesion to its surface, creating a low-pressure recirculation zone, typically in the object's wake. This phenomenon often occurs at high angles of attack or with objects that have abrupt geometries.

Tracing streamlines in the flow helps to reveal whether the flow remains attached to an object's surface. When streamlines closely follow the shape of the body (Figure 6.1 (a)), the flow is attached. If the streamlines deviate from the surface (Figure 6.1 (b)), the flow is detached or separated, often leading to an unsteady wake and turbulence.

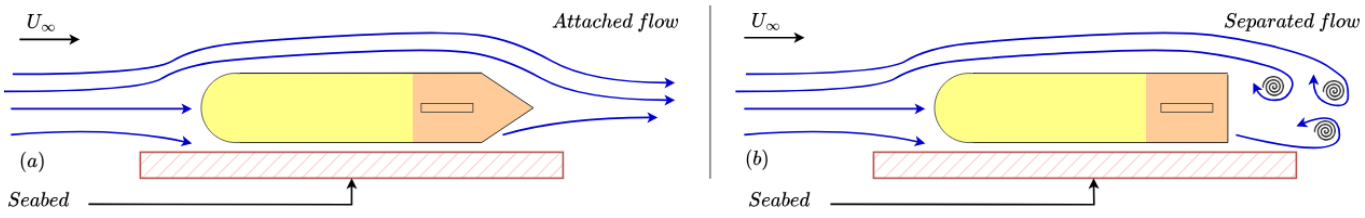


Figure 6.1: (a) Attached flow over a streamlined UV and (b) flow separation behind a modified UV. Adapted from: [98].

Analyzing local vorticity and flow separation, it is essential to understand the velocity distribution throughout the fluid domain. Streamlines are useful for visualizing fluid trajectories and the magnitude of velocity along these paths. In our simulations, streamlines were calculated to assess both the flow velocity magnitude and direction at various points. This analysis helps identify high-velocity areas, which increase drag, and low-velocity areas, which may indicate flow separation or recirculation zones.

Stagnation and Circulation

At the stagnation point, the fluid decelerates completely, causing an increase in static pressure. Streamlines converge, and velocity decreases significantly, resulting in high pressure. The pressure difference between the leading and trailing edges creates circulation around the profile, which is crucial for lift generation.

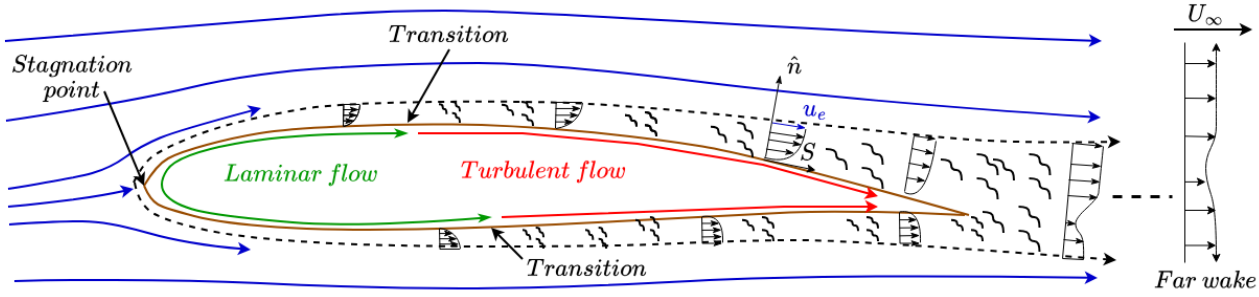


Figure 6.2: Boundary layer and wake development on a typical airfoil, shown by the u(n) velocity profiles. Adapted from: [99].

Fluid Flow and Pressure Distribution

To assess the impact of fluid flow on an object's body, pressure distribution graphs on the upper and lower surfaces are helpful. These graphs illustrate how pressure varies along the profile, particularly around the wing surface. By visualizing the pressure distribution, we can evaluate lift by identifying regions of high and low pressure and understanding their influence on the lift generated by the profile at various angles of attack.

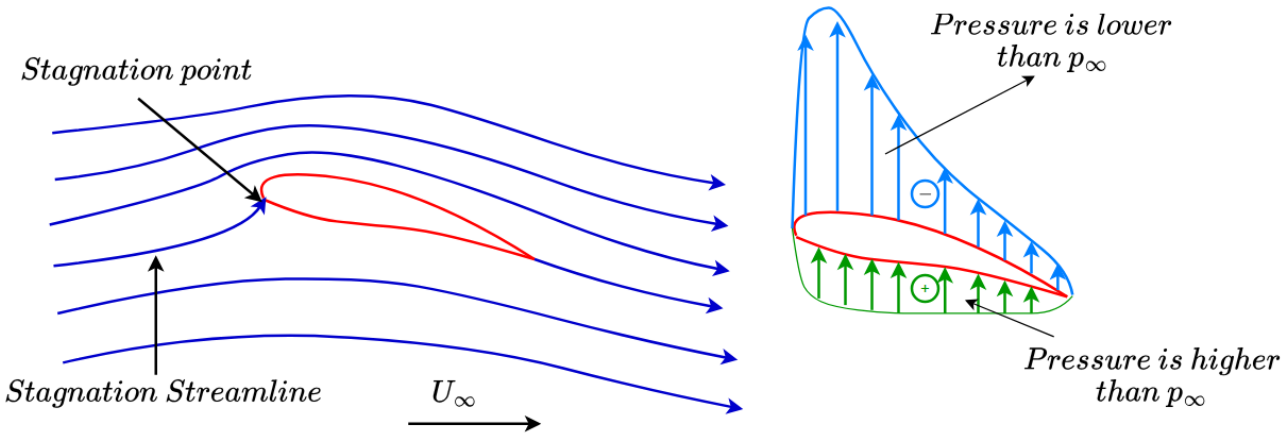


Figure 6.3: Streamlines near an airfoil and the resulting pressure distribution. Adapted from: [11].

6.3 Pressure Field and Near-Field hydrodynamic forces

As the fluid flows along the surface of an object, it generates a pressure distribution across the body, resulting in a Near-Field hydrodynamic forces that can be decomposed into two main components, the **Pressure Force** and **Friction Force**,

$$F = F_{pressure} + F_{friction} \tag{87}$$

$$F_{\text{pressure}} \equiv \iint -p_w \hat{n}\, dS = \iint (p_\infty - p_w)\hat{n}\, dS \tag{88}$$

$$F_{\text{friction}} \equiv \iint \tau_w\, dS, \quad \tau_w = \vec{\tau}_w \cdot \hat{n} \tag{89}$$

Where p_{w} is the pressure strees and $\boldsymbol{\tau}_{\text{w}}$ represent the viscous stress vector acting on a surface element $\mathrm{d}\mathcal{S}$, with unit normal $\hat{\mathbf{n}}$. The substitution of $-p_{\text{w}}$ with $p_\infty - p_{\text{w}}$ in the second pressure integral is justified because the uniform pressure p_∞ does not exert a net force on the body [99]. This follows from the identity.

$$\iint \hat{\mathbf{n}}\, \mathrm{d}\mathcal{S} = 0 \tag{90}$$

The Friction Drag (τ_ω) consists of tangential forces caused by the viscosity of the fluid. It is most relevant for streamlined objects, where the fluid stays attached to the surface for a longer duration. Pressure strees (p_w), on the other hand, arises from pressure differences acting perpendicular to the surface of the object.

This is especially significant for blunt bodies, where flow separation creates a low-pressure region behind the object, as shown in Figure 6.4. The pressure difference between the front and rear generates a net drag force, which increases as the separation region grows, leading to phenomena such as vortex shedding.

6.3.1 Pressure Field distribution visualization

One way to visualize this distribution is using contour plots that show lines of constant values for a property. These lines can represent pressure, velocity, and other flow properties. Contour plots help identify areas of high or low values quickly. They can either display simple contour lines or be filled to highlight different levels more clearly [15].

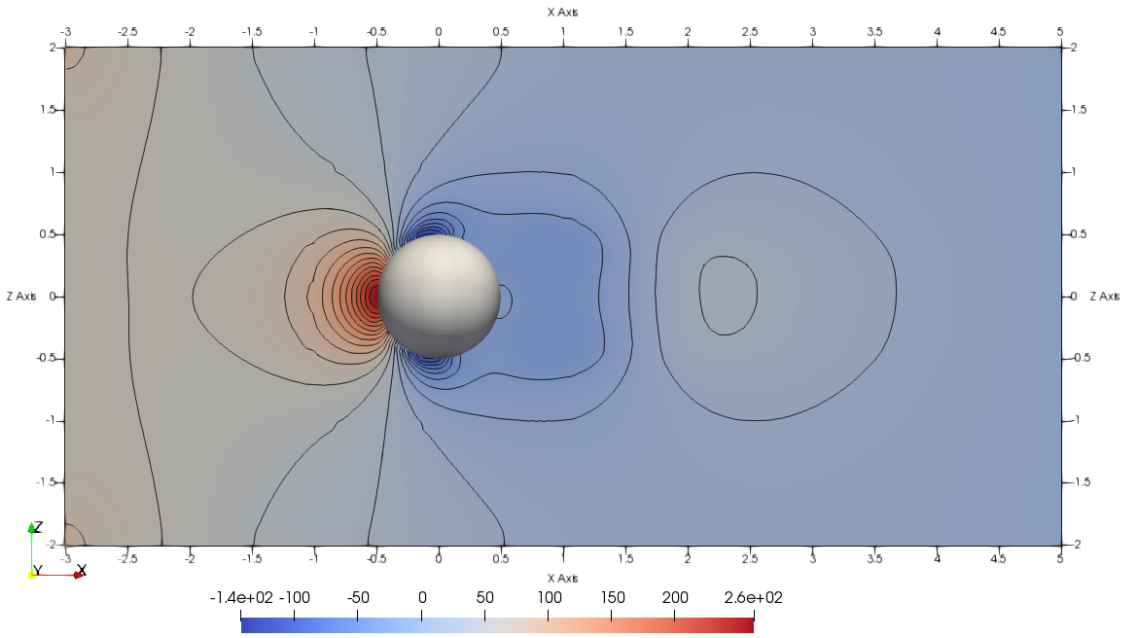


Figure 6.4: Pressure contour distribution around a sphere.

6.3.2 Drag Force

The drag force is the component of the force exerted by the fluid that acts in the same direction as the flow. This force opposes the movement of a vehicle and is therefore used to determine power or speed requirements. Lower drag reduces energy consumption and improves the operational efficiency of the vehicle. When the x-axis is aligned with the freestream, $U_\infty = U_\infty \hat{x}$, the streamwise drag force component is then $F_D = F \cdot \hat{x}$, which has pressure and friction contributions [99].

$$F_D = F \cdot \hat{x} = D_{\text{pressure}} + D_{\text{friction}} \tag{91}$$

$$D_{\text{pressure}} \equiv \iint (p_\infty - p_{\text{w}})\, \hat{n} \cdot \hat{x} \mathrm{d}\mathcal{S} \tag{92}$$

$$D_{\text{friction}} \equiv \iint \tau_w \cdot \hat{x} \mathrm{d}\mathcal{S} \tag{93}$$

The pressure gradient influences this behavior, when the gradient is favorable $\frac{\partial p}{\partial x} < 0$, the pressure decreases in the flow direction, allowing the fluid to remain attached to the object, reducing flow separation. In contrast, when the pressure gradient is increase $\frac{\partial p}{\partial x} > 0$, the pressure increases in the flow direction, promoting flow separation and increasing drag

6.3.3 Lift Force

The lift force is the aerodynamic component that acts perpendicular to the direction of the flow. This force is crucial in applications where control and stability depend on the object's ability to stay suspended or balanced within a fluid, as is the case with airfoils and stabilizing fins in underwater vehicles.

The viscous stress contribution to lift and side force is generally negligible, especially at high Reynolds numbers, where pressure forces dominate. In this context, the transverse-horizontal component of force is referred to as the side force Y, while the transverse-vertical component is known as the lift F_L [99].

$$F_Y = F \cdot \hat{y} \simeq (p_\infty - p_{\text{w}})\, \hat{n} \cdot \hat{y}\, \mathrm{d}\mathcal{S} \tag{94}$$

$$F_L = F \cdot \hat{z} \simeq (p_\infty - p_{\text{w}})\, \hat{n} \cdot \hat{z}\, \mathrm{d}\mathcal{S} \tag{95}$$

$$\tag{96}$$

In the subsequent analysis, viscous stress contributions to Y and L will be neglected, as they are typically insignificant compared to pressure forces at high Reynolds numbers.

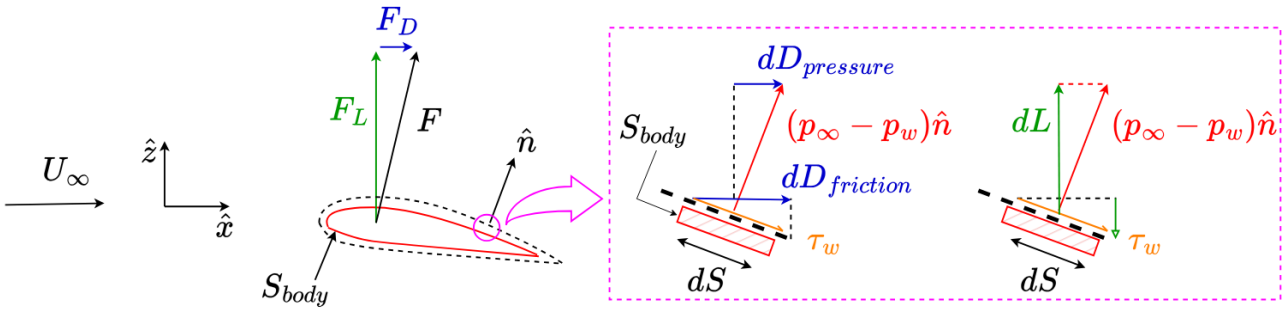


Figure 6.5: Surface pressure and viscous stress forces decomposed into drag and lift components.
Adapted from: [99]

6.4 Vorticity

Vortices are flow structures where the fluid rotates around a central axis, commonly found in turbulent flows or around objects that generate lift, such as wings and fins. Vortices can form due to flow separation, changes in velocity, or the geometry of the object in the flow. This section analyzes eddies, the Kármán effect, and induced drag at the tips of hydrofoil profiles.

6.4.1 Rate of rotation (angular velocity)

One of the easiest ways to describe a continuous movement of fluid is through rates of change. In the case of vortices, or points where the flow rotates around a point, it is described by the rotational rate, which is related to the **angular velocity** $\vec{\omega}$. The Angular velocity at a point is defined as the average rotation rate of two initially perpendicular lines that intersect at that point [12, 15].

$$\vec{\omega} = \frac{1}{2}\left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right)\mathbf{i} + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)\mathbf{j} + \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)\mathbf{k} \tag{97}$$

Thus, the angular velocity is half of the **vorticity vector** $\vec{\zeta}$, which we will discuss in the following subsection. The relation between angular velocity and vorticity is given by:

$$\vec{\omega} = \frac{1}{2}\nabla \times \vec{v} = \frac{1}{2}\vec{\zeta} \tag{98}$$

6.4.2 Vorticity Vector

The formation of vortices depends on various factors, including changes in fluid velocity, pressure gradients, and the presence of boundaries [13].One way to understand vortices is by deriving them from the Navier-Stokes equations. However, in the context of vorticity, these equations are simplified to exclude pressure and gravitational forces, focusing only on velocity [12]. We can define the **vorticity vector** as:

$$\vec{\zeta} = \nabla \times \vec{v} = \text{curl}(\vec{v}) \tag{99}$$

In Cartesian coordinates, the vorticity vector $\vec{\zeta}$ is expressed as:

$$\vec{\zeta} = \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z}\right)\mathbf{i} + \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}\right)\mathbf{j} + \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right)\mathbf{k} \tag{100}$$

6.4.3 Vorticity Equation

The vorticity equation is derived by taking the curl of the Navier-Stokes equations. It describes how the vorticity vector evolves within the flow over time, influenced by the velocity field, viscosity, and external forces. The general form of the vorticity equation is given by:

$$\frac{D\vec{\zeta}}{Dt} = (\vec{\zeta} \cdot \nabla)\vec{v} + \nu\nabla^2\vec{\zeta} \tag{101}$$

In Cartesian coordinates, the vorticity equation can be expressed as three scalar equations for the x, y, and z components of vorticity.

$$\begin{aligned}
\frac{D\zeta_x}{Dt} &= \zeta_x\frac{\partial u}{\partial x} + \zeta_y\frac{\partial u}{\partial y} + \zeta_z\frac{\partial u}{\partial z} + \nu\left(\frac{\partial^2\zeta_x}{\partial x^2} + \frac{\partial^2\zeta_x}{\partial y^2} + \frac{\partial^2\zeta_x}{\partial z^2}\right), \\
\frac{D\zeta_y}{Dt} &= \zeta_x\frac{\partial v}{\partial x} + \zeta_y\frac{\partial v}{\partial y} + \zeta_z\frac{\partial v}{\partial z} + \nu\left(\frac{\partial^2\zeta_y}{\partial x^2} + \frac{\partial^2\zeta_y}{\partial y^2} + \frac{\partial^2\zeta_y}{\partial z^2}\right), \\
\frac{D\zeta_z}{Dt} &= \zeta_x\frac{\partial w}{\partial x} + \zeta_y\frac{\partial w}{\partial y} + \zeta_z\frac{\partial w}{\partial z} + \nu\left(\frac{\partial^2\zeta_z}{\partial x^2} + \frac{\partial^2\zeta_z}{\partial y^2} + \frac{\partial^2\zeta_z}{\partial z^2}\right).
\end{aligned}$$

6.4.4 Vortices and Flow Separation in Submerged Objects

When an object is submerged in a moving fluid, various forces arise due to the interactions between the fluid and the object. In blunt bodies, like a sphere, pressure drag is the dominant component. This is largely due to flow separation. As the fluid flows around the object, it initially accelerates, leading to a decrease in pressure, a phenomenon known as a **favorable pressure gradient**. However, at a certain point, the fluid begins to decelerate, resulting in a pressure increase, known as an **adverse pressure gradient**. This adverse gradient can cause the flow to detach from the surface of the object, leading to **flow separation**.

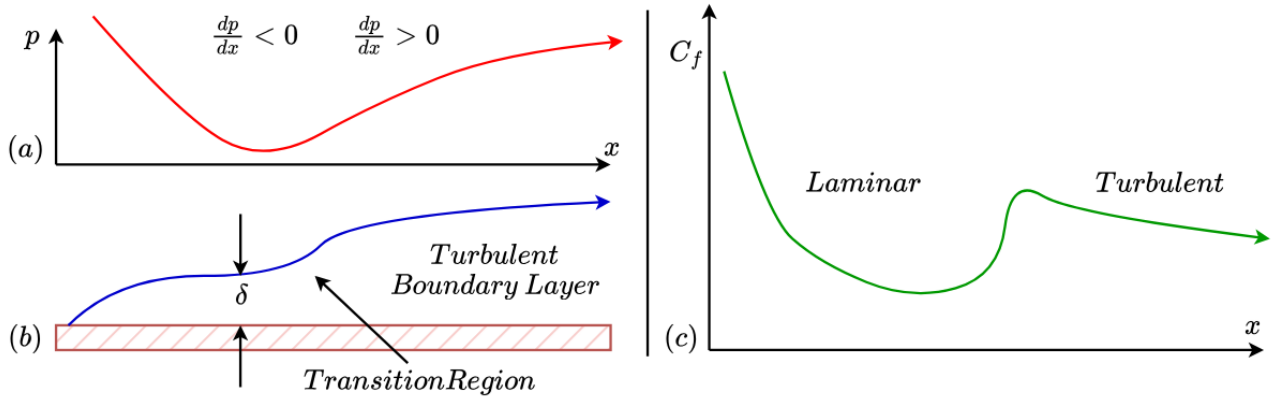


Figure 6.6: Effect of Pressure Gradient on Boundary Layer: The varying pressure distribution (a) leads to a transition to turbulent flow (b), with corresponding changes in skin friction (c). Adapted from: [14].

Flow separation creates a **low-pressure region** behind the object, called the **separation zone**, which results in a significant drag force. Additionally, this separation can lead to **vortex shedding**, where swirling vortices are formed behind the object, causing instability and, in some cases, unwanted vibrations in structures. These vortices form a periodic shedding pattern known as a **von Kármán vortex street**.

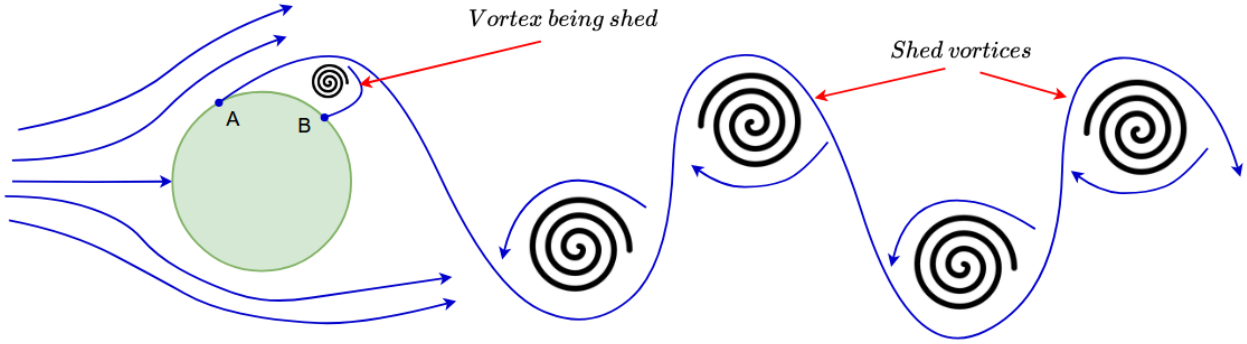


Figure 6.7: Vortex Shedding and fluid separation.
(A) detaching point laminar flow, (B) detaching point Turbulent flow. Adapted from: [12].

In turbulent flows, vortex shedding is accompanied by the creation of a hierarchy of **eddies at different scales**. Large eddies generate smaller ones, in a process described by Kolmogorov's hypothesis. This cascade of energy from large to small scales is a fundamental feature of turbulent flows, and it plays a crucial role in mixing and diffusion within the fluid, influencing the overall behavior of the flow.

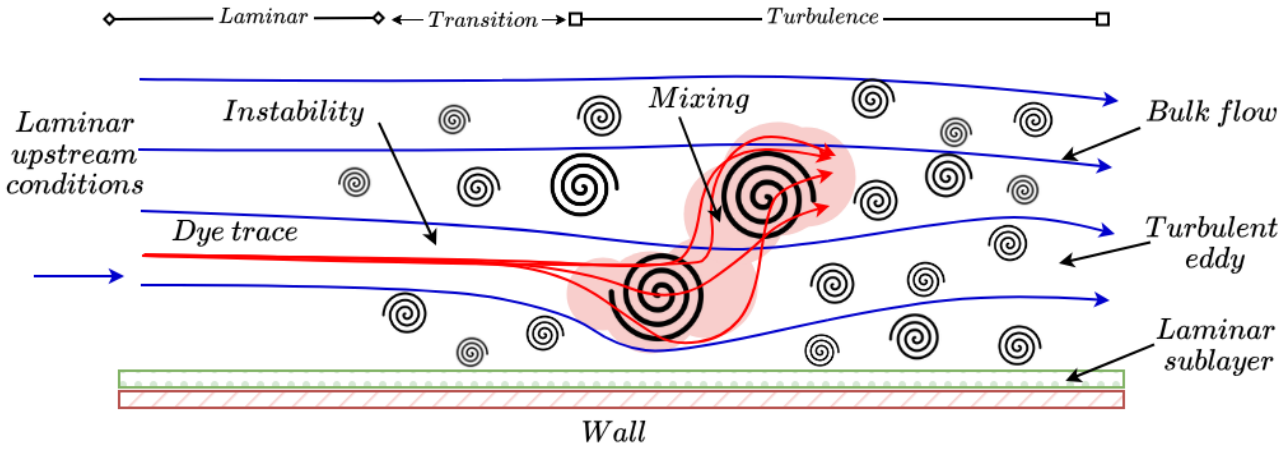


Figure 6.8: Eddies in a cascading mixing process. Adapted from: [11].

The vortex phenomena modelling and understanding is widely used in optimizing performance in applications like aircraft design, submarine design, weather prediction, and improving industrial processes such as chemical reactions and energy production.

6.4.5 Trailing Vortices

This type of vortex, also known as a wingtip vortex, and they are of specially interest in aerospace engineering, is formed due to the difference in static pressure between the upper and lower surfaces of the wing, as a result of the downward flow **(downwashing)** and upward flow **(upwashing)** of the fluid. This difference causes the fluid flow to curl around the tips of the fins, resulting in the formation of vortices, generating a type of drag known as induced drag. Under the influence of viscosity and turbulence, the trailing vortex is usually strongly rolled up within 2 to 3 chord lengths behind the wing. However, the induced velocity decreases inversely with distance from the vortex, so at about half the wingspan away, its effects diminish significantly.

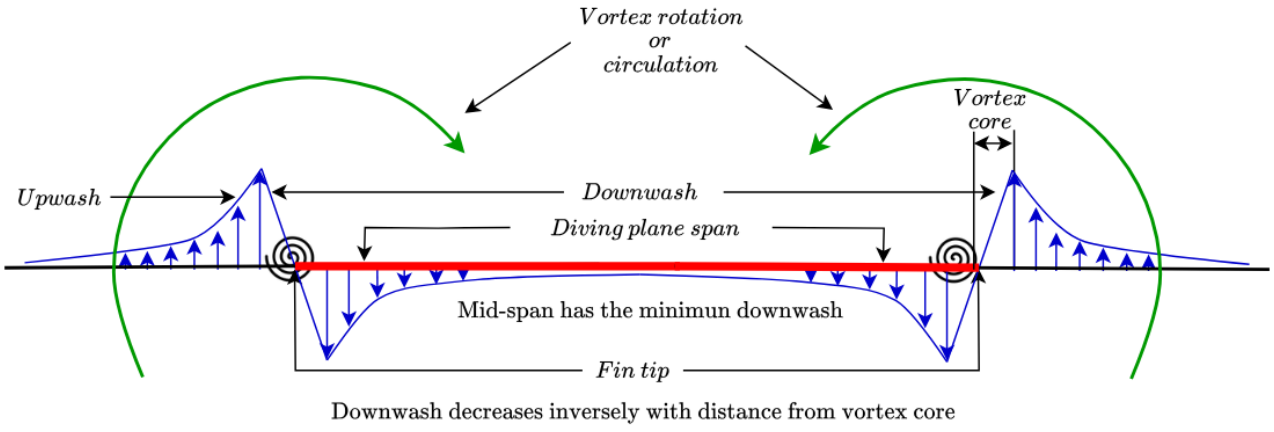


Figure 6.9: Trailing Vortices with Associated Downwash and Upwash Patterns. Adapted from: [13].

6.5 Hydrodynamics and Variable Calculations in ParaView

This section covers the **velocity vector Field**, **pressure distribution**, **drag force**, **lift force**, and **vorticity**, in sections (6.2,6.2,6.3.2,6.3.3,6.4), respectively and their calculations in Paraview. From the **.foam** or **.vti** files for CFD or PINN simulations. Both CFD and PINN simulations share similar steps, requiring to progress to the final simulation step via the **time** parameter. For visualization, CFD uses the imported object, while PINN algorithms specify object setup. Example implementations are available as **.pvsm** files at [1].

6.5.1 Fluid Velocity Field $\vec{U}$ algorithm in Paraview

The visualization of the flow lines of the Velocity Vector Field, are the most basic representation of the behavior of the fluid around the objects immersed in it because as seen previously in the section 6.2, they offer a quick way to reflect the magnitude of the velocity, the formation of vortices, the boundary, and the separation layer among others, for this the Algorithm 2 was implemented.

Algorithm 2 Visualization of Streamlines for the Fluid Velocity Field $\vec{U}$ in Paraview

Require: *Boundary.vti*, *Velocity.vti* ▷ Required only for PINN simulations
Require: *Simulation.foam* ▷ Required only for CFD simulations
procedure ExtractBoundary(*Boundary.vti*) ▷ Only for PINN simulations
    Compute the boundary contour from *Boundary.vti* using `Contour`
    Apply a box clip to refine the computed boundary contour using `Clip`
end procedure
procedure GenerateStreamLines(*velocity.vti* or *Simulation.foam*)
    Generate StremaLines using `Stream Tracer`
    Set line parameters (point 1 and point 2) from computed `StreamTracer`
    Set resolution from computed `StreamTracer`
    Set coloring as velocity and magnitude from computed `StreamTracer`
    Add directional arrows to computed `StreamTracer` using `Glyph`
    Set Orientation and Scale as Velocity from computed `Glyph`
    Set scale factor from computed `Glyph`
    Set maximum number of samples from computed `Glyph`
end procedure

6.5.2 Pressure Field Distribution p algorithm in Paraview

On the other hand, the pressure visualization allows seeing the effects of the fluid on the fluid domain and on the object immersed in it, especially the areas where the pressure exerts a greater influence. The result of this is the appearance of the Near-Field hydrodynamic forces that are evidenced in the section 6.2 and that can be visualized using the algorithm 3.

Algorithm 3 Visualization of Pressure Field Distribution p in Paraview

Require: *Boundary.vti*, *Pressure.vti* ▷ Required only for PINN simulations
Require: *Simulation.foam* ▷ Required only for CFD simulations
procedure ExtractBoundary(*Boundary.vti*) ▷ Only for PINN simulations
    Compute the boundary contour from *Boundary.vti* using `Contour`
    Apply a box clip to refine the computed boundary contour using `Clip`
end procedure
procedure GeneratePressureDistributionContour(*Pressure.vti* or *Simulation.foam*)
    Apply a plane clip to the *Pressure.vti* using `Clip`
    to Adjust the origin and normal vectors of the computed plane.
    Extract feature edges for contour visualization using `Contour`
    Configure contour range and resolution (step size) from the computed Contour
    Set solid coloring and select a suitable color from the computed Contour
    Customize the background for better visibility from the `Load a color palette` option in the task bar
end procedure

[1] https://github.com/YosefGuevara012/underwater_CFD_data/tree/master/Variable_Calculations_files

6.5.3 Drag Force (F_D) and Lift Force (F_L) algorithm in Paraview

Drag force and lift force are two of the variables that act directly on the body of the submerged object as a result of the behavior of the pressure and velocity of the surrounding flow. Drag is fundamental to determining the power required to propel a submerged object in the fluid, while lift, although not fundamental to marine robotics, is a force that can influence the stability and control of a vehicle in water. For them, the Algorithm 4 provides a basic framework for calculating these forces.

Algorithm 4 Calculation of Drag Force (F_D) and Lift Force (F_L) in Paraview

Require: *Boundary.vti*, *Pressure.vti* ▷ Required only for PINN-based simulations
Require: *Simulation.foam* ▷ Required only for CFD-based simulations
procedure ExtractAttributes(*Boundary.vti*, *Pressure.vti*) ▷ Only for PINN simulations
    Append attributes from *Boundary.vti* and *Pressure.vti* using `AppendAttributes`
    Compute the boundary contour based on the appended attributes using `Contour`
    Apply a box clip to refine the computed boundary contour using `Clip`
end procedure
procedure ExtractObjectMesh(*Simulation.foam*) ▷ Only for CFD simulations
    Select only object mesh from the Mesh Region in the *Simulation.foam* file.
end procedure
procedure ComputeDragAndLift(*AppendedAttributes* or *Simulation.foam*)
    Extract surfaces from the appended attributes using `ExtractSurfaces`
    Generate surface normals from the extracted surfaces using `GenerateSurfaceNormals`
        ▷ Note: Normal components (X, Y, Z) depend on object orientation in the simulation
    Compute F_D (Drag Force) as the product of pressure and surface normals
    Compute F_L (Lift Force) as the product of pressure and surface normals
    Integrate F_D and F_L over the surface to derive total drag and lift forces using `IntegrateVariables`
end procedure

6.5.4 Calculation of Vorticity vector $\vec{\zeta}$ algorithm in Paraview

Vortices are one of the major objects of interest in this work. Currently, no simulator as of the date of writing of this paper (Nov. 2024), no simulator can replicate and implement their effects. Vortices are the rotation about a point as a result of velocity changes, given the shape of the object in different fluid regimes. In the CFD simulations used in this work, it is sufficient to apply the Algorithm 2, and apply coloring for the visualization of vorticity and spredsheetview for the values of these phenomena at each point. However, for PINN simulations, it is necessary to use the Algorithm 5.

Algorithm 5 Calculation of Vorticity vector $\vec{\zeta}$ in Paraview

Require: *Boundary.vti*, *Velocity.vti* ▷ Required only for PINN simulations
Require: *Simulation.foam* ▷ Required only for CFD simulations
procedure ExtractBoundary(*Boundary.vti*) ▷ Only for PINN simulations
    Compute the boundary contour from *Boundary.vti* using `Contour`
    Apply a box clip to refine the computed boundary contour using `Clip`
end procedure
procedure CalculateVoricityAndRotation(*velocity.vti* or *Simulation.foam*)
    Generate a slice using (`Slice`)
    Set plane Origin and normals from computed `Slice`
    Compute the Vorticity using `Gradient of Unstructured DataSet`
    Display vorticity from computed Vorticity.
    Set coloring as velocity and magnitude from computed `StreamTracer`
    Add directional arrows to computed `StreamTracer` using `Glyph`
    Set Orientation and Scale as Velocity from computed `Glyph`
    Set scale factor from computed `Glyph`
    Set maximum number of samples from computed `Glyph`
end procedure

CHAPTER 7

Results: Simulation Analysis and Comparison

Contents

This chapter provides a detailed analysis and comparison of simulation results, focusing on hydrodynamic phenomena and forces calculation using the PINN architectures (4.2) and comparing these results against each other and the CFD baseline. The first section (7.1) examines the training performance of Pruned-UNet and U-Net, focusing on convergence and loss patterns. The second section (7.2) evaluates fluid convergence and stabilization during simulation across both PINN architectures and CFD. Sections three and four (7.3, 7.4) compare flow patterns and pressure distribution for various geometries, including spheres, flat plates, and NACA 0018 profiles. Section five (7.5) analyzes lift and drag under different flow conditions, while section six (7.6) focuses on vortex formation. The seventh and eighth sections (7.7, 7.8) discuss computational time analysis and boundary condition challenges. Section nine (7.9) examines fluid simulations over a torpedo-shaped UV. And Section ten (7.10) the summary of the findings.

7.1 Training Results Comparison

As shown in Figures 7.1 and 7.2, corresponding to the **Pruned-UNet** and **U-Net** architectures, the Compound Loss Function $\mathcal{L}$ defined in Equation (76) generally decreases over time. Training was conducted on an **NVIDIA A100 80GB GPU (CUDA)** and took approximately 4 weeks per architecture, without a stopping criteria.

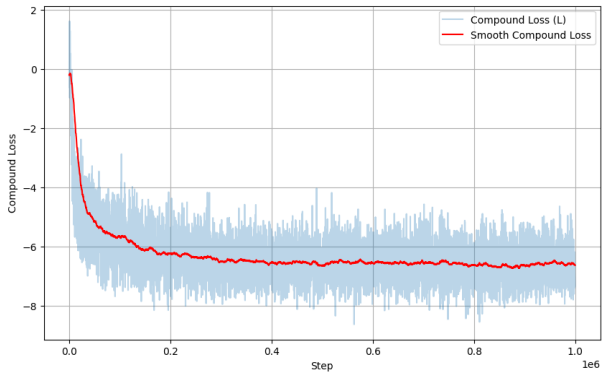


Figure 7.1: Pruned-UNet training Loss Evolution.

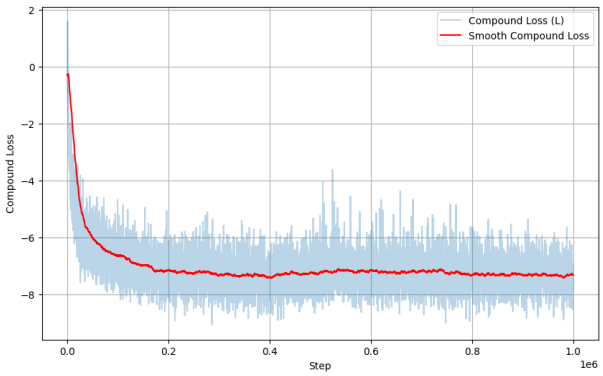


Figure 7.2: U-Net training Loss Evolution.

Both models display notable spikes in the Compound Loss $\mathcal{L}$ during training, which originate from nonlinear terms in the Navier-Stokes equations, advection, diffusion, pressure gradients, and velocity-pressure interactions—compounded by boundary conditions. The **Adam** optimizer struggles to achieve a path due to these complexities. As training progresses, sharp directional changes in the optimizer appear from mini-batch samples of varied object shapes, leading to oscillations.

Additional sources of these spikes include surface discontinuities, strong velocity gradients at boundaries, and interactions like flow separation and vortex shedding. These challenges make it difficult for both networks to balance constraints, such as boundary conditions, incompressibility, and momentum equations. After 200,000 steps, both models stabilize, though fluctuations persist with diminishing intensity. The full U-Net achieves a lower Compound Loss, indicating that its additional parameters better capture complex fluid patterns. By 600,000 steps, loss values converge, as presented in Table 7.1.

Architecture	**Converged Compound Loss Range**
Pruned U-Net	-6.5 to -7.0
Full U-Net	-7.5 to -8.0

Table 7.1: Converged Loss Ranges of Pruned and U-Net Architectures.

Given this behavior, the training results were smoothed to improve interpretability using the Algorithm 6. Although some spikes still present.

Algorithm 6 Exponential Moving Average Data Smoothing

Require: *Data Array*
Ensure: $weight \geq 0.60$
  $smoothed \leftarrow []$ ▷ Initialize the smoothed data array
  $last \leftarrow data[0]$ ▷ Set the initial value to the first data point
  for each $point \in data$ **do**
    $smoothed\_val \leftarrow last \times weight + (1 - weight) \times point$
    Append $smoothed\_val$ to $smoothed$
    $last \leftarrow smoothed\_val$
  end for
  return *smoothed*

The following figures show the behavior of the remaining mean loss function evolution for the two architectures and the compound loss function, after being processed by the Exponential Smoothing (Algorithm 6), each with its corresponding scale differentiated by colors.

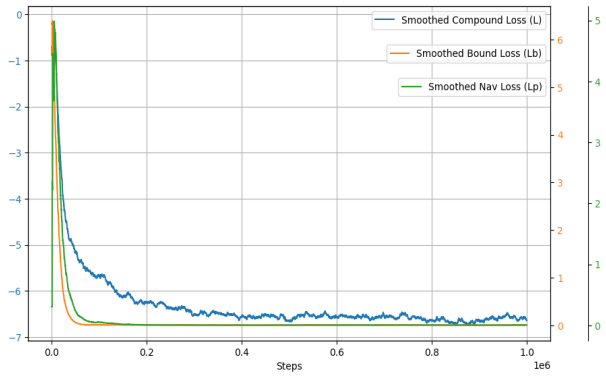


Figure 7.3: Pruned-UNet Architecture Loss Evolution.

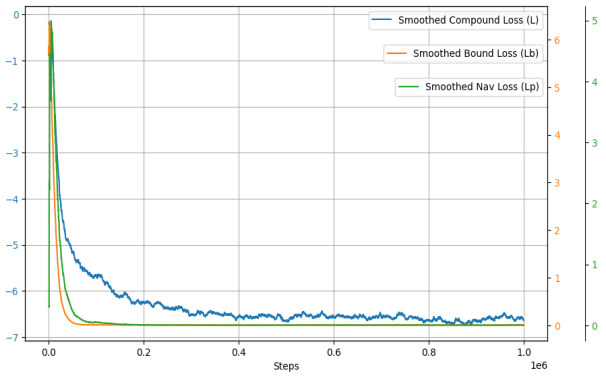


Figure 7.4: U-Net Architecture Loss Evolution.

The **Smoothed Mean of the logarithmic loss bound (orange line)** and **Smoothed Mean of the logarithmic loss nav (green line)** exhibit similar trends, indicating that both models converge to satisfy values close to the physical constraints of boundary conditions and the Navier-Stokes momentum equations.

Despite the large spikes due to the initial conditions, the smoothed loss bound for both architectures quickly approaches zero. In the U-Net, this loss starts with a significant spike early in training but decreases to nearly zero before the 200,000-step mark. Slight spikes appear between 300,000 and 700,000 steps, with spikes reaching around 0.25 before stabilizing. These fluctuations suggest problems in satisfying boundary conditions during training. The smoothed loss nav square follows a similar trend, reaching near-zero after initial spikes, particularly for the U-Net around 50,000 steps.

Overall, the Pruned U-Net maintains performance comparable to the full U-Net despite its simpler architecture. The pruned U-Net demonstrates similar performance reducing the lost function without significant differences, indicating that the pruning process successfully reduces model complexity without compromising convergence quality. Both models suggested stable solutions to boundary conditions and the Navier-Stokes equations. This suggests a physical behavior of the fluid approximation.

7.2 Convergence and Minimum valuesEvaluation

Focusing on the convergence values during the simulation, we use a cube with side length $S = 1\,\text{m}$ (or 16 voxels) as a benchmark, as was mentioned previously (Section 6.1). Based on that, we set up a dynamic viscosity $\mu = 0.1\,\text{Ns/m}^2$, and density $\varrho = 5\,\text{Kg/m}^3$, with a flow velocity of $\vec{v} = 1\,\text{m/s}$. Where the Reynolds Number Re=800 used in [10] which corresponds to a turbulent flow regime. Using **absolute divergence** as the metric (blue line), Figures 7.5 and 7.6 show that Pruned-UNet and U-Net architectures reaches a minimum in approximately 500 and 400 iterations, respectively.

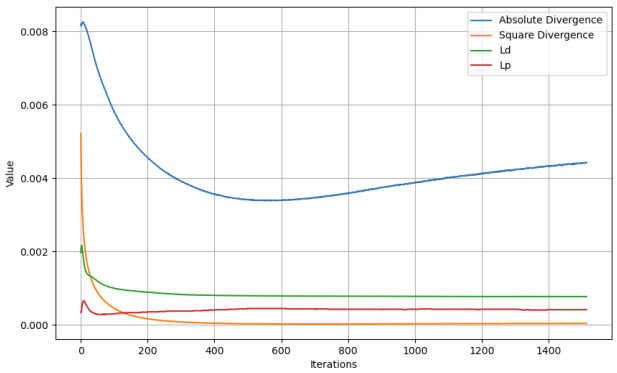


Figure 7.5: Pruned-UNet Convergence Evolution.

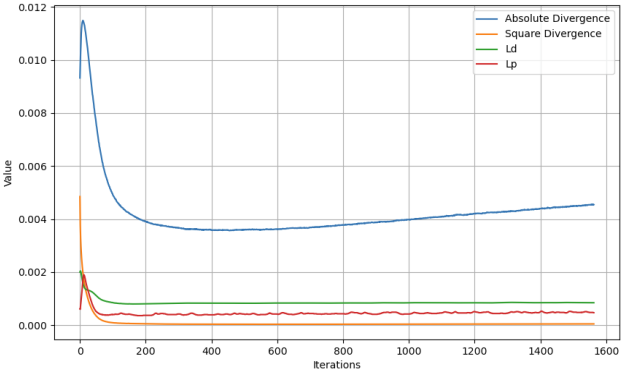


Figure 7.6: UNet Convergence Evolution.

The absolute divergence values are 0.0034 for the Pruned U-Net and 0.0036 for the U-Net, with both models showing almost identical divergence values despite the simpler architecture of the Pruned U-Net. This small divergence value suggested that both models are capable of fulfilling the incompressibility conditions. The remaining metrics for both architectures stabilized around 200 iterations, as shown in Table 7.2.

	Absolute Divergence	*Divergence*^{2}	Ld	Lp
Pruned-UNet	0.0034	3.50342e-05	0.00079	0.00045
UNet	0.0036	3.60375e-05	0.00083	0.00039

Table 7.2: Comparison of Pruned and U-Net Simulation Convergence.

Comparing this minimum value with the CFD simulation results for the cube, where the same flow velocity and kinematic viscosity were used, we observe that the $k-\omega SST$ model with the SIMPLE Algorithm 1 converges to a Drag coefficient C_d of 1.3993×10^{-2} after approximately 500 iterations, which takes around 10–30 minutes, per shape. The logarithm of this value was used for visualization in Figure 7.7.

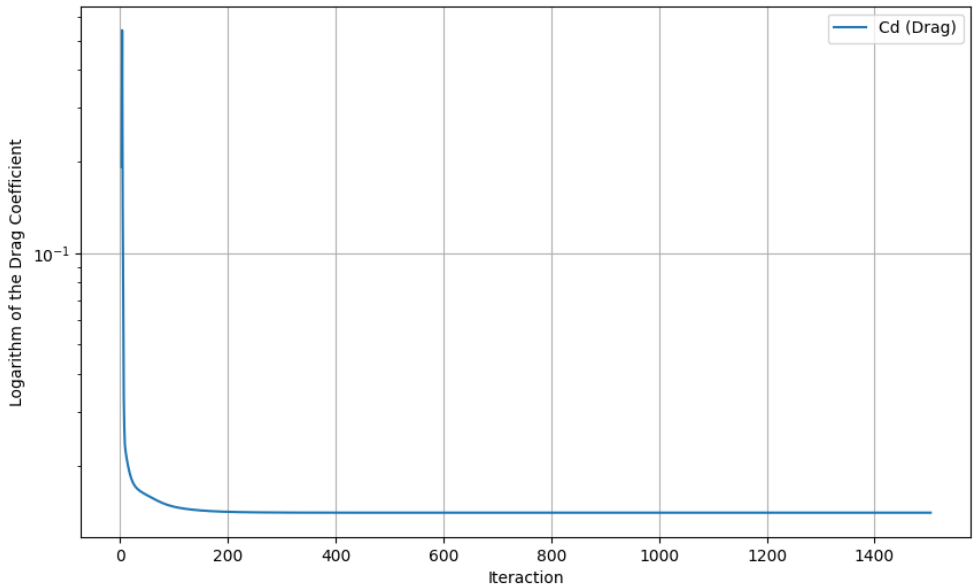


Figure 7.7: Logarithm of the Drag coefficient over iterations.

7.3 Comparison of Flow Patterns

Now, we will focus on the results of the flow comparison using streamlines. The analysis is based on three objects: a sphere, a flat plane, and our simplified voxelized hydrofoil profile based on the NACA0018. The aspects of the comparison include the stagnation point, the fluid field development and the boundary layer for each shape.

7.3.1 Sphere

For a sphere with a diameter $D = 1\,\mathrm{m}$ or (16 voxels), submerged in a laminar flow regime with a dynamic viscosity $\mu = 5\,\mathrm{Ns/m^2}$, density $\varrho = 0.2\,\mathrm{kg/m^3}$, and velocity vector field $U = 1\,\mathrm{m/s}$, we obtain a Reynolds number $\mathrm{Re} = 0.64$ referencing [10]. As expected with $\mathrm{Re} < 1$ from the theory, no fluid separation is observed [100]. When comparing the Pruned-UNet to the U-NET, the Pruned-UNet shows a greater reduction in velocity field in the laminar wake. The flow in the Pruned-UNet recovers the initial velocity after 70 voxels (4.4 meters) Figure 7.8, compared to 30 voxels (1.8 meters) in the U-NET Figure 7.9. This suggests that the Pruned-UNET introduces more numerical dissipation, which may reduce the accuracy in regions requiring detailed flow structure.

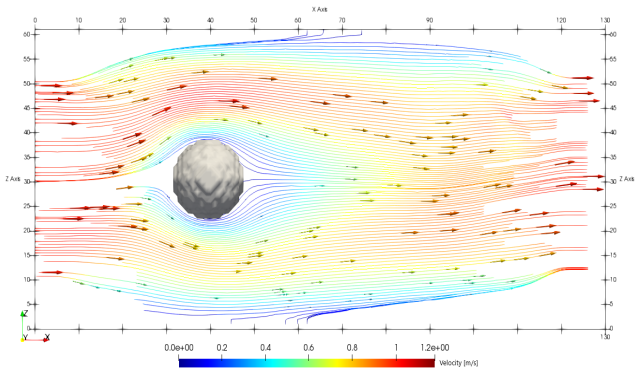


Figure 7.8: Pruned UNet: streamlines laminar flow.

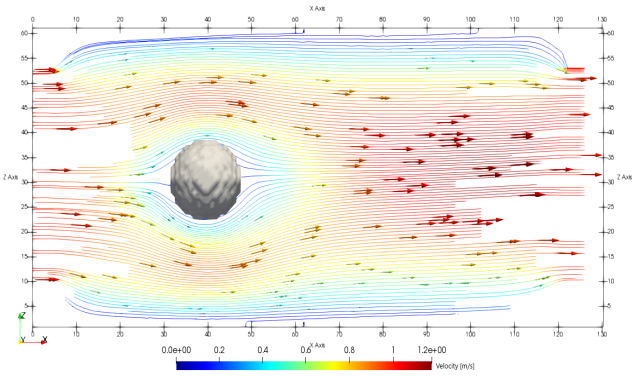


Figure 7.9: U-Net: streamlines laminar flow.

On the other hand, the behavior of the fluid changes in a turbulent flow regime with $\mu = 1\,\mathrm{Ns/m^2}$, $\varrho = 0.02\,\mathrm{kg/m^3}$ and the velocity vector field $U = 1\,\mathrm{m/s}$, referencing [10]. As expected, in both cases, vortex formation is observed due to the appearance of a separation region. The vortices are larger for the U-Net Figure 7.11 than for the Pruned-UNet Figure 7.10. In the Pruned-UNet case, the flow separation occurs around 75°, which is close to the expected behavior in laminar flows for Re>1, where separation typically occurs at around 80° [100].

In contrast, for the U-NET, the separation occurs between 90° and 120°, which is closer to theoretical predictions for turbulent flows, where the separation occurs at 130° [100]. However, it is important to note that, although the simplified shape of our sphere in the PINN acts like a rough surface, the separation region is not further delayed as expected [100]. This suggests that the PINN is unable to represent the behavior that would occur on rough surfaces, where separation typically occurs at higher degrees, to reduce the separation region.

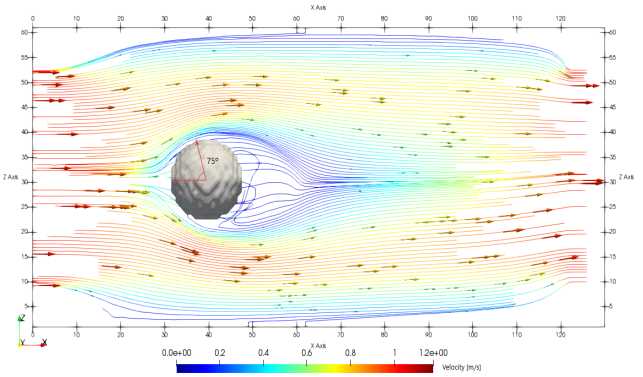


Figure 7.10: Pruned UNet: streamlines Turbulent flow.

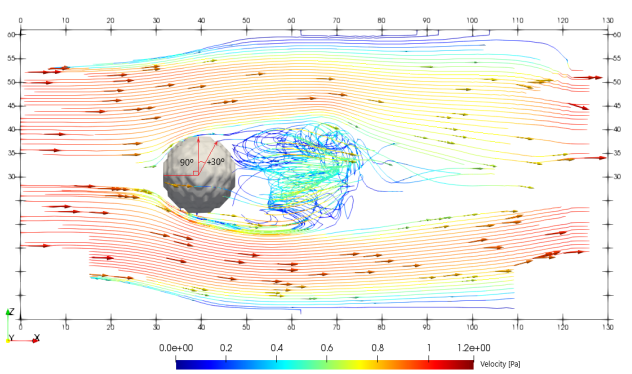


Figure 7.11: U-Net: streamlines Turbulent flow.

For both architectures and flow regimes, the fluid velocity approaches zero near the surface of the sphere and the domain walls, satisfying the Dirichlet boundary conditions, where the flow near the wall adopts the surface velocity [86]. The Pruned-UNet reduced vortices indicates that it may struggle to capture the finer scale turbulent structures, leading to smoother, less chaotic flow compared to the U-Net.

On the other hand, under the same fluid conditions, the CFD simulation does not exhibit the distinct laminar and turbulent behavior as seen in the PINN simulations for Re = 0.64 or Re = 800 as referenced in [10]. Instead, both scenarios in the CFD simulation behave similarly, remaining in a laminar regime, as shown in Figure 7.12. This suggests that the PINN simulation is more sensitive to generating a turbulent flow regime under the same fluid conditions. To achieve turbulent behavior in the CFD simulation, the velocity must be increased, as shown in Figure 7.13, where the velocity was raised to 16 m/s for the same values of density and viscosity.

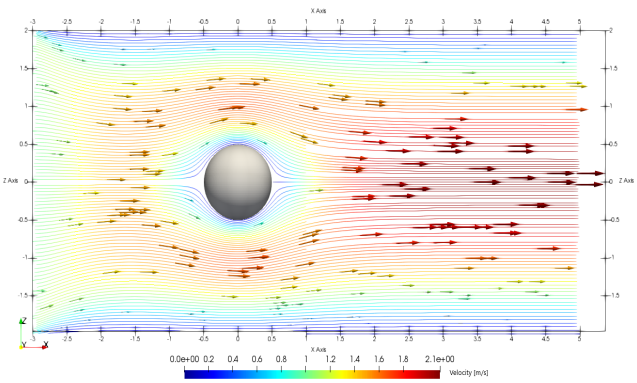


Figure 7.12: CFD: Laminar flow regime.

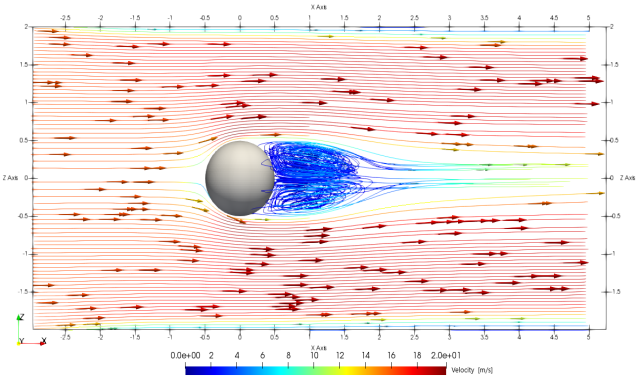


Figure 7.13: CFD: Turbulent flow regime.

7.3.2 Curl Analysis.

By examining the same simulations from an isometric perspective, we observe that even in the laminar regime, where Re = 0.64, the fluid is not curl-free ($\nabla \times (\nabla q) = 0$) and exhibits rotational behavior. The Pruned-UNet shows a larger curl compared to the U-Net, which is significantly smaller or negligible curl in Figure 7.15. As previously mentioned, this explains one of the reasons why the velocity magnitude in the wake region behind the sphere is lower in the Pruned-UNet than in the U-Net.

The larger curl in the Pruned-UNet indicates stronger rotational motion with a maximum rotation of 0.672 rad/s over the mid-part of the sphere (Figure 7.14), which contributes to greater energy dissipation and a slower recovery of the flow velocity. The Pruned-UNet appears to be more sensitive to the fluid interaction with the walls, likely due to the simpler architecture increasing the numerical dissipation, which amplifies the effects of boundary conditions.

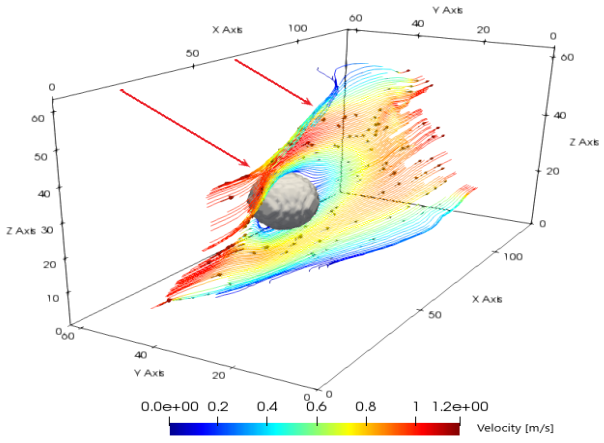


Figure 7.14: Pruned-UNet: Curl in a Laminar flow.

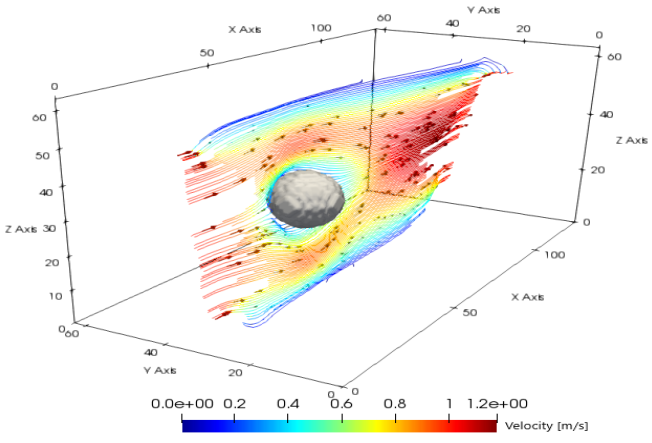


Figure 7.15: U-Net: Curl in a Laminar flow.

If we focus on the turbulent regime Re= 800 referencing 6.1, we can see that the curl decreases for the Pruned U-Net with a maximum rotation of 0.0131 rad/s over the z-axis (Figure 7.16), while for the U-Net 7.17, the curl behaves similarly in both flow regimes, if we do not consider the appearance of vortices in the turbulent regime. In contrast, a curl-free behavior appears in the laminar regime and the turbulent regime, on CFD simulations on the parallel streamlines.

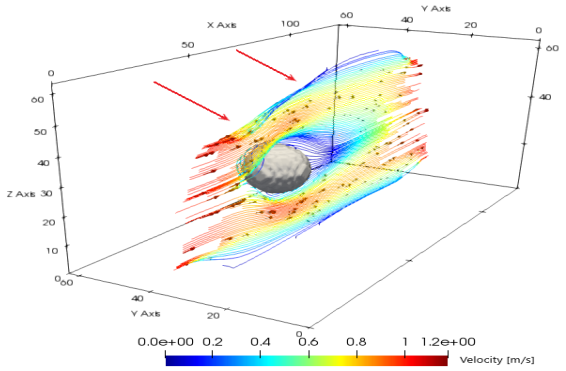


Figure 7.16: Pruned-UNet: Curl in a Turbulent flow.

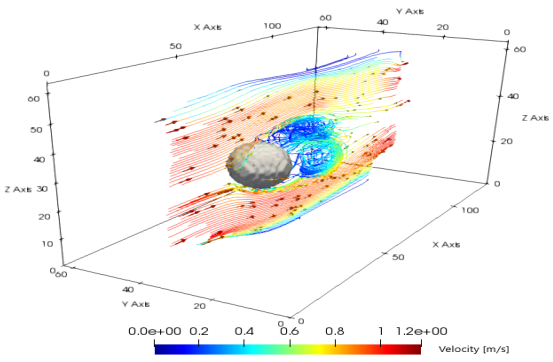


Figure 7.17: U-Net: Curl in a Turbulent flow.

7.3.3 Flat plate

For blunt bodies like the flat plane at 90° (Heave orientation) as shown in Figure 7.18, the flow easily separates, creating a wide separation region described as turbulence with Re = 800, as referenced in 6.1. These turbulences are more evident in the simulation using the U-Net architecture, as shown in Figure 7.19.

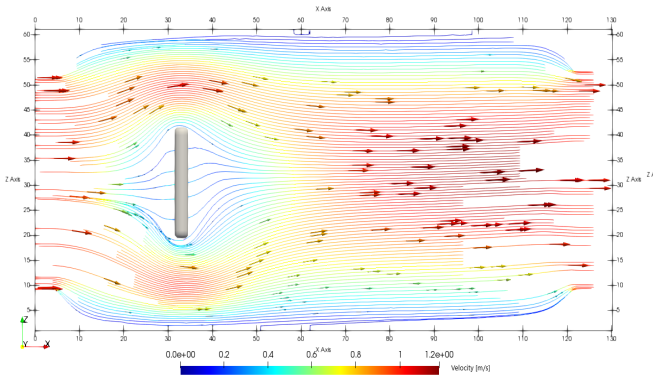


Figure 7.18: Pruned-UNet: Flat Plane at 90°.

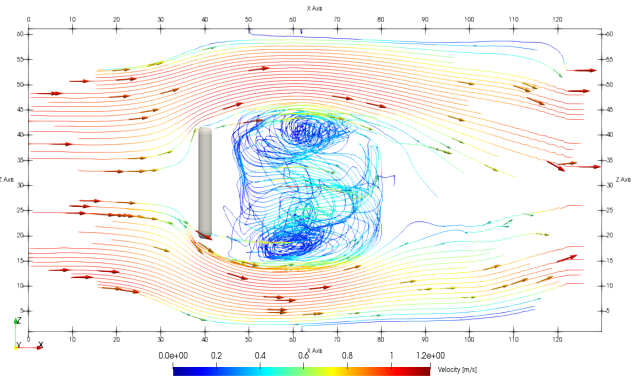


Figure 7.19: U-Net: Flat Plane at 90°.

Analyzing the velocity behavior along the x-axis for the flat plane aligned with the flow direction (Surge oriented) at 0°, the U-Net simulation with Re = 800 shows a well-defined velocity gradient, consistent with theory [12, 14, 15]. As seen in Figure 7.20, the velocity progressively decreases as it approaches the object's surface along the z-axis, reflecting boundary layer development (Figure 7.21).

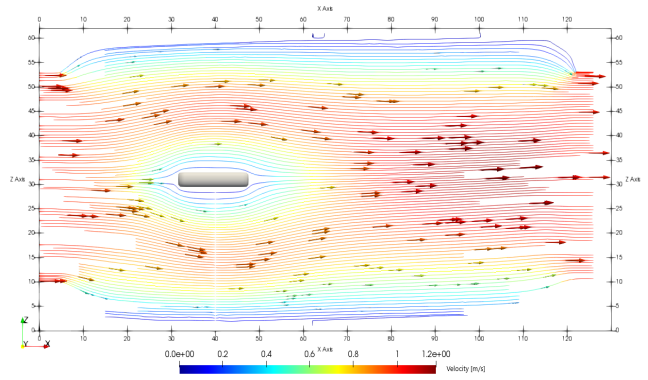


Figure 7.20: UNet: Flat Plane at 0°.

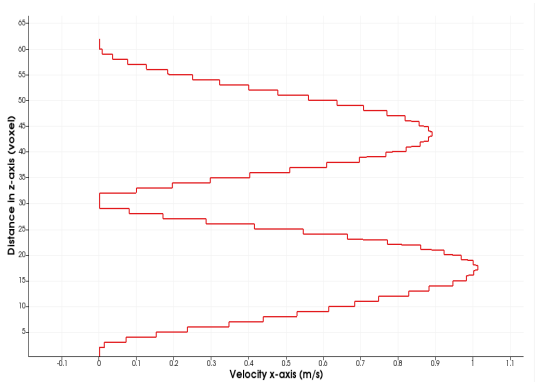


Figure 7.21: Velocity gradient for a Flat Plane. Re = 800, according to [10].

7.4 Pressure Field Distribution Evaluation

To analyze the pressure distribution, we used our cube as a benchmark, setting Re = 800 according to 6.1. In the CFD simulation in Figure 7.22, the pressure is symmetrically distributed on the top and bottom faces of the cube, which is typical for symmetric geometries [13]. The front face of the cube shows high pressure due to the stagnation point, where the flow directly impacts the surface and the fluid velocity drops almost to zero [11, 13, 99]. This high pressure is represented by a dense clustering of contour lines and a red color. Additionally, small pressure accumulations were observed in the rear corners of the cube.

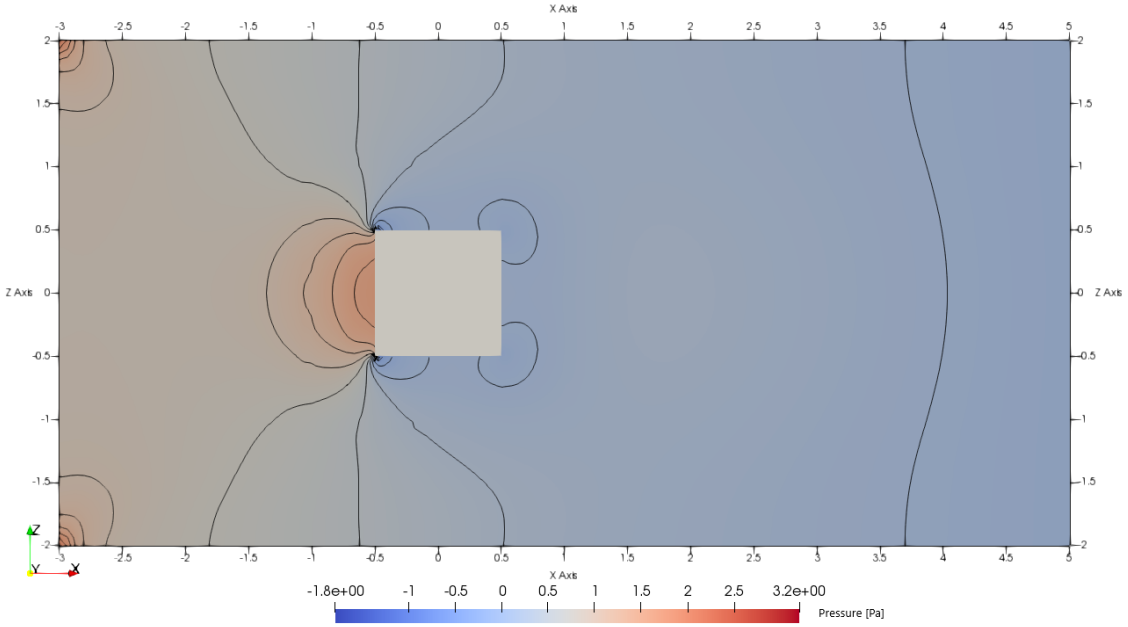


Figure 7.22: CFD: Cube pressure distribution.

Comparing the results obtained from the Pruned-UNet (Figure 7.23) and U-Net (Figure 7.24) with the CFD simulation, we observed a similar pressure accumulation on the front face of the cube. However, this accumulation is lower than that seen in the CFD results. Neither model shows the pressure accumulation at the rear corners of the cube, as observed in the CFD simulation. Instead, Kármán vortex streets are present in the wake of the cube, with a higher magnitude in the U-Net simulation compared to the Pruned-UNet.

Additionally, neither of the PINN models exhibits a symmetric pressure distribution in the fluid domain, likely due to significant pressure accumulation at the domain boundaries, as indicated by the blue contour lines. This phenomenon is more visible in the rear part of the domain, possibly because of the PINN simulations tend to generate more turbulent flows for the same flow conditions and problems with the simulation for handling boundary conditions. In contrast, in the CFD simulation, the pressure accumulations are more localized at the front corners of the domain.

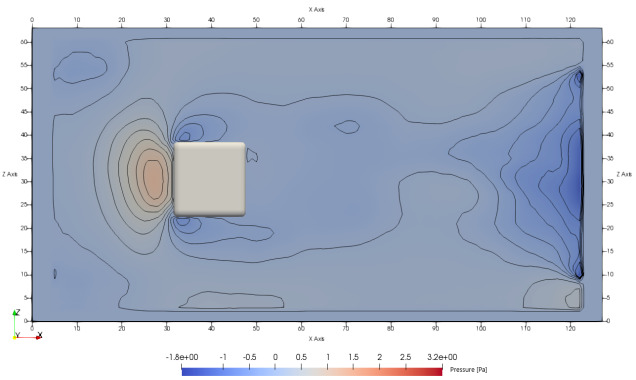


Figure 7.23: Pruned-UNet: Cube pressure distribution.

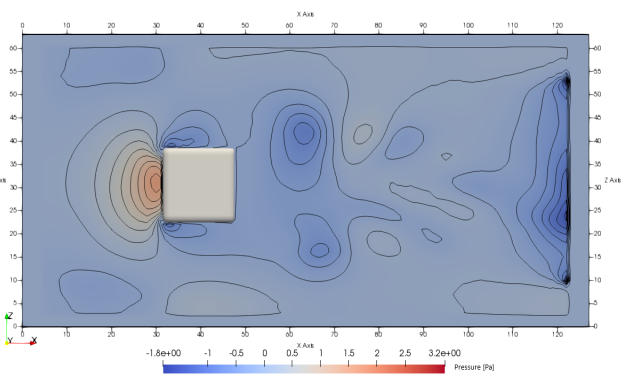


Figure 7.24: U-Net: Cube pressure distribution.

7.4.1 Flat plane

For the flat plane, a high-pressure zone forms at the front stagnation point, reaching about 2 Pa. In turbulent regime, eddies appear in both cases. When aligned with the flow (Surge oriented at 0°), the separation zone is smaller (Figure 7.25). In Heave oriented (90°), the stagnation point shifts to the center of the front face, with small eddies forming at the corners due to sharp pressure gradients (Figure 7.26). A larger separation region then develops, creating a Kármán vortex street in the wake.

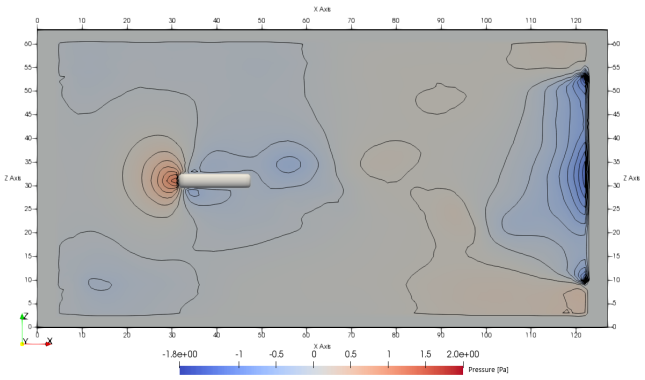


Figure 7.25: Flat Plane at 0° pressure distribution.

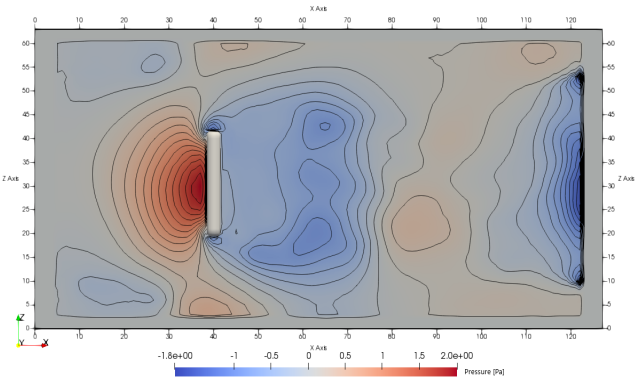


Figure 7.26: Flat Plane at 90° pressure distribution.

Comparing CFD results (7.27) under the same conditions with Re = 800 (see 6.1), both CFD and U-Net streamlines follow a similar pattern. However, maximum velocity in CFD is 0.3 m/s higher than in U-Net (7.20). The velocity field in CFD recovers within 4 m downstream, while U-Net regains initial velocity in about 40 cm, likely due to numerical dissipation.

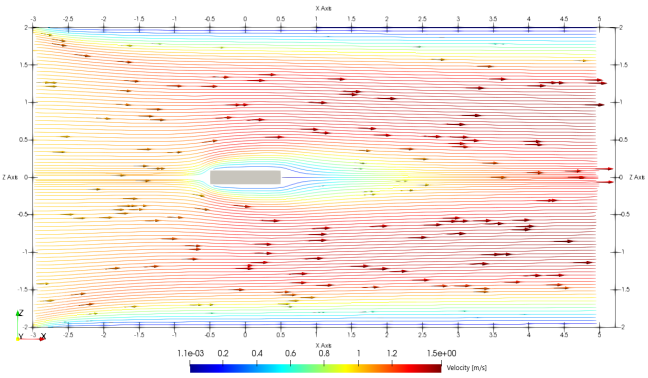


Figure 7.27: Flat Plate streamlines in CFD.

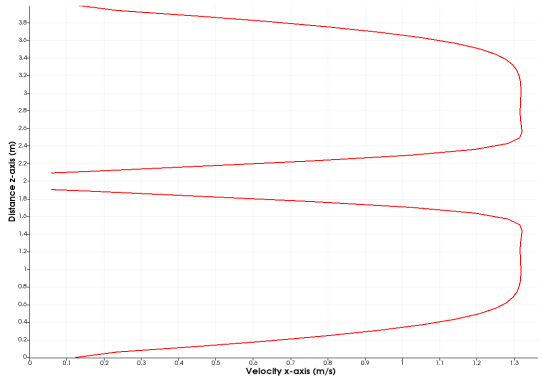


Figure 7.28: Flat Plate Velocity Gradient in CFD.

Looking at the velocity gradients on the flat plane surface, both CFD (7.28) and U-Net (7.28) display similar profiles; however, CFD being less affected by boundary effects. In terms of pressure, CFD (7.29) and U-Net (7.25) reach similar maximum values, but CFD maintains nearly zero pressure accumulation throughout the domain, whereas U-Net shows marked pressure buildup.

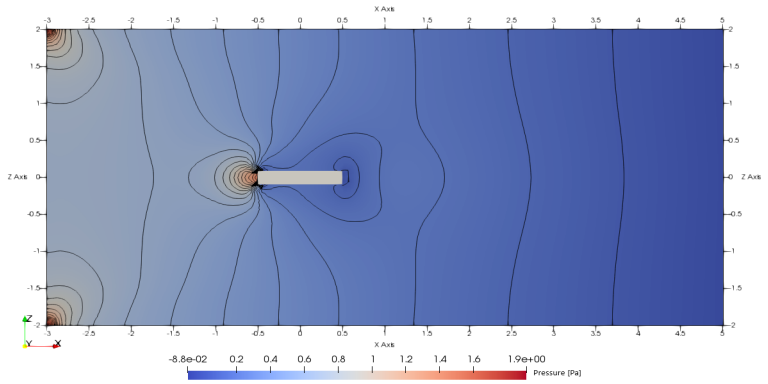


Figure 7.29: Flat Plate Pressure Distribution in CFD.

7.4.2 NACA 0018 Hydrofoil Profile

A comparison was made between the behavior of the NACA 0018 profile (Figure 7.30), a symmetric Hydrofoil 4.3.1, which shows equal pressure distribution on both the upper and lower surfaces, consistent with theoretical expectations [13]. A similar test was conducted by modifying the lower surface to break the symmetry (Figure 7.31). Despite this new shape not being part of the training set, the model still generated wake vortices and a coherent pressure distribution. As expected due to theory [13], a pressure difference between the upper and lower surfaces was observed, evidenced by negative pressure values on the surface, suggesting possible flow separation or reversal, consistent with turbulent regime behavior.

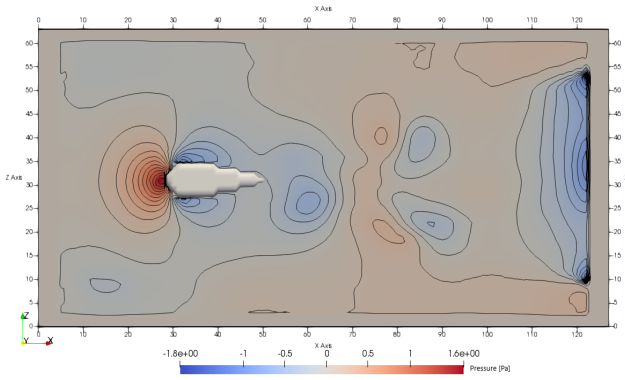


Figure 7.30: U-Net: Hydrofoil pressure distribution.

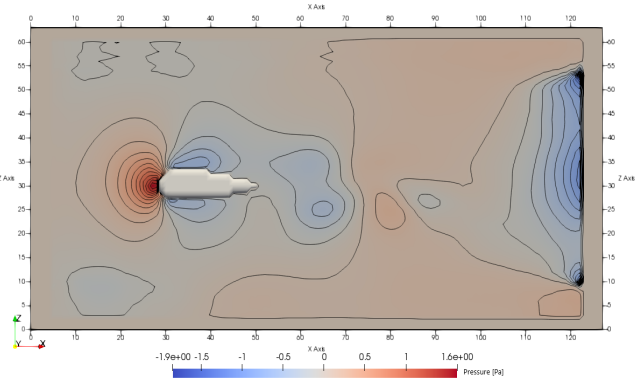


Figure 7.31: U-Net: Modified Hydrofoil pressure distribution.

As previously shown in the comparison of velocity distribution via streamlines in CFD (7.32), the NACA 0018 profile exhibits a stagnation point, where fluid velocity reaches zero, and a symmetric reduction in fluid velocity around the surface. Under the same fluid conditions, the profile does not develop turbulent flow, as observed in U-Net. This velocity profile leads to a symmetric pressure distribution around the object (7.33), similar to the behavior seen in the U-Net simulation (7.30). Additionally, in CFD and U-Net, pressure accumulation occurs only at the stagnation point and around the leading-edge corners, with a minor difference in maximum pressure values, around 0.2 Pa.

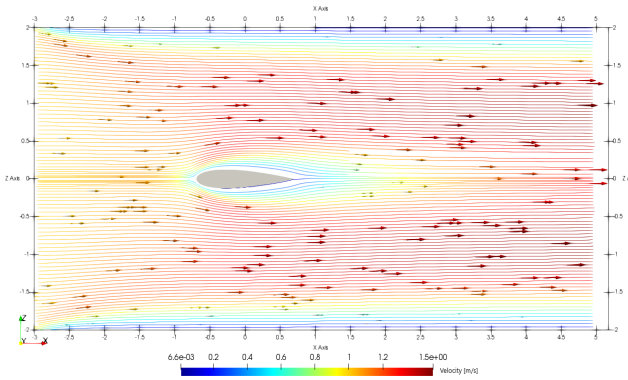


Figure 7.32: Streamlines for NACA 0018 Profile.

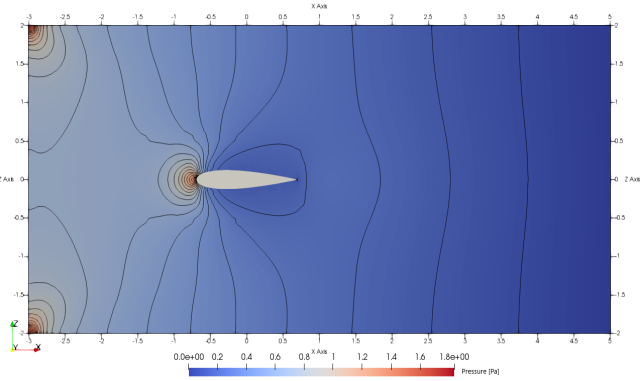


Figure 7.33: NACA 0018 Profile Pressure Distribution.

7.5 Lift and Drag Forces Comparison

Drag and lift (N) are the result of forces exerted by the fluid on a submerged object, as they are calculated from dynamic pressure and the reference area. As described in the methodology, these forces were calculated using the approximations in 6.3.3. The results of this analysis are presented in the tables drag force 7.3 y lift force 7.4 tables, based on the conditions defined in 6.1, for Re = {0.64, 80, 800}, comparing the PINN architectures Pruned-UNet, U-Net, and CFD, with a flow velocity of 1 m/s in all simulations.

7.5.1 Drag Force Evaluation

Focusing on the behavior of different shapes within the same architecture, we observe that drag is influenced by both the orientation and shape of the object in relation to the flow, as was expected from the theory [13, 14]. For instance, in the Flat plane oriented in Heave oriented 90°, drag increases as the object becomes blunt, generating a larger separation region. In contrast, minimal flow separation occurs in the Flat plane 0°, resulting in minimal pressure drag.

Regarding discrepancies between the PINN architectures and, CFD, these are more significant at low Reynolds numbers (Re = 0.64), for example, the cube the calculated drag force is 9.927 (N), but in CFD is 964,226 (N) where the flow is laminar and dominated by viscous effects. The PINN appear to struggle under these conditions, possibly due to the higher sensitivity of laminar flows to boundary conditions and subtle flow features like boundary layer attachment.

	Pruned-Unet			U-Net			CFD		
Re	**0,64**	**80**	**800**	**0,64**	**80**	**800**	**0,64**	**80**	**800**
Cube	9,927	15,520	51,619	111,507	26,300	92,618	694,226	6,000	1,362
Sphere	9,463	12,731	7,953	64,453	16,617	57,540	310,14	2,740	0,652
Flat plane (0°)	1,559	5,641	3,615	22,830	4,647	23,102	154,809	1,314	0,249
Flat plane (90°)	20,449	22,926	18,344	296,743	72,339	312,54	616,321	5,725	1,559
Hydrofoil	4,845	14,759	7,694	53,744	14,257	59,630	82,400	0,801	0,191
Hydrofoil modified	3,419	10,406	5,315	49,689	12,384	11,841	—	—	—

Table 7.3: Drag Force F_L (N) values across different Reynolds numbers (Re) using three methodologies.

7.5.2 Lift Force Evaluation

For symmetrical objects, lift force is expected to be negligible due to the symmetry of pressure distribution, as confirmed by the CFD results showing near-zero lift. However, both PINN architectures (Pruned U-Net and U-Net) exhibit non-negligible lift values, especially at high Reynolds numbers (Re = 800). For instance, the Pruned U-Net predicts a lift of -8.473 N for the cube, while the U-Net predicts 2,598 N. These discrepancies with CFD likely result from numerical artifacts, particularly boundary condition handling issues, leading to pressure accumulation at the domain boundaries.

On the other hand, for the modified hydrofoil profile, PINN predictions show reasonable lift force, consistent with expectations due to the Hydrofoil's asymmetry. The U-Net predicts a lift of 7.83 N at Re = 0.64, reflecting the pressure difference between the upper and lower surfaces, in line with aerodynamic theory [13, 14].

	Pruned-Unet			U-Net			CFD		
Re	**0,64**	**80**	**800**	**0,64**	**80**	**800**	**0,64**	**80**	**800**
Cube	-1,952	-2,666	-8,473	-9,405	-1,456	2,598	0,022	0,000	0,000
Sphere	-0,237	-0,846	-1,644	-2,847	0,154	-5,591	0,013	0,000	0,000
Flat plane (0°)	-0,281	0,213	-2,333	2,937	-2,75	-8,124	0,047	0,000	-0,001
Flat plane (90°)	-1,561	-0,521	-1,115	1,061	1,107	8,960	0,019	0,000	0,000
Hydrofoil	2,590	0,177	-2,220	2,278	-3,879	17,387	-0,827	-0,007	-0,003
Hydrofoil modified	2,469	-0,146	-1,810	7,803	-2,309	-2,646	—	—	—

Table 7.4: Lift Force F_L (N) values across different Reynolds numbers (Re) using three methodologies.

In general, we can say that at low Reynolds numbers, the PINN models struggle to accurately capture lift and drag force due to the sensitivity of laminar flows to boundary conditions and small features like boundary layer attachment. In contrast, at higher Reynolds numbers (Re = 800), where turbulent eddies dominate, the networks align more closely with CFD results, but keeping a high error value. Probably because large-scale flow structures are easier for PINN models to approximate, and boundary condition errors have less influence on overall behavior.

7.6 Vortex Analysis Comparison

We calculated the vorticity over the U-Net architecture, for a Reynolds number Re = 800, under the conditions detailed in section 6.1, shows the most physically accurate behavior according to previous results. As shown in Figure 7.34, a small amount of vorticity is observed near the lateral boundaries of the domain, which was previously mentioned in Section 7.4. This behavior is caused by the interaction between the flow and the applied boundary conditions, resulting in exaggerated rotation in these areas.

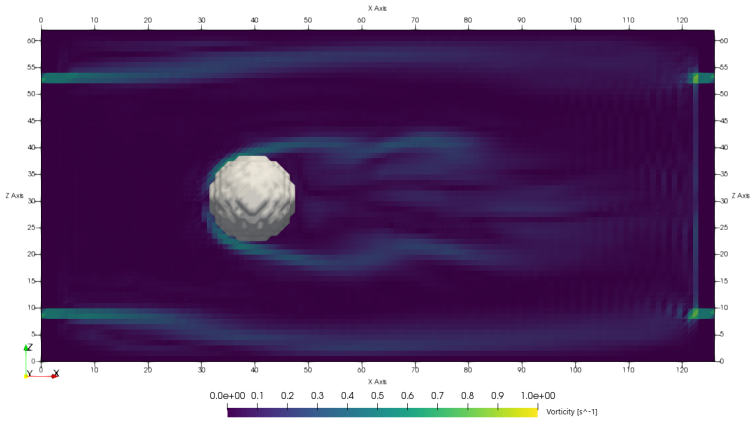


Figure 7.34: U-Net Vorticity in a sphere.

In the wake behind the sphere, the formation of what appears to be a Kármán vortex street is clearly evident, a typical feature of flows at high Reynolds numbers. Interestingly, the vorticity region starts from the front of the sphere as well. When viewed in 3D, the vortices not only form an alternating pattern behind the object but also appear influenced by a rotation curl (7.35) mentioned earlier in 7.3.2, giving them a more chaotic and three-dimensional behavior compared to what would be seen in a CFD simulation. In terms of magnitude, the dominant vorticity in the wake is around 0.6 s^{-1}, indicating moderate rotation around 0.4 rad/s, though local fluctuations can reach higher values due to the flow's instability.

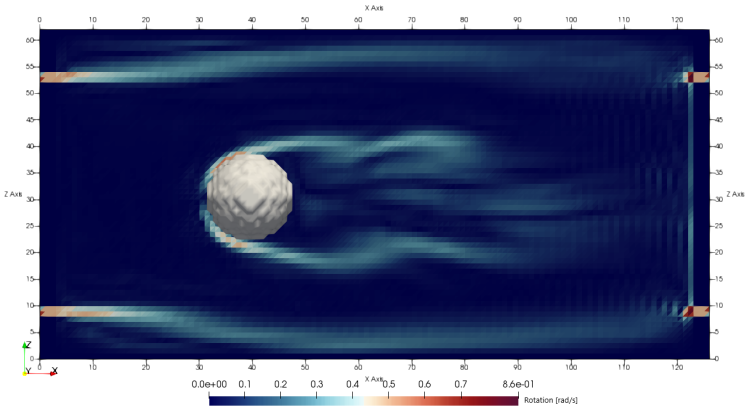


Figure 7.35: U-Net Analysis of Vortex Rotation within a Sphere

7.6.1 Trailing Vortex Study

To analyze trailing vortices under the same fluid conditions as the sphere simulation, the NACA 0018 profile was used. When comparing this simulation to the CFD results, as shown in Figure 7.36, the formation of a trailing vortex is observed. This is evident from the apparent reduction in the distance between parallel streamlines and the vertical distortion in the streamline profile

However, it is important to note that the rotation is minimal due to the low flow velocity of 1 m/s and the fact that the profile was aligned at an angle of attack of 0°. This configuration does not generate a significant pressure difference between the upper and lower surfaces of the profile, contributing to the absence of a significant lift force, which is the main cause of such vortices [99].

In the U-Net simulation as shown in Figure 7.37, while there are some similarities in the formation of trailing vortices, they are clearly affected by the non-physical streamline rotations previously mentioned in Section 7.3.2. This makes it difficult to determine whether the vortex is generated exclusively by the pressure difference or influenced by the limitations of the numerical model used.

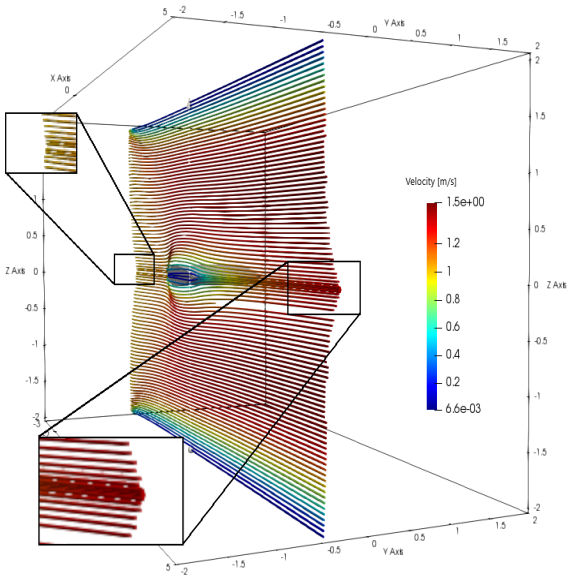


Figure 7.36: CFD Trailing Vortex on NACA 0018.

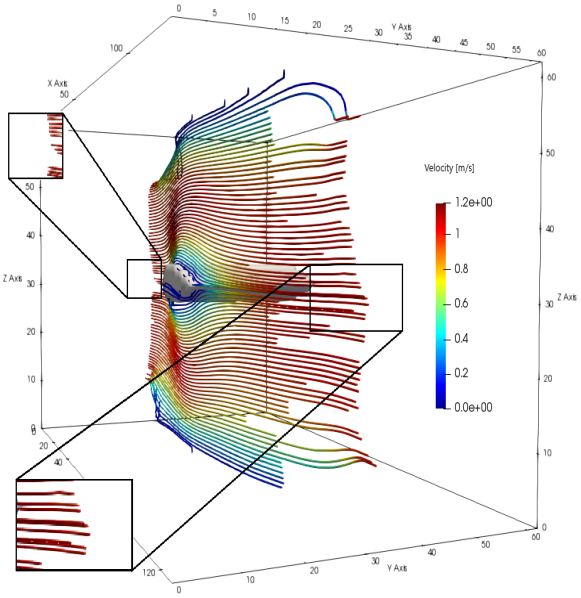


Figure 7.37: U-Net Trailing Vortex on NACA 0018.

7.6.2 Divergence Study

One of the possible causes of this behavior may be the divergence. Figure 7.38 shows that the divergence is not zero throughout the domain $\nabla \times v \neq 0$, suggesting that the fluid is not perfectly incompressible. This effect is more pronounced in regions where turbulent flow is present, particularly in the wake of the sphere. In these areas, the fluid diverges and expands, reaching values up to 0.17 s^{-1} (yellow).

These small values accumulate over time in specific areas, generating artifacts, such as low-density regions or even vacuum effects in the simulation. These artifacts are a result of fluid expansion in these regions. The fluid tends to contract in the wake and in front of the sphere, with divergence values reaching as low as -1.8 s^{-1} (purple). These concentrations can lead to undesirable effects such as local pressure increases, fluid accumulation, or numerical instability in certain areas of the domain.

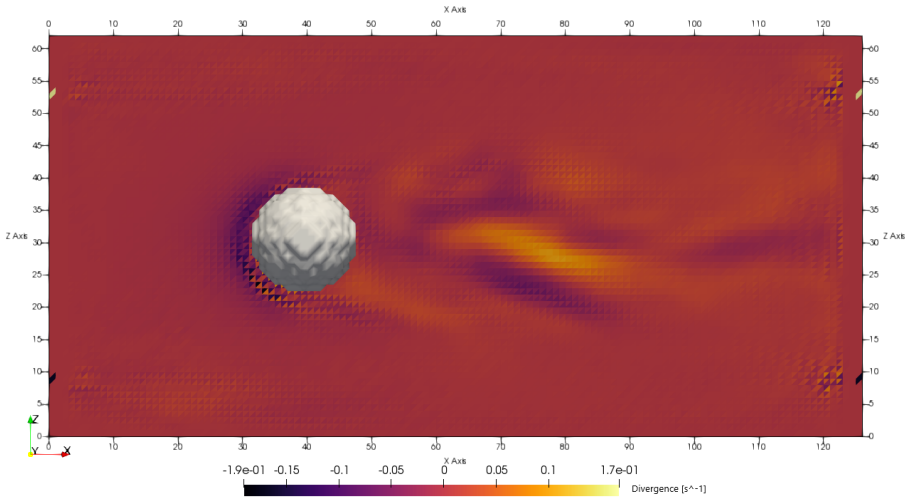


Figure 7.38: U-Net Divergence in a sphere.

7.7 Computational Time Comparison

The Table 7.5 shows the comparative computational times for different domain sizes in voxels for the Pruned-UNet and U-Net architectures. This comparison aims to verify the computational cost as the fluid domain increases.

	128x64x64		160x96x96		256x64x64		256x128x128	
Architecture	**FPS**	**Seconds**	**FPS**	**Seconds**	**FPS**	**Seconds**	**FPS**	**Seconds**
Pruned-UNet	18	28	7	71	7	71	2	250
U-Net	9	56	3	167	5	100	-	-

Table 7.5: FPS and execution time comparison for different resolutions between Pruned-UNet and U-Net.

In general, Pruned-UNet is considerably quickerthan U-Net in across all resolutions, achieving higher FPS with 500 and 400 iterations, respectively. At (128x64x64) resolution, Pruned-UNet achieves 18 FPS compared to U-Net's 9 FPS. As resolution increases, performance decreases for both, but Pruned-UNet remains faster, reaching 2 FPS at (256x128x128), whereas U-Net exceeded memory limits after 20 iterations on an Nvidia RTX 3050. In comparison, CFD simulations for the same fluid domain take approximately 10 minutes for basic shapes and **30 minutes** for torpedo-shaped UV on the same hardware.

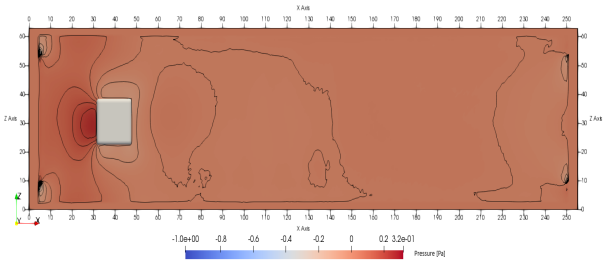


Figure 7.39: U-Net 256x64x64 domain size.

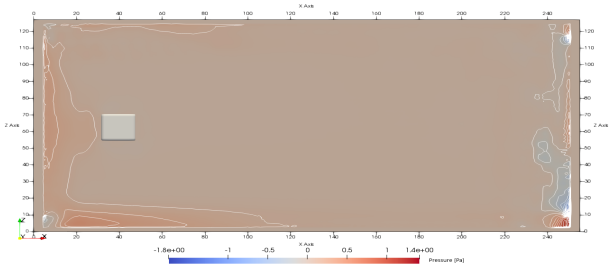


Figure 7.40: Pruned-UNet 256x128x128 domain size.

Note: Although one of the reasons for increasing the fluid domain is to reduce the pressure accumulation effects on the lateral and upper walls relative to the pressure exerted on the object, neither architecture can generalize well under different domain sizes, not seen during the training. The previous exercise was conducted solely to verify computational times.

7.8 Boundary Removal in PINN Simulation

As previously mentioned, one of the issues observed during the simulation was the accumulation of pressure at the fluid boundaries. To address this, an attempt was made to remove the boundaries entirely, to simulate an object immersed in an infinite fluid domain. However, this led to numerical instability, which lead to an erratic pressure and velocity behavior, with the fluid accumulating in the center of the domain, as shown in Figure 7.41.

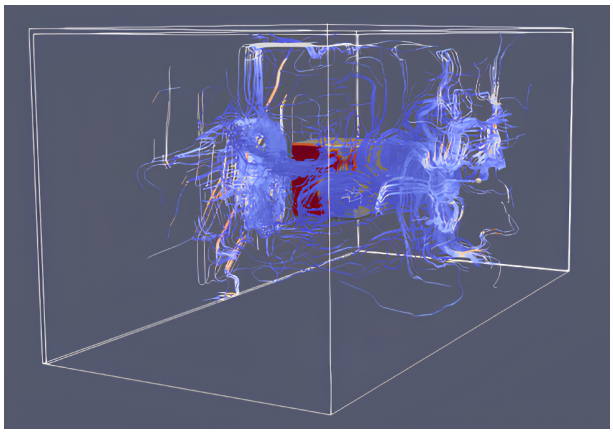


Figure 7.41: U-Net simulation after boundary removal.

7.9 UV Torpedo-Shaped UV Evaluation

As discussed earlier, torpedo-shaped forms are the most common in UV due to their streamlined design, which reduces the separation region at the rear of the vehicle. Hence, the results offered for the PINN architecture are relevant in Marine Robotics. In this section, we analyzed how this shape, mentioned in subsection 4.3.1, behaves under the experimental conditions described in 6.1, both in Surge oriented (0°, aligned with the flow) and Heave oriented (90°, perpendicular to the flow).

7.9.1 Streamline Flow Analysis

As done previously, the main flow analysis was conducted using the U-Net architecture with Re = 1750[1], and streamlines were used to verify the behavior of the flow of a fully submerged torpedo shape UV.

Figure 7.42 shows the formation of a boundary layer in the upper and lower domains of the fluid, which extends further in the U-Net simulation than in CFD (Figure 7.43)—approximately 75 cm (12 voxels) and 50 cm, respectively. Additionally, the boundary layer on the surface of the UV body is larger in the U-Net simulation, with no clear gradual increase in size as seen in CFD as the theory states [12, 15, 11].

Instead, it remains fairly uniform throughout the flow along the vehicle, with both simulations showing the same stagnation point in the front face of the AUV. Small deviations in the streamlines surrounding the vehicle are also observed in the U-Net simulation, which do not correspond to any physical properties.

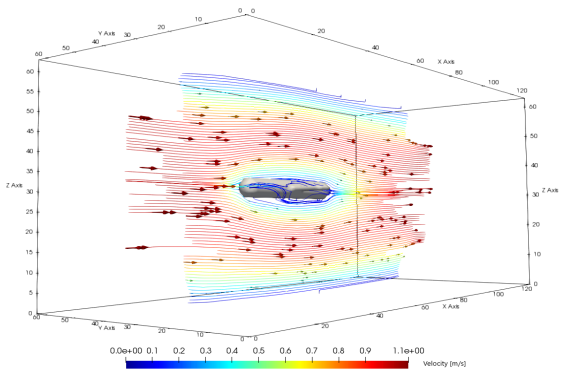


Figure 7.42: Streamlines Torpedo-Shaped UV U-Net architecture.

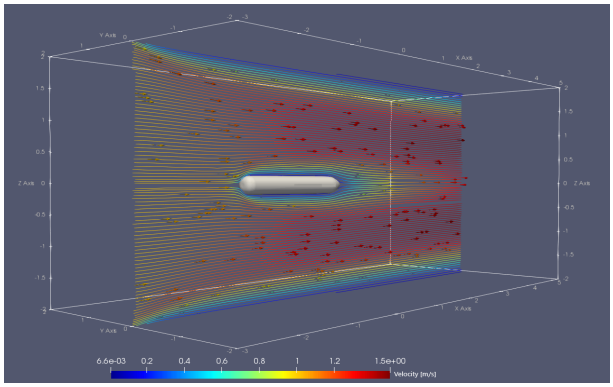


Figure 7.43: Streamlines Torpedo-Shaped UV CFD.

7.9.2 Drag and Lift on Torpedo-Shaped UV

Based on the lift and drag generated, we observe that, similar to the other shapes analyzed in 7.5, the fluid behaves inversely to what is expected based on CFD results. Specifically, drag (Tables 7.6 and 7.7) increases as the Reynolds number rises (i.e., as viscosity decreases), with smaller errors observed at higher Reynolds numbers.

	Pruned-Unet			U-Net			CFD		
Reynolds Number	**1.4**	**175**	**1750**	**1.4**	**175**	**1750**	**1.4**	**175**	**1750**
Torpedo-shaped UV (0°)	1,053	3,167	2,055	12,931	2,021	3,142	73,458	0,654	0,116

Table 7.6: Drag force calculation comparison on a Torpedo Shape UV (0°).

	Pruned-Unet			U-Net			CFD		
Reynolds Number	**0,2**	**25**	**250**	**0,2**	**25**	**250**	**0,2**	**25**	**250**
Torpedo-shaped UV (90°)	7,156	13,292	7,455	101,46	17,039	24,333	646,507	6,009	1,529

Table 7.7: Drag force calculation comparison on a Torpedo Shape UV in Heave oriented (90°).

[1]The Reynolds number varies with the UV orientation characteristic length L, while fluid conditions in 6.1 remain constant.

When comparing the UV orientation relative to the flow, for both Surge and Heave orientations against the fluid motion, the drag error calculated by the U-Net architecture is smaller than that of Pruned-UNet. On average, drag given by the U-Net calculations is 11.61 and 8.32 times larger than CFD for surge and heave, respectively.

	Pruned-Unet			U-Net			CFD		
Reynolds Number	**1.4**	**175**	**1750**	**1.4**	**175**	**1750**	**1.4**	**175**	**1750**
Torpedo-shaped UV (0°)	-0,081	-1,696	-1,76	-0,468	-1,763	-1,697	0,327	0,015	0,001

Table 7.8: Lift force calculation comparison on a Torpedo Shape UV (0°).

	Pruned-Unet			U-Net			CFD		
Reynolds Number	**0,2**	**25**	**250**	**0,2**	**25**	**250**	**0,2**	**25**	**250**
Torpedo-shaped UV (90°)	0,426	-0,431	-1,797	-3,224	-0,966	-1,06	-0,146	0,005	0,006

Table 7.9: Lift force calculation comparison on a Torpedo Shape UV in Heave oriented (90°).

Regarding lift force, as the tables (7.8 and 7.9) show, the results are similar to those observed earlier in 7.5, with values much closer to the theoretical expectation of zero, given the symmetry of the object shape relative to the fluid.

7.9.3 Vortex Development on Torpedo-Shaped UV

When analyzing vortex formation, we once again observe a vortex formation along the upper and lower boundaries of the fluid domain 7.44. Additionally, vorticity forms around the UV diving fins, with a magnitude of approximately $2\,s^{-1}$ in the U-Net simulation. In contrast, CFD shows a more clear vortex formation due to slight twisting of the streamlines, with a maximum vorticity of $14\,s^{-1}$ at the tip of the diving fin forming a trailing vortex, approximately seven times greater than the values predicted by U-Net.

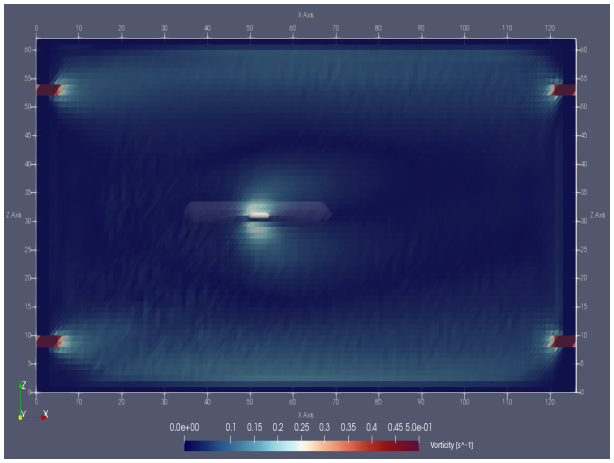


Figure 7.44: U-Net Vortex Formation on a UV.

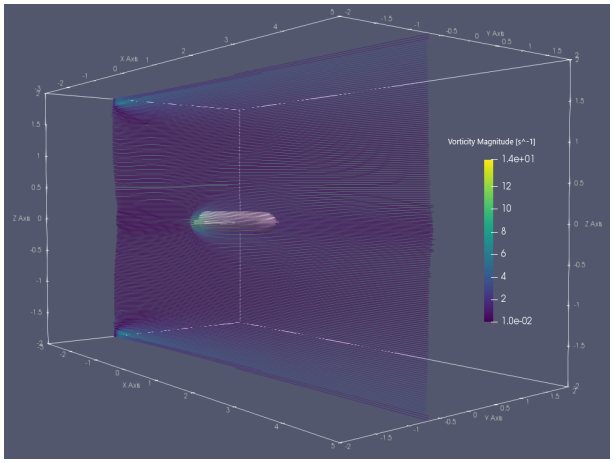


Figure 7.45: CFD Vortex Formation on a UV.

Note: In Heave oriented motion 0° related to the fluid motion, no PINN architecture could represent vortex formation, as the fluid passes through the glider due to its small diameter, this eliminates vortex generation.

7.10 Summary of the Results

In this section, the summary of the results derived from the comparison between the Pruned-UNet and U-Net architectures with the baseline CFD simulations are presented across the different physical parameter representations. The findings highlight the strengths and limitations of each PINN architecture in replicating the hydrodynamic behaviors. Throughout the evaluation, the CFD simulations served as a baseline.

1. Training: Both the Pruned-UNet and U-Net architectures showed significant spikes in the Compound Loss Function $\mathcal{L}$ during early training, primarily due to the non-convex nature of solving the Navier-Stokes equations. Discontinuities, vortex shedding, and separation zones contributed to instability in the loss trajectory. The Adam optimizer seems to struggled to maintain a smooth descent path, especially with varying object shapes and boundary interactions.

2. Convergence: Pruned-UNet and U-Net showed similar convergence behavior, with low divergence values and minimal incompressibility errors. However, the full U-Net demonstrated a higher ability in capturing finer flow details.

3. Fluid flow behavior: The simulations revealed significant differences in vortex structure capture between the two architectures. U-Net was more accurate in replicating complex phenomena, such as Kármán vortex streets and boundary layer development in turbulent flow configurations. The PINN methodology struggled with behaviors typical on rough surfaces where higher-angle flow separation occurs, the simulation seems unable to simulate behavior on rough surfaces despite the non-uniform surface shape of simplifications.

   In laminar flows, Pruned-UNet exhibited greater numerical dissipation, suppressing some flow characteristics and slowing the recovery of velocity in the wake of objects like the sphere. Both architectures tended to develop turbulence under fluid conditions similar to CFD models. Although both models generally exhibit fluid-like behavior, the Pruned-Unet is more likely to violate the conditions $\nabla \cdot (\nabla \times \vec{a}) = 0$ and $\nabla \times (\nabla q) = 0$ of incompresibility specially under low Reynolds numbers.

4. Pressure distribution: A general pressure accumulation was observed along the fluid domain, which could lead to buildup near the fluid boundaries after convergence to a stable value. This describes difficulties in managing boundary conditions effectively.

   The analysis of the flat plane and hydrofoil revealed pressure variations and vortex formation influenced by the object orientation. In both simulations PINN nad CFD, consistent flow patterns were identified, with accurate representation of high-pressure zones and vortex formation in the wake. Although minor differences in velocity recovery were observed between models, the results show consistent flow and pressure distributions near the object surface, demonstrating that the PINN methodology effectively represents hydrodynamic phenomena across different flow regimes and fluid conditions.

5. Lift and Drag Forces: The simulations showed that the PINN methodology deviates more from CFD results in laminar flows at low Reynolds numbers. In contrast, at higher Re, the results aligned more closely with CFD $k-\omega SST$ models, likely due to sensitivity to boundary conditions and limited ability to represent thin boundary layers.

6. Computational Time and Efficiency: The comparative analysis indicated that Pruned-UNet is significantly more computationally efficient across all resolutions, making it suitable for real-time applications. While U-Net provides greater accuracy for fine details like vortex formation, its higher computational cost limits its use in real-time scenarios, but provides closer physical representation results. However, both architectures must be rectified to match the values measured by CFD.

7. Fluid Domain Limitations: One limitation observed was the inability of the PINN methodology to generalize effectively to fluid domains of varying sizes, not seen during training.

8. Fluid Behavior on a Torpedo Shape UV: Simulations of torpedo-shaped bodies indicated that U-Net based architectures can represent the boundary layer, although it does not progressively develop from stagnation points. The trailing vortex formation around the diving fins was also captured. However, in perpendicular flows (heave), neither architecture accurately simulated vortex formation, likely due to the small characteristic length scales of the simulations.

CHAPTER 8

Conclusions and Further Work

Contents

8.1 Conclusions

This thesis presents, for the first time to the best of the author's knowledge (as of Nov. 2024), the successful implementation of real-time AI-based specifically a PINN fluid simulation techniques for fully submerged objects in confined environments, such as wind tunnels, for Newtonian incompressible isothermal fluids. Motivated by potential applications in marine robotics, where real-time computations are critical, this work achieves a balance between accurate hydrodynamic behavior representation and computational efficiency. The main contribution is a PINN-based simulation methodology capable of replicating real-time hydrodynamic phenomena, including flow separation, vortex formation, drag, and lift forces, across various flow regimes and fluid conditions. Demonstrating its potential as an alternative to over simplified fluid models or computationally expensive fluid simulations under relatively common hardware conditions.

A significant contribution of this work was the application of the Pruned-UNet and U-Net volumetric CNN architectures in a PINN based methodology. Pruned-UNet proved to be efficient for real-time applications, offering a balance between speed and physical representation, specially under turbulent regimes. Although more computationally expensive, U-Net is more effective in capturing fine-scale in laminar, turbulent regimes and complex flow structures such vortices, making it more suitable for higher physics representation simulations.

One notable advantage of this PINN-based methodology is its ability to generalize effectively across diverse shapes and fluid conditions, including those unseen during training, by relying solely on physics-based constraints. Unlike other approaches, which typically require specific PINN models and extensive data for each shape and fluid condition, this method achieves flexibility without reference data.

To support this methodology, a voxelized underwater shape dataset was created, offering a general approach to minimize shape detail loss during ths shape simplification. This dataset proved important in validating the PINN models against computational fluid dynamics (CFD) baselines, providing a logical approach to evaluate to generate shapes for the PINN model performance across various geometries and flow conditions.

However, a voxel-based representation limits the handling of angles of attack, reducing accuracy in different displacement motions and making it difficult to capture fine geometric details, such as high angles and thin surfaces. Eventually, this leads to less accurate results. Therefore, further robustness is needed, particularly in cases involving sharp features or complex geometries, where the voxelized representation struggles to capture fine details, such as thin objects or intricate surfaces, allowing fluid to pass through rather than interact with the object.

Despite the discrepancies in hydrodynamic variable calculations, such as drag, lift, and vorticity, between the PINN methodology and CFD results, particularly in laminar flows where curl effects have a significant role. The current state of PINN demonstrates potential for qualitatively reproducing different fluid types. This capability could be applied in fields like video games or the audiovisual industry.

One limitation of this methodology is its inability to accurately represent fluid motion at velocities above 1 m/s (or 2 knots) due to numerical instability, leading to unpredictable behavior. Another constraint is the mandatory use of CUDA for processing simulations, which currently restricts its application to machines equipped with an Nvidia GPU. Similarly, it presents scalability issues with domain sizes not seen during training, resulting in even greater pressure accumulation than in smaller domains and potential GPU memory limitations.

It can be said that this thesis successfully demonstrates the potential of AI-based fluid simulations in robotics and marine environments, particularly in scenarios where traditional CFD methods are computationally prohibitive. For practical applications, additional improvements are necessary to enhance quantitative accuracy in complex flows and to optimize model performance under varying conditions. Despite some limitations in handling low Reynolds number flows and boundary conditions, the PINN methodology shows promise in qualitatively and quantitatively representing fluid dynamics.

8.2 Future work

Future work should focus on incorporating Neumann boundary conditions to better manage pressure and velocity gradients at the boundaries of the fluid domain, as are use in CFD simulations [93]. This would help attenuate boundary layer effects and reduce the artificial pressure accumulation observed at the fluid boundaries. Addressing the pressure accumulation issue in boundary layers could involve adaptive boundary condition adjustments or pressure diffusion techniques, both of which may help mitigate the boundary effects seen in simulations.

To enhance the performance of the model across different shapes and geometries, a multi-objective optimization approach could be explored, similar to the one presented in [101] to escape from local minimums. This would allow for a balanced adjustment of multiple refined loss functions, improving the ability of the model to generalize fluid interactions across various domains while maintaining computational efficiency. To overcome the limitations in handling angles of attack, incorporating smoother boundary representations or higher-order interpolation methods could help the model better capture flow separation and lift dynamics, especially in scenarios with sharp flow direction changes, like the one presented in [79]. Similarly, the PINN methodology could also be compared with other CFD turbulence models, such LES models or other RAS models.

As noted, the current model struggles with very thin objects, where the fluid tends to pass through the geometry rather than interacting properly. Implementing refined meshing techniques or using higher-resolution grids could improve the accuracy of fluid-structure interactions for such objects, but will increment the computational requirements. One promising direction is to replace the voxelized grid with a graph-based or adaptive mesh, as the ones seen in [75, 76]. This would allow for finer resolution near complex surfaces and thin structures, enabling the model to handle these details more effectively. However, this approach may reduce the capacity of the model to generalize to different geometries. On the other hand, a simpler possibility is to integrate correction factors as the ones mentioned in [99] for achieving the simulation of Physical variables in a not confined environments of fluids (e.g., deep ocean) could expand the scope of simulation without increasing the computational cost, since it does not have to deal with a loss function based on the Neumann boundary conditions.

In the future, since there is no accurate ground truth about pressure and velocity behavior on the surface of an underwater vehicle, it could be useful to implement a sensitive skin, such as a piezoelectric sensor cover around the hull, as the Figure 8.1 tries to represent. This would allow data to be collected directly on the vehicle and train a PINN with real information obtained in the operating environment.

Figure 8.1: Piezoelectric sensitive skin concept.

As the next step, the predictions of the model could be tested further by integrating them into real-time simulators like Holoocean or StoneFish [102, 37] and evaluating fluid behavior while interacting with moving objects, such as UVs or robotic components like propellers and propeller wake vortices. This would help validate its application in marine simulators and similar environments. Future research could also explore the integration of external flows, higher Reynolds number regimes, and further optimization of the PINN architecture to handle complex boundary conditions and larger domains. Overall, this thesis lays a solid foundation for efficient, scalable, and physically plausible fluid simulations, which can drive advancements in marine robotics and other fluid engineering or science fields.

Bibliography

[1] Mabel M. Zhang et al. "DAVE Aquatic Virtual Environment: Toward a General Underwater Robotics Simulator". en. In: *2022 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*. Singapore: IEEE, Sept. 2022, pp. 1–8. ISBN: 978-1-66541-689-4. DOI: 10.1109/AUV53081.2022.9965808.

[2] Jack Collins et al. "A Review of Physics Simulators for Robotic Applications". en. In: *IEEE Access* 9 (2021), pp. 51416–51431. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3068769.

[3] Michael V. Jakuba. *Modeling and control of an autonomous underwater vehicle with combined foil/thruster actuators*. en. Woods Hole, MA: Massachusetts Institute of Technology and Woods Hole Oceanographic Institution, 2003. DOI: 10.1575/1912/2460.

[4] João Victor Nunes De Sousa et al. "On the Study of Autonomous Underwater Vehicles by Computational Fluid-Dynamics". en. In: *Open Journal of Fluid Dynamics* 10.01 (2020), pp. 63–81. ISSN: 2165-3852, 2165-3860. DOI: 10.4236/ojfd.2020.101005.

[5] Jerry Tessendorf. *Gilligan: A prototype framework for simulating and rendering maritime environments*. School of Computing, Clemson University. Feb. 2017. URL: https://people.cs.clemson.edu/~jtessen/papers_files/simdoc.pdf.

[6] Cem Yuksel. "Real-time Water Waves with Wave Particles". PhD thesis. Texas AM University, 2010.

[7] Dan Koschier et al. "A Survey on SPH Methods in Computer Graphics". en. In: *Computer Graphics Forum* 41.2 (May 2022), pp. 737–760. ISSN: 0167-7055, 1467-8659. DOI: 10.1111/cgf.14508.

[8] Nuttapong Chentanez and Matthias Müller. "Real-time simulation of large bodies of water with small scale details". en. In: *Eurographics/ ACM SIGGRAPH* (2010). DOI: 10.2312/SCA/SCA10/197-206.

[9] Matthias Müller et al. "Real time physics: class notes". en. In: *ACM SIGGRAPH 2008 classes*. Los Angeles California: ACM, Aug. 2008, pp. 1–90. ISBN: 978-1-4503-7845-1. DOI: 10.1145/1401132.1401245.

[10] Nils Wandel, Michael Weinmann, and Reinhard Klein. "Teaching the incompressible Navier–Stokes equations to fast neural surrogate models in three dimensions". en. In: *Phys. Fluids (1994)* 33.4 (Apr. 2021), p. 047117. DOI: 10.1063/5.0047428.

[11] Joseph Katz. *Introductory Fluid Mechanics*. Cambridge, England: Cambridge University Press, Aug. 2010. ISBN: 978-0521192453.

[12] Merle C Potter, David C Wiggert, and Bassem H Ramadan. *Mechanics of fluids*. en. South Melbourne, VIC, Australia: Cengage Learning, Jan. 2011. ISBN: 978-0495667735.

[13] J. Gordon Leishman. *Introduction to Aerospace Flight Vehicles*. First edition. Embry-Riddle Aeronautical University, Aug. 2024. ISBN: 979-8-9852614-0-0.

[14] Joseph Katz. *Automotive Aerodynamics*. First Edition. John Wiley & Sons, Ltd, 2016. ISBN: 978-1119185727.

[15] John M. Cimbala Yunus A. Çengel. *Fluid mechanics: fundamentals and applications*. Fourth Edition. McGraw-Hill, 2006. ISBN: 978-1259696534.

[16] Ricardo Mendonça et al. "Kelpie: A ROS-Based Multi-robot Simulator for Water Surface and Aerial Vehicles". In: *2013 IEEE International Conference on Systems, Man, and Cybernetics*. 2013, pp. 3645–3650. DOI: 10.1109/SMC.2013.621.

[17] Volker Bertram. *Submarine Hull Design*. NTNU – Norwegian University of Science and Technology. 2012. URL: https://www.ntnu.edu/documents/20587845/1266707380/2012Chennai_SubmarineDesign.pdf/9bb180be-a08d-48ea-af4e-7e5e19210d3b.

[18] Roy Burcher and Louis J. Rydill. *Concepts in Submarine Design*. First Edition. Vol. 1. Cambridge University Press, July 1994. ISBN: 978-0521559263.

[19] Ivan Loncar et al. "MARUS - A Marine Robotics Simulator". en. In: *OCEANS 2022, Hampton Roads*. Hampton Roads, VA, USA: IEEE, Oct. 2022, pp. 1–7. ISBN: 978-1-66546-809-1. DOI: 10.1109/OCEANS47191.2022.9976969.

[20] Volker Bertram. *Practical Ship Hydrodynamics*. Second Edition. Elsevier, 2012. ISBN: 978-1483299716.

[21] Phil Stoffer. *Introduction to Oceanography, Miracosta College*. web writer: pstoffer@miracosta.edu. 2024. URL: https://gotbooks.miracosta.edu/oceans/index.html.

[22] Reginaldo Durazo. *Physical Oceanography, Universidad Autonoma de baja California*. As a part of the course in Physical Oceanography for the Universidad autonoma de baja california. 2024. URL: http://rdurazo.ens.uabc.mx/educacion/ocefis/Dinamica_de_Ekman.pdf.

[23] Jerry Tessendorf. "Simulating Ocean Water". In: School of Computing, Clemson University. 2004. URL: https://people.computing.clemson.edu/~jtessen/reports/papers_files/coursenotes2004.pdf.

[24] L. S. Jensen and R. Goliáš. "Deep-Water Animation and Rendering". In: *Proceedings of the Game Developer's Conference*. 2001. URL: http://whitenight-games.chez-alice.fr/Docs/Water.pdf.

[25] Christopher J Horvath. "Empirical directional wave spectra for computer graphics". In: *Proceedings of the 2015 Symposium on Digital Production*. New York, NY, USA: ACM, Aug. 2015. DOI: 10.1145/2791261.2791267.

[26] Jingcong Zhang. "Implementation and Applications of Art-directable Ocean Simulation Tools". In: School of Computing, Clemson University. 2018.

[27] Leo H Holthuijsen. *Waves in oceanic and coastal waters*. Cambridge, England: Cambridge University Press, Feb. 2010. DOI: 10.1017/CBO9780511618536.

[28] Stefan Jeschke and Chris Wojtan. "Generalizing Shallow Water Simulations with Dispersive Surface Waves". In: 42.4 (July 2023). ISSN: 0730-0301. DOI: 10.1145/3592098.

[29] E Darles et al. "A survey of ocean simulation and rendering techniques in computer graphics". en. In: *Comput. Graph. Forum* 30.1 (Mar. 2011), pp. 43–60.

[30] Stefan Jeschke et al. "Water surface wavelets". en. In: *ACM Transactions on Graphics* 37.4 (Aug. 2018), pp. 1–13. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3197517.3201336.

[31] Reagan Samuel Burke. "Rigid Body Dynamics of Ship Hulls via Hydrostatic Forces Calculated From FFT Ocean Height Fields". en. In: (). School of Computing, Clemson University.

[32] Alexandra L. Zheleznyakova. "Physically-based method for real-time modeling of ship motion in irregular waves". In: *Ocean Engineering* 195 (2020), p. 106686. ISSN: 0029-8018. DOI: 10.1016/j.oceaneng.2019.106686.

[33] Doug Perrault et al. "Sensitivity of AUV added mass coefficients to variations in hull and control plane geometry". In: *Ocean Engineering* 30 (Apr. 2003), pp. 645–671. DOI: 10.1016/S0029-8018(02)00041-0.

[34] Hamid Zeraatgar, Aliasghar Moghaddas, and Kazem Sadati. "Analysis of surge added mass of planing hulls by model experiment". In: *Ships and Offshore Structures* 15 (Aug. 2019), pp. 1–8. DOI: 10.1080/17445302.2019.1615705.

[35] Hampus Tober. "Evaluation of Drag Estimation Methods for Ship Hulls". Second Cycle. Stockholm, Sweden: KTH, School of Engineering Sciences (SCI), Mechanics, 2020.

[36] Thanh-Long Le and Duc-Thong Hong. "Computational Fluid Dynamics Study of the Hydrodynamic Characteristics of a Torpedo-Shaped Underwater Glider". en. In: *Fluids* 6.7 (July 2021), p. 252. ISSN: 2311-5521. DOI: 10.3390/fluids6070252.

[37] Patryk Cieslak. "Stonefish: An advanced open-source simulation tool designed for marine robotics, with a ROS interface". In: *OCEANS 2019 - Marseille*. Marseille, France: IEEE, June 2019. DOI: 10.1109/OCEANSE.2019.8867434.

[38] Thor Fossen. "Handbook of marine craft hydrodynamics and motion control". en. In: (Apr. 2011). DOI: 10.1002/9781119994138.

[39] Seong-Keon Lee et al. "Evaluation of the added mass for a spheroid-type unmanned underwater vehicle by vertical planar motion mechanism test". In: *International Journal of Naval Architecture and Ocean Engineering* 3.3 (Sept. 2011), pp. 174–180. DOI: 10.2478/ijnaoe-2013-0060.

[40] Josefine Severholt. "Generic 6-DOF Added Mass Formulation for Arbitrary Underwater Vehicles based on Existing Semi-Empirical Methods". KTH, School of Engineering Sciences (SCI), Mechanics, 2017.

[41] R A Gingold and J J Monaghan. "Smoothed particle hydrodynamics: theory and application to non-spherical stars". In: *Mon. Not. R. Astron. Soc.* 181.3 (Dec. 1977), pp. 375–389. DOI: 10.1093/mnras/181.3.375.

[42] Nghia Truong et al. "Particle Merging-and-Splitting". In: *IEEE Transactions on Visualization and Computer Graphics* (2021). ISSN: 1077-2626. DOI: 10.1109/TVCG.2021.3093776.

[43] Miles Macklin et al. "Unified particle physics for real-time applications". In: *ACM Trans. Graph.* 33.4 (July 2014). ISSN: 0730-0301. DOI: 10.1145/2601097.2601152.

[44] Emmanouil Angelidis et al. "Gazebo Fluids: SPH-based simulation of fluid interaction with articulated rigid body dynamics". en. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Kyoto, Japan: IEEE, Oct. 2022, pp. 11238–11245. ISBN: 978-1-66547-927-1. DOI: 10.1109/IROS47612.2022.9982036.

[45] Nicolas Gartner et al. "Can smoothed particle hydrodynamics simulate physically realistic movements of underwater vehicles?" en. In: *Advanced Robotics* 37.20 (Oct. 2023), pp. 1283–1300. ISSN: 0169-1864, 1568-5535. DOI: 10.1080/01691864.2023.2263046.

[46] 2001 C. C. Mei. *Advanced Environmental Fluid Mechanics*. Mar. 2001. URL: http://web.mit.edu/fluids-modules/www/basic_laws/1-1-LagEul.pdf.

[47] John David Anderson. *Computational Fluid Dynamics*. en. McGraw-Hill series in mechanical engineering. New York, NY: McGraw-Hill Professional, Feb. 1995. ISBN: 0-07-113210-4.

[48] J U Brackbill and H M Ruppel. "FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions". en. In: *J. Comput. Phys.* 65.2 (Aug. 1986), pp. 314–343. DOI: 10.1016/0021-9991(86)90211-1.

[49] Mark Harris. "Fast Fluid Dynamics Simulation on the GPU". In: *GPU Gems*. Ed. by Randima Fernando. Addison-Wesley, 2004. Chap. 38, pp. 637–665. URL: https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/.

[50] Simon Clavet, Philippe Beaudoin, and Pierre Poulin. "Particle-based viscoelastic fluid simulation". In: *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA: ACM, July 2005. DOI: 10.1145/1073368.1073400.

[51] T R Hagen et al. "Visual simulation of shallow-water waves". en. In: *Simul. Model. Pract. Theory* 13.8 (Nov. 2005), pp. 716–726. DOI: 10.1016/j.simpat.2005.08.006.

[52] J. Kim et al. "Fast GPU Computation of the Mass Properties of a General Shape and Its Application to Buoyancy Simulation". In: *The Visual Computer* 22 (2006), pp. 856–864. DOI: 10.1007/s00371-006-0071-x.

[53] Adrien Treuille, John Lewis, and Zoran Popovic. "Model Reduction for Real-Time Fluids". In: *ACM Transactions on Graphics* 25.3 (2006), pp. 826–834. DOI: 10.1145/1141911.1141962.

[54] H. Cords. "Mode-Splitting for Highly Detailed, Interactive Liquid Simulation". In: *GRAPHITE '07: Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia*. 2007, pp. 265–272. DOI: 10.1145/1321261.1321309.

[55] Keenan Crane, Ignacio Llamas, and Sohaib Tariq. "Real-Time Simulation and Rendering of 3D Fluids". In: *GPU Gems 3*. Ed. by Hubert Nguyen. Addison-Wesley, 2007. Chap. 30, pp. 633–675.

[56] Eric Perlman et al. "Data exploration of turbulence simulations using a database cluster". In: *Proceedings of the 2007 ACM/IEEE conference on Supercomputing*. Reno Nevada: ACM, Nov. 2007. DOI: 10.1145/1362622.1362654.

[57] Luciano Castillo, Andrew Pollard, and Luminita Danaila, eds. *Whither turbulence and big data in the 21st century?* en. 1st ed. Cham, Switzerland: Springer International Publishing, Aug. 2016. DOI: 10.1007/978-3-319-41217-7.

[58] Andrei N. Kolmogorov. "The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 434 (1991), pp. 13–9. DOI: 10.1098/rspa.1991.0075.

[59] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. "Machine learning for fluid mechanics". en. In: *Annu. Rev. Fluid Mech.* 52.1 (Jan. 2020), pp. 477–508. DOI: 10.1146/annurev-fluid-010719-060214.

[60] P Kutler and U Mehta. "Computational aerodynamics and artificial intelligence". In: *17th Fluid Dynamics, Plasma Dynamics, and Lasers Conference*. Snowmass,CO,U.S.A.: American Institute of Aeronautics and Astronautics, June 1984.

[61] M W M G Dissanayake and N Phan-Thien. "Neural-network-based approximations for solving partial differential equations". en. In: *Commun. Numer. Methods Eng.* 10.3 (Mar. 1994), pp. 195–201. DOI: 10.1002/cnm.1640100303.

[62] I E Lagaris, A Likas, and D I Fotiadis. "Artificial neural networks for solving ordinary and partial differential equations". en. In: *IEEE Trans. Neural Netw.* 9.5 (1998), pp. 987–1000. DOI: 10.1109/72.712178.

[63] R González-García, R Rico-Martínez, and I G Kevrekidis. "Identification of distributed parameter systems: A neural net based approach". en. In: *Comput. Chem. Eng.* 22 (Mar. 1998), S965–S968. DOI: 10.1016/S0098-1354(98)00191-4.

[64] Maziar Raissi and George Em Karniadakis. "Hidden physics models: Machine learning of nonlinear partial differential equations". en. In: *J. Comput. Phys.* 357 (Mar. 2018), pp. 125–141. DOI: 10.1016/j.jcp.2017.11.039.

[65] Onofrio Semeraro et al. "Qualitative dynamics of wave packets in turbulent jets". In: *Phys. Rev. Fluids* 2.9 (Sept. 2017). DOI: 10.1103/PhysRevFluids.2.094605.

[66] Nils Thuerey et al. "Deep learning methods for Reynolds-averaged Navier–stokes simulations of airfoil flows". en. In: *AIAA J.* 58.1 (Jan. 2020), pp. 25–36. DOI: 10.2514/1.J058291.

[67] Yin Cheng et al. "ThermalNet: A deep reinforcement learning-based combustion optimization system for coal-fired boiler". en. In: *Eng. Appl. Artif. Intell.* 74 (Sept. 2018), pp. 303–311. DOI: 10.1016/j.engappai.2018.07.003.

[68] Pantelis R Vlachas et al. "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks". en. In: *Proc. Math. Phys. Eng. Sci.* 474.2213 (May 2018), p. 20170844. DOI: 10.1098/rspa.2017.0844.

[69] Byungsoo Kim et al. "Deep fluids: A generative network for parameterized fluid simulations". en. In: *Comput. Graph. Forum* 38.2 (May 2019), pp. 59–70. DOI: 10.1111/cgf.13619.

[70] You Xie et al. "tempoGAN". en. In: *ACM Trans. Graph.* 37.4 (Aug. 2018), pp. 1–15. DOI: 10.1145/3197517.3201304.

[71] Arvind T Mohan et al. "Embedding hard physical constraints in neural network coarse-graining of 3D turbulence". In: (Jan. 2020). arXiv: 2002.00021 [physics.comp-ph].

[72] M Raissi, P Perdikaris, and G E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". en. In: *J. Comput. Phys.* 378 (Feb. 2019), pp. 686–707. DOI: 10.1016/j.jcp.2018.10.045.

[73] Junhyuk Kim and Changhoon Lee. "Deep unsupervised learning of turbulence for inflow generation at various Reynolds numbers". en. In: *J. Comput. Phys.* 406.109216 (Apr. 2020), p. 109216. DOI: 10.1016/j.jcp.2019.109216.

[74] Nicholas Geneva and Nicholas Zabaras. "Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks". en. In: *J. Comput. Phys.* 403.109056 (Feb. 2020), p. 109056. DOI: 10.1016/j.jcp.2019.109056.

[75] Tobias Pfaff et al. "Learning mesh-based simulation with graph networks". In: (Oct. 2020). arXiv: 2010.03409 [cs.LG].

[76] Davide Roznowicz et al. "Large-scale graph-machine-learning surrogate models for 3D-flowfield prediction in external aerodynamics". en. In: *Adv. Model. Simul. Eng. Sci.* 11.1 (Mar. 2024). DOI: 10.1186/s40323-024-00259-1.

[77] Nils Thuerey et al. *Physics-based Deep Learning*. WWW, 2021. URL: https://physicsbaseddeeplearning.org.

[78] Nils Wandel, Michael Weinmann, and Reinhard Klein. "Learning incompressible fluid dynamics from scratch – towards fast, differentiable fluid models that generalize". In: (June 2020). arXiv: 2006.08762 [cs.LG].

[79] Nils Wandel et al. "Spline-PINN: Approaching PDEs without data using fast, physics-informed Hermite-spline CNNs". In: (Sept. 2021). arXiv: 2109.07143 [cs.LG].

[80] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. "Physics informed deep learning (part II): Data-driven discovery of nonlinear partial differential equations". In: (Nov. 2017). arXiv: 1711.10566 [cs.AI].

[81] Ilias Bilionis. *Introduction to Scientific Machine Learning (Lecture Book)*. These are the lecture notes for ME 539 Introduction to Scientific Machine Learning. By lias Bilionis (ibilion@purdue.edu). 2022. URL: https://predictivesciencelab.github.io/data-analytics-se/index.html#.

[82] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering*. en. 2nd ed. Cambridge, England: Cambridge University Press, May 2022. ISBN: 978-1009098489.

[83] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. "Hidden fluid mechanics: A Navier-Stokes informed deep learning framework for assimilating flow visualization data". In: (Aug. 2018). arXiv: 1808.04327 [cs.CE].

[84] Aditi S Krishnapriyan et al. "Characterizing possible failure modes in physics-informed neural networks". In: (Sept. 2021). arXiv: 2109.01050 [cs.LG].

[85] Franz M Rohrhofer et al. "Data vs. Physics: The apparent Pareto front of physics-informed neural networks". In: *IEEE Access* 11 (2023), pp. 86252–86261. DOI: 10.1109/ACCESS.2023.3302892.

[86] M.A.C. Celis, J.B.V. Wanderley, and M.A.S. Neves. "Numerical simulation of dam breaking and the influence of sloshing on the transfer of water between compartments". In: *Ocean Engineering* 146 (2017), pp. 125–139. ISSN: 0029-8018. DOI: 10.1016/j.oceaneng.2017.09.029.

[87] Martin H. Sadd. "13 - Displacement Potentials and Stress Functions". In: *Elasticity*. Ed. by MARTIN H. SADD. Burlington: Academic Press, 2005, pp. 347–369. ISBN: 978-0-12-605811-6. DOI: 10.1016/B978-012605811-6/50014-2.

[88] Özgün Çiçek et al. "3D U-net: Learning dense volumetric segmentation from sparse annotation". In: (June 2016). arXiv: 1606.06650 [cs.CV].

[89] Zhi-Hua Liu et al. "Numerical simulation and experimental study of the new method of horseshoe vortex control". In: *J. Hydrodynam. B* 22.4 (Aug. 2010), pp. 572–581. DOI: 10.1016/s1001-6058(09)60090-1.

[90] National Committee for Fluid Mechanics Films. *Fluid Mechanics Film Program*. London, England: MIT Press, Jan. 1973. URL: http://web.mit.edu/hml/ncfmf.html.

[91] Christopher A. Eggert and Christopher L. Rumsey. *CFD Study of NACA 0018 Airfoil with Flow Control*. 2017. URL: https://ntrs.nasa.gov/api/citations/20170004500/downloads/20170004500.pdf.

[92] Krzysztof Rogowski, Martin Otto Laver Hansen, and Ryszard Maroński. "Steady and unsteady analysis of NACA 0018 airfoil in vertical-axis wind turbine". In: *J. Theor. Appl. Mech.* (2018), p. 203. DOI: 10.15632/jtam-pl.56.1.203.

[93] Christopher Greenshields and Henry Weller. *Notes on Computational Fluid Dynamics: General Principles*. Reading, UK: CFD Direct Ltd, 2022. ISBN: 978-1399920780. URL: https://doc.cfd.direct/notes/cfd-general-principles/.

[94] F. R. Menter, M. Kuntz, and R. Langtry. "Ten Years of Industrial Experience with the SST Turbulence Model". In: *Proceedings of the 4th International Symposium on Turbulence, Heat and Mass Transfer*. West Redding: Begell House Inc., 2003, pp. 625–632.

[95] SIMSCALE. *K-Omega Turbulence Models*. URL: https://www.simscale.com/docs/simulation-setup/global-settings/k-omega-sst/.

[96] L S Caretto et al. "Two calculation procedures for steady, three-dimensional flows with recirculation". In: *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 60–68. DOI: 10.1007/BFb0112677.

[97] Suhas V. Patankar and D. B. Spalding. "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows". In: *International Journal of Heat and Mass Transfer* 15 (1972), pp. 1787–1806. DOI: http://dx.doi.org/10.1016/0017-9310(72)90054-3.

[98] Joseph Katz. *Race car aerodynamics*. First Edition. Bentley Publishers, 1995. ISBN: 0-8376-0142-8.

[99] Mark Drela. *Flight vehicle aerodynamics*. The MIT Press. London, England: MIT Press, Feb. 2014. ISBN: 978-0262526449.

[100] John Southard. MIT OpenCourseWare. *Flow Past A Sphere Ii: Stokes' Law, The Bernoulli Equation, Turbulence, Boundary Layers, Flow Separation*. 2006. URL: https://ocw.mit.edu/courses/12-090-introduction-to-fluid-motions-sediment-transport-and-current-generated-sedimentary-structures-fall-2006/7841d9b1681d6748fa2f5cbc6d6f1cb2_ch3.pdf.

[101] Binghang Lu, Christian Moya, and Guang Lin. "NSGA-PINN: A multi-objective optimization method for physics-informed neural network training". In: *Algorithms* 16.4 (Apr. 2023), p. 194. DOI: 10.3390/a16040194.

[102] Easton Potokar et al. "HoloOcean: An Underwater Robotics Simulator". In: *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA: IEEE, May 2022. DOI: 10.1109/ICRA46639.2022.9812353.