CITY UNIVERSITY LONDON

Aerodynamic control via Deep Reinforcement Learning

Laerte Adami

City, University of London

MSc in Artificial Intelligence

Supervisor: MICHAEL GARCIA ORTIZ

Submission date: 27/09/2023

Declaration

By submitting this work, I declare that this work is entirely my own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the assessment instructions and any other relevant programme and module documentation. In submitting this work I acknowledge that I have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. I also acknowledge that this work will be subject to a variety of checks for academic misconduct.
Signed:

Laerte Adami
London, United Kingdom
27/09/2023

Abstract

This project explores the application of Deep Reinforcement Learning (DRL) techniques to control and optimise the flow patterns around a 2D square cylinder, an iconic problem in fluid dynamics and aerodynamics. The interaction between flow and bluff bodies, like the square cylinder, is a complex phenomenon with wide-ranging applications in engineering.
In this research, state-of-the-art DRL algorithms are leveraged to develop an intelligent control system that adapts and optimises the flow characteristics in real time. The DRL agent learns to make decisions regarding the control inputs (i.e., pressure measurements) by reducing the drag produced.
The project's methodology involves creating a realistic and high-fidelity simulation environment using Computational Fluid Dynamics (CFD) techniques. The DRL agent interacts with this environment, continuously learning and adapting its control policies through reinforcement learning. The performance of the DRL-controlled system is evaluated by assessing its ability to achieve aerodynamic objectives.
The outcomes of this research demonstrate the potential of DRL as a powerful tool for optimising complex fluid dynamics problems, contributing to advancements in control theory and aerodynamics.

KEYWORDS: Aerodynamic, Flow Control, Deep Reinforcement Learning, Proximal Policy Optimisation, Deep Q-Network

Contents

Table of Acronyms

Notation	Description
C_D	Drag coefficient.
$\overline{C_D}$	Mean drag coefficient over time.
$\overline{C_{D\tau}}$	Mean drag coefficient in the control time scale.
CFD	Computational Fluid Dynamics.
CSM	Computational Structural Mechanics.
DRL	Deep Reinforcement Learning.
DQN	Deep Q-Network.
MDP	Markov Decision Process.
N-S	Navier-Stokes equations.
PPO	Proximal Policy Optimization.
R	Reward in DRL.
Re	Reynolds number.
RL	Reinforcement Learning.
ΔC_D	Scaled drag coefficient.
Δt	Aerodynamic time scale.
$\Delta \tau$	Control time scale.

List of Figures

List of Tables

Chapter 1

Problem Description and Objectives

This work aims to extend the capabilities of applying reinforcement learning techniques to fluid dynamics. In particular, the domain of the problem is a cylinder immersed in a current with fixed geometric parameters.

A body immersed in a current produces an opposing force to motion known as drag. By opposing motion, drag is a force that is sought to be minimised, except in very special cases such as air brakes or parachutes whose operation is based precisely on the generation of drag.

There are many techniques for reducing drag, the research and implementation of which is a growing and developing area. This multitude of techniques includes design solutions, which therefore include the creation of parts whose shape reduces drag, but also active solutions such as the introduction of fixed or moving surfaces on components immersed in the flow.

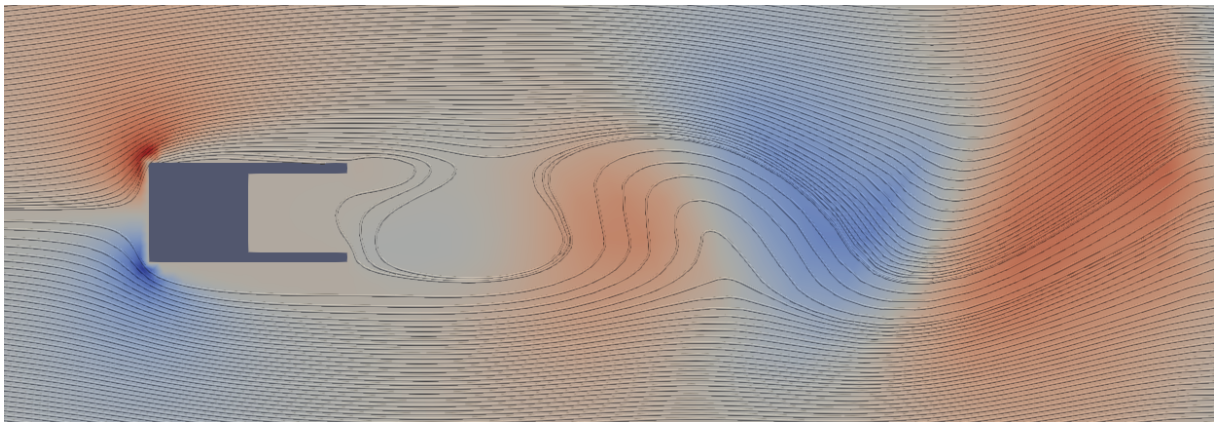


Figure 1.1: Overview of flow structures behind a square with flaps

In this project, on the back of the body are attached moving surfaces as in figure 1.1 whose movement is controlled by a reinforcement learning algorithm. This control will be aimed at optimising the aerodynamic performance of the cylinder, reducing the drag generated by the cylinder. The drag is measured in terms of drag coefficient, computed as follows:

$$C_D = \frac{2D}{\rho U_0^2 A} \tag{1.0.1}$$

where D is the drag computed, ρ is the air density (1.225 kg/m^2), U_0 is the free-stream velocity (in this case 1 m/s) and A is a reference area (in this case 1 m^2).

The fluid-dynamic environment is simulated with an aerodynamic solver to create a realistic simulation framework.

This project therefore has a dual purpose. Firstly, to carry out a study and implement a controller based on artificial intelligence algorithms aimed at aerodynamic control. Secondly, to carry out a feasibility assessment on the proposed case study to evaluate its extension to more complex problems. The aim is to provide a preliminary study, based on an applied practical experiment, to assess the feasibility and efficiency of this type of control.

Since the perimeter of this application remains within the scope of academic research, this work is not aimed at a possible commercial use. On the contrary, it will be aimed at enriching the literature scenario on the topic of the coupling between aerodynamics and artificial intelligence, a still immature area in the process of development.

A more detailed overview of the project objectives and methodology is given in the following sections.

1.1 Objectives

The products of this study can therefore be listed as:

- a greater understanding and analysis of the usefulness of new control logic for improving the aerodynamic performance of a square cylinder;
- the development of a control logic tested using an aerodynamic simulator;
- an assessment of the feasibility and importance that this study could have in extending the field of action of artificial intelligence techniques to the field of fluid dynamics.

1.2 Methodology overview

The planned methodology to complete this project has consisted of several steps as follows:

- Research and design
- Aerodynamic solver discovery
- Systems coupling
- Reinforcement learning development
- Integrated systems training and testing
- Integrated systems evaluation

More details are provided in the following sections.

1.2.1 Research and design

An initial research and development phase has been necessary for an in-depth analysis of the literature and to consolidate the planning of subsequent activities. In this phase, it has been possible to make a more accurate estimate of workload management and, above all, to try to intercept possible risks and problems in advance for prompt resolution.

1.2.2 Aerodynamic solver discovery

This phase was an actual discovery of the aerodynamic solver. The key part has been to evaluate the software and, if it wouldn't fit the requirements, to find another compatible one. Therefore, the objective of this phase was to obtain a stable aerodynamic solver, the functionality of which has been proven by a software test phase. In this case, having assessed the feasibility at an early stage, the occurrence of an incompatibility finding during the ongoing project is avoided.

A first proposal, for this project, was the open-source software turleFSI (`https://github.com/KVSlab/turtleFSI`) [1].

Another possible candidate as an aerodynamic solver was the open-source CFD software OpenFoam (`https://openfoam.org/`).

1.2.3 Systems coupling

Once the aerodynamic representation of the environment is developed and tested, it has been possible to integrate it with the deep reinforcement learning control logic framework to create a single system.

A descriptive diagram of this coupling is shown in figure 3.19 with the control part on the left and the aerodynamic model part on the right.

Operationally, this task has been achieved by employing Gymnasium (cf. `https://pypi.org/project/gymnasium/`), a tool that provides an API to communicate between learning algorithms and environments.

1.2.4 Reinforcement learning development

This is the core phase of the project. It consists of three distinct but interconnected moments: a research and in-depth reinforcement learning phase aimed at identifying one or more possible algorithms useful for control logic followed by a selection phase of algorithms and an evaluation phase.

At this stage, a thorough study of the available literature has been important to know the state of the art and thus identify the most viable possibilities for innovation. The literature on reinforcement learning is rich, providing a wide choice of algorithms.

About flow control, the main algorithms implemented, also in different scenarios, are deep Q-learning (DQN), asynchronous DQN, trust region policy optimisation (TRPO), asynchronous actor-critic methods (A3C) and PPO [2].

The reference library was stable baselines 3 [3].

1.2.5 Integrated systems training and testing

Once the framework is functional, the actual training phase of the deep reinforcement learning algorithm in the scenario coupled with the aerodynamic solver has begun. It is only at this stage that the performance of the system became clear and the feasibility of the required learning algorithm was assessed. The training phase is followed by a testing phase of the resulting system. It is then at this stage that it will be possible to begin an

evaluation, mostly numerical, of the results obtained.

However, it should be noted that the training and testing phases have been part of a larger cyclic process in which several configurations were tried to achieve the best possible design of the control logic.

1.2.6 Integrated systems evaluation

Once the numerical development phase of the solution has been completed, a physical interpretation phase of the results was developed. This allowed a deeper explanation of the results, analysing the aerodynamic motivations behind the performance improvement enabled by the control law.

It is important to note that this phase is temporally at the end of the code testing phase, but at certain times it took place in parallel. Anticipating the aerodynamic understanding helped to more correctly evaluate the results produced, if not in numerical terms, at least in physical terms.

Operationally, this step was carried out by analysing the temporal and spatial trends of the calculated aerodynamic quantities (such as pressure and velocity fields) of the aerodynamic solver.

Chapter 2

Context

Aerodynamics is the study of how air interacts with objects in motion and has long been a cornerstone of aviation and engineering. It forms the scientific foundation for the design and operation of aircraft, wind turbines, and various other applications where understanding and manipulating airflow is essential. Artificial intelligence, with its remarkable ability to process vast amounts of data and uncover complex patterns, stands at the forefront of cutting-edge technology, representing one of the most transformative and impactful advancements nowadays.

The synergy between these two domains can reshape the way to explore and optimise aerodynamic phenomena, leading to groundbreaking advancements like in aviation efficiency [4] and renewable energy production [5]. In particular, the combination of artificial intelligence and fluid dynamics is an area that has only recently gained interest, despite its many possible applications and great potential [2].

Over the past few years, the emergence of deep neural networks (DNN) has equipped reinforcement learning with formidable new tools. The fusion of deep learning and RL, known as deep reinforcement learning (DRL), has successfully overcome significant challenges that previously impeded classical RL methods. DRL has revolutionised the field by enabling the utilisation of high-dimensional state spaces and harnessing the feature extraction prowess of DNNs. Notably, DRL has exhibited remarkable efficiency in various domains, including robotics [6] and language processing [7], achieving unprecedented performance levels, even surpassing human capabilities in numerous games [8].

The most popular and promising developments regarding aerodynamics concern flow control, i.e. the practice of controlling a body (such as a wing or a structure) through direct action on the fluid in which it is immersed, or shape optimisation, i.e. the optimisation of an aerofoil shape to enhance the aerodynamic performances [9].

Recently, aerodynamics started to benefit from advances in the field of machine learning. Of the various techniques, the first to be exploited to this advantage was genetic programming [10]. Several studies have been carried out in this regard, including [11] which employed the linear genetic programming to enhance jet mixing and discover novel wake patterns, and [12] adopted GP-identified control laws to suppress vortex-induced vibrations in a numerical simulation environment.

Often, the primary challenge lies in the substantial computational resources required to assess the sensitivity of the objective function to variations in design parameters through repetitive flow calculations. Robust and versatile search methods, such as evolutionary

or genetic algorithms, possess several appealing characteristics. These approaches are, in fact, particularly well-suited for addressing multi-objective multi-parameter problems and have demonstrated success in practical scenarios, including design shape optimisation within the aerospace [13] and automotive industries [14].

Deep reinforcement learning (DRL) has been adopted in a wide range of physics and engineering domains for its ability to solve decision-making problems that were previously out of reach due to a combination of non-linearity and high dimensionality [15]. In particular, compared with model-based closed-loop control methods, deep reinforcement learning avoids modelling the complex flow system and effectively provides an intelligent end-to-end policy exploration paradigm [16]. A major advantage of DRL is that it can be used as an agnostic tool for control and optimisation tasks, both in continuous and discrete contexts. Its only requirement is a well-defined interface from the environment, which can consist of a numerical simulation or a real-life experiment [2].

Among the several AI technologies, Reinforcement Learning proved to be effective in learning a control strategy by trial-and-error via stochastic agent-environment interactions [15]. Among the others, Proximal Policy Optimisation (PPO) has been used successfully to develop RL controllers for fluid flows [17], and has been chosen in previous studies among other policy-based methods due to its relative simplicity in implementation, better sample efficiency, and ease of hyper-parameter tuning [18]. Another algorithm widely used for its versatility and efficiency in producing high-quality results is deep Q-networks (DQN), which is also used in various studies applied to fluid dynamics [19]. A relevant application of this algorithm was done in [20], where DRL was applied to swimming kinematics finding a strategy that reduces the energy expenditure in a viscous incompressible flow.

Aerodynamic control refers to the deliberate manipulation and management of airflow around an object or within a fluid system to achieve specific desired outcomes. This concept is crucial in various fields, including aerospace engineering, automotive design, wind energy, and even sports equipment.

An important example concerns the control of bluff bodies, characterised by their high drag and separation-prone wakes, which have been a subject of study in aerodynamics for decades. Understanding and controlling the wake behind these bodies is crucial for reducing drag and improving the overall efficiency of various engineering systems, including vehicles, buildings, and wind turbines. The study presented in [21] is based on the deliberate manipulation of the airflow around a rectangle or square 2D cylinder using rear-mounted flaps that are oscillated harmonically, as shown in figure 2.1. This technique is employed to control and enhance aerodynamic performance, reduce drag, and even delay flow separation. Although no active control based on artificial intelligence techniques is included, it explains and demonstrates the effectiveness of introducing flaps at the trailing edge to control aerodynamic performance.

With the advancement of AI and machine learning, aerodynamic control systems are becoming smarter and more adaptive, capable of real-time adjustments for maximum efficiency and safety. On this topic, the study presented [22] has shown that a passive (i.e., without control) flow control device (flap) inspired by self-actuating covert feathers of birds can improve lift. This lift improvement, tested through numerical analysis, is explained by the step in the pressure distribution (or pressure coefficient as shown in

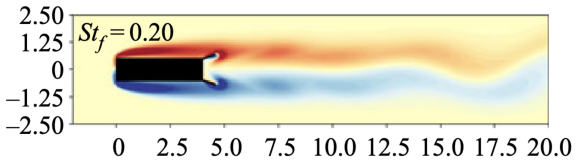


Figure 2.1: Bluff body controlled by two flaps with harmonic motion [21]

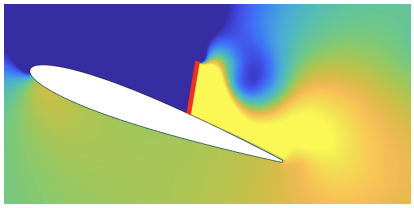


Figure 2.2: Pressure coefficient over bio-inspired flap (in red) [22]

figure 2.2), over the airfoil [23].

The approach proposed by [22] presents a step forward and it was the inspiration for the research carried out in this project. The innovative idea was in fact to connect a DRL-based controller to the aerodynamic problem. A framework was therefore created in which the DRL neural network receives as input the system's observations, i.e. aerodynamic measurements of vorticity and flap deflection. While it produces as output the control signal, i.e. the stiffness of the torsional spring to which the flap is hinged. The system's reward is a function of lift, i.e. the variable that the control system must maximise. The hybrid controller provided lift improvements as high as 136% with respect to the flap-less airfoil. These lift improvements are attributed to large flap oscillations due to stiffness variations mediated by the DRL algorithm. This configuration proved to be extremely effective in improving the aerodynamic performance of the problem at hand, thus being an excellent guideline in solving this project.

The variety of aerodynamic control phenomena and problems and the extent of valid applications for Deep Reinforcement Learning algorithms make this a prolific area of research and study opportunities. Despite recent research attention, it is a field yet to be explored, whose actual practical applications are still being understood. It is in this context that this study is set. It is intended to be a preliminary analysis of how aerodynamics can benefit from artificial intelligence. Without claiming to be an exhaustive study on the subject, the aim will be precisely to support or not support the usefulness of artificial intelligence even in this fascinating yet complex field of aerodynamics.

Chapter 3

Methods

3.1 Aerodynamic simulation

When developing control logic with Deep Reinforcement Learning, the definition of the environment with which the agent interacts is crucial. In the problem developed here, the scenario is a domain containing fluid (air) with which the agent interacts. The simulation of an aerodynamic environment presents several critical issues that require in-depth preliminary study in order to ensure that physically sensible results are obtained.

3.1.1 Aerodynamic problem

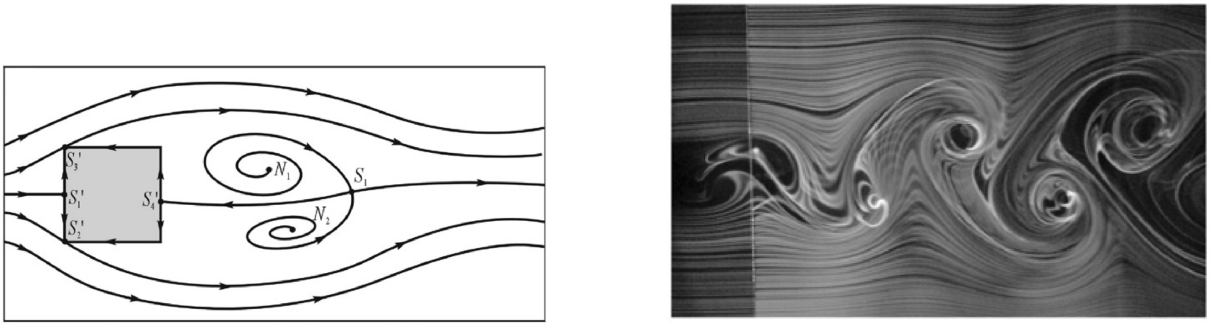
This study focuses on the aerodynamic flow around an object with a bluff shape, in particular a 2D square cylinder. Architectural designs, such as buildings, bridge decks, monuments, and heat exchangers, have frequently utilised square or rectangular cylinders. These shapes tend to induce sudden and fluctuating flow patterns as the fluid interacts with them. Various dynamic occurrences, like detachment of flow, formation of separation bubbles, instability in the shear layer, shedding of vortices, and more, manifest as the fluid flows around these cylinders [24].

Flow around a square cylinder involves the interaction between a fluid (such as air or water) and a solid square-shaped object placed in its path. This phenomenon is commonly studied in fluid dynamics due to its complex and interesting characteristics.

As the fluid encounters the square cylinder, it divides into different regions, each with distinct flow behaviours. On the side facing the oncoming flow, the fluid initially approaches the square cylinder smoothly. However, as it reaches the corners of the square, the flow starts to slow down and experience pressure variations due to the abrupt change in geometry. This often leads to the development of vortices at the corners, where the fluid curls around and forms swirling patterns.

As the fluid continues to move around the square cylinder, it creates a complex interaction between alternating areas of low and high pressure. These pressure differences cause the fluid to separate from the surface of the cylinder, creating a flow separation region. This region is characterised by a re-circulation of the fluid, forming what is known as a separation bubble. The detachment of the flow from the cylinder's surface contributes to the formation of vortices in the wake region, which are periodically shed into the downstream flow.

The periodic shedding of vortices creates a phenomenon called vortex shedding, as



(a) Topological flow patterns (b) Smoke-streak flow patterns

Figure 3.1: Flow characteristics around a square cylinder [24]

seen in figure 3.1. These vortices are like swirling eddies that detach from the corners of the square cylinder and trail downstream in a regular pattern. This shedding leads to the creation of an alternating pattern of low and high pressure zones in the fluid's wake, resulting in a characteristic vortex pattern.

The flow behaviour around a square cylinder is influenced by factors such as the Reynolds number (a dimensionless parameter representing the ratio of inertial forces to viscous forces), which determines whether the flow is laminar or turbulent. At higher Reynolds numbers, the flow becomes more complex, with multiple shedding frequencies and interactions between vortices.

3.1.2 Aerodynamic solver

An aerodynamic solver is a computational tool used in the field of aerospace engineering and fluid dynamics to simulate and analyse the behaviour of air or other gases as they interact with objects, surfaces, or structures. As these solvers play a crucial role in understanding, designing, and optimising various aerodynamic systems, its choice is an important part in the design of the project.

3.1.2.1 Fluid-structure interaction overview

In Newtonian mechanics, here taken over by [25], the fundamental laws of solid and fluid mechanics, considered as continuum, are obtained from the equations of conservation of mass and momentum. In addition, these equations are accompanied by constitutive relations for material characterisation of the solid and fluid.

Considering solid mechanics, the equation for deformation $\mathbf{d}$ in the solid domain S is:

$$\rho_s \frac{\partial \mathbf{d}^2}{\partial t^2} = \nabla \cdot (P) + \rho_s f \quad in\ S \tag{3.1.1}$$

where ρ_s is the solid density, P is the first Piola-Kirchhoff stress tensor and f are the body forces, originated outside of the domain.

Considering fluid mechanics, under the hypothesis of an incompressible fluid, the equations for the velocity field $\mathbf{u}$ and the pressure field p, known as the Navier-Stokes equations,

in the fluid time domain $F(t)$ are:

$$\rho_f \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = \nabla \cdot \sigma_f + \rho_f f \quad in\ F(t) \tag{3.1.2}$$

$$\nabla \cdot \mathbf{u} = 0 \quad in\ F(t) \tag{3.1.3}$$

where ρ_f is the fluid density, f is body force and σ_f is the Cauchy stress tensor, $\sigma_f = \mu_f(\nabla \mathbf{u} + \nabla \mathbf{u}^T) - pI$ with μ_f a constant for the Newtonian fluid and I the identity matrix.

The complexity of the Navier-Stokes equations lies in the fact that there is no analytical solution (except for very specific cases) due to the non-linearity of the convection term to the left-hand side of the equation. For this reason, the only possible technique is numerical resolution using Newton's scheme.

For a fluid-structure interaction problem to be well posed, boundary conditions are finally required at the borders of the solid ∂S and fluid domain ∂F. In this work, a constant velocity profile U_0 is imposed at the front edge of the fluid domain while at the border between solid and fluid, two conditions are imposed: 'non-penetration' whereby the velocity profile perpendicular to the solid is zero (i.e. there is no fluid loss and mass is conserved) and 'non-slip' whereby the velocity profile parallel to the solid is zero (i.e. there is friction between the body and the fluid in which it is immersed).

3.1.2.2 turtleFSI

For the purpose of this project, it has been used the open-source software turleFSI (`https://github.com/KVSlab/turtleFSI`) [1]. This software is based on the concept of fluid-structure interaction (FSI) which couples computational fluid dynamics (CFD) and computational structural mechanics (CSM).

Generally, CFD and CSM are individually well numerically resolved. FSI introduces a layer of complexity tracking of the interface separating the fluid and solid domains. Both mechanics are treated with separate governing equations of the individual fluid and structure problem, together with unique auxiliary kinematic, dynamic, and material relations [26].

turtleFSI is a monolithic fluid-structure interaction solver written in FEniCS, and has out-of-the-box high performance capabilities. The goal of turtleFSI is to provide a simple, but robust solver to investigate fluid structure interaction problems. An example of its output is shown in figure 3.2.

3.1.3 Aerodynamic mesh

The basis on which the aerodynamic equations can be solved is a geometric mesh that discretizes the fluid.

In general, the design of a mesh is a complicated and iterative process. It is indeed necessary to identify a trade-off point between complexity (and thus computational cost) and correctness of the solution obtained.

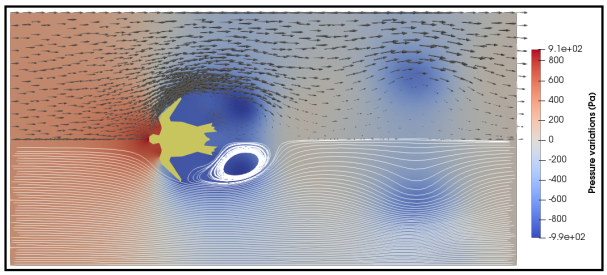


Figure 3.2: turtleFSI demo

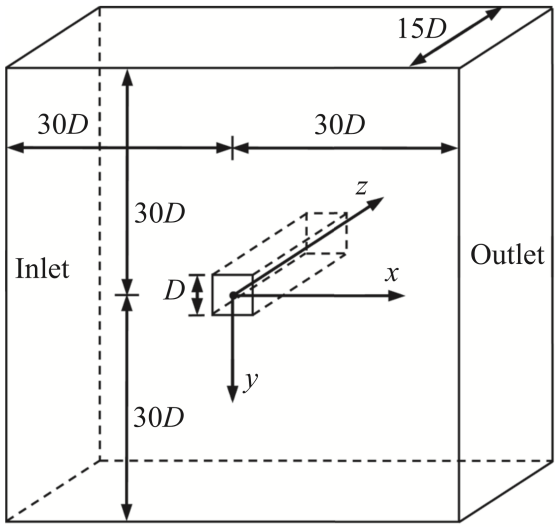


Figure 3.3: Original domain [27]

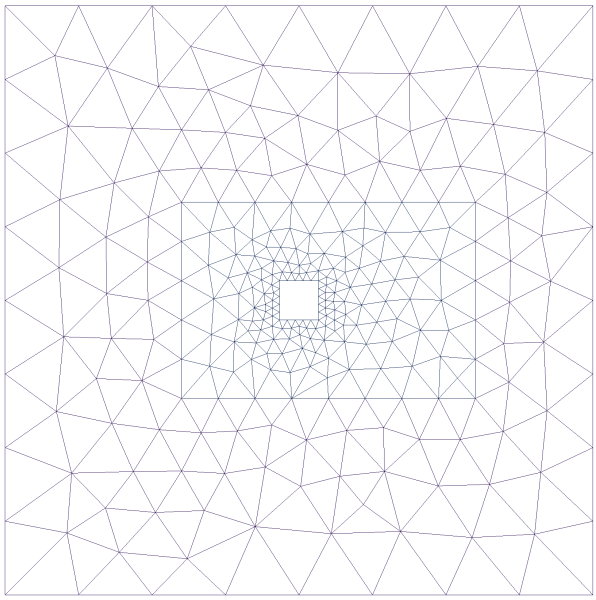


Figure 3.4: Vanilla aerodynamic mesh

In other words, it is important to identify a mesh that is neither too refined (which correctly solves the aerodynamic problem but whose resolution will be long and expensive) nor too coarse (on the contrary, it will be very fast but with incorrect results).

Some key parameters for the computational mesh are specified as follows:

- external dimensions: these concern the dimensions of the fluid domain. It is necessary that aerodynamic phenomena are adequately contained in the domain, without dispersion of aerodynamic quantities. The outermost areas are usually discretized in a coarser way to increase the speed of resolution;
- internal dimensions: in addition to the fixed dimensions of the bodies to be studied, it is often appropriate to study the presence of areas of greater discretization near the aerodynamic phenomena of interest in order to increase the resolution of the solution;
- sizes of the calculation mesh: these dimensions represent the actual spatial discretization of the domain. They must be correctly established by indicating areas of minor or major discretization.

The study presented in [27] was used as a reference for the setup of the geometry to be solved. A comprehensive and detailed study of the two- and three-dimensional hydrodynamic characteristics of a flow around a square cylinder by means of direct numerical simulations is presented here. In particular, variations in hydrodynamic forces and wake

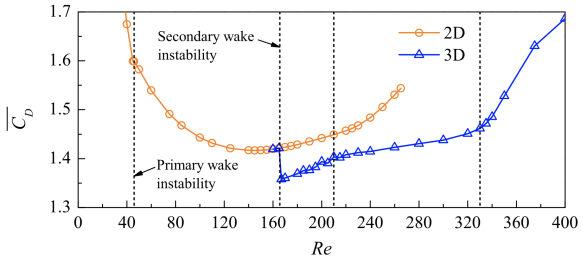


Figure 3.5: Drag coefficient vs Reynolds number [27]

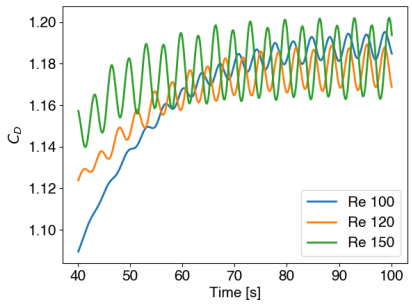


Figure 3.6: C_D vs time for different Reynolds numbers

characteristics are presented as the Reynolds number changes. The basic 3D geometry presented in this work is shown in figure 3.3. The domain employed in this project is a 2D section of the 3D geometry. A preliminary representation of the domain in shown in figure 3.4.

3.1.4 Reynolds number

The Reynolds number is a parameter of fundamental importance in the study and simulation of fluids. It is a dimensionless measure of the relationship between inertial and viscous forces. It is therefore important for characterising fluid dynamics: at low Reynolds numbers, flows tend to have a laminar characteristic, while at high Reynolds numbers, flows tend to be turbulent.

It is calculated as:

$$Re = \frac{\rho U L}{\mu} \tag{3.1.4}$$

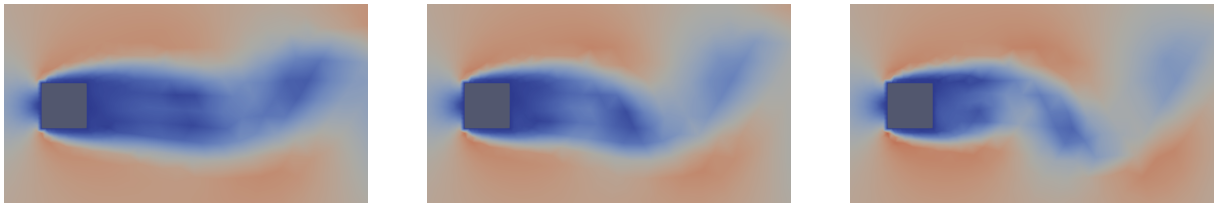
where ρ is the fluid density, u is the flow speed, L is a characteristic length (in this case is set to 1 as the square has unit dimension) and μ is the fluid dynamic viscosity.

In particular, figure 3.5 taken from [27] shows the drag coefficient trend for the simulation around a 3D cylinder (see figure 3.3) as the Reynolds number changes, distinguishing a 2D and a 3D region. This distinction describes the flow behaviour, with a threshold value of approximately 160. Below this value, despite the 3D geometry, the flow tends to behave two-dimensionally, above this value, the dynamics restore the three-dimensionality of the flow.

The simulations that will be carried out in this project will be in two dimensions. To be sure, therefore, that no information about the fluid dynamics is being lost, it will be necessary to maintain a Reynolds number of less than 160.

Three Reynolds number values were tested to assess their effect on the aerodynamic characteristics of the obtained solution. In particular, values 100, 120 and 150 were tested, all below the previously identified threshold value of 160.

The figure 3.6 shows the drag coefficient trend over a simulation time window of 40 to 100 seconds. A transition phase of the coefficient can still be seen for the first ten to twenty seconds until a stationary phase is reached.



(a) Re=100 (b) Re=120 (c) Re=150

Figure 3.7: Velocity field at t = 40 s

For the three cases tested, the average coefficient values are very similar, as shown in table 3.1. The biggest difference lies in the turbulent dynamics associated with the formation of the wake vortexes. In particular, a higher Reynolds implies a greater turbulence contribution and therefore an earlier engagement of vortex shedding. This can be seen in figure 3.7 which shows the velocity field for the 3 proposed cases, at 40 seconds from the start of the simulation.

Re	$\overline{C_D}$
100	1.167
120	1.166
150	1.176

Table 3.1: $\overline{C_D}$ vs Reynolds number

Indeed, it can be seen that an increase in Reynolds speeds up the attainment of a stable condition. For this reason, considering that many simulations will be necessary in the control design phase, it was decided to use a Reynolds number of 150, which guarantees correct aerodynamic behaviour and achieves the desired wake effect in less time.

3.1.5 Mesh convergence analysis

The mesh convergence analysis phase is necessary and fundamental in order to obtain a mesh that is suitable for the required use.

For this phase, basically 2 degrees of freedom have been chosen, referring to the dimension of the domain and the mesh size (or discretization).

The two parameters employed are:

- δ_L: parameter used to scale the domain dimensions,
- δ_D: parameter used to scale the mesh size.

Figure 3.8 shows the domain. The domain is characterised by a square of unit side (red square in the figure) inside the centre of a square domain of side $30 \cdot \delta_L$ (called the outer domain, blue square in the figure). There is also a more detailed inner domain to allow for more accurate resolution in the vicinity of the square. This is represented by the green rectangle in the figure, placed 3 units from the square, has a height of 5, is vertically centred and has a length of $15 \cdot \delta_L$.

Thus, the size parameter controls the side of the outer domain and the length of the inner domain.

With regard to mesh refinement, three different measures were used:

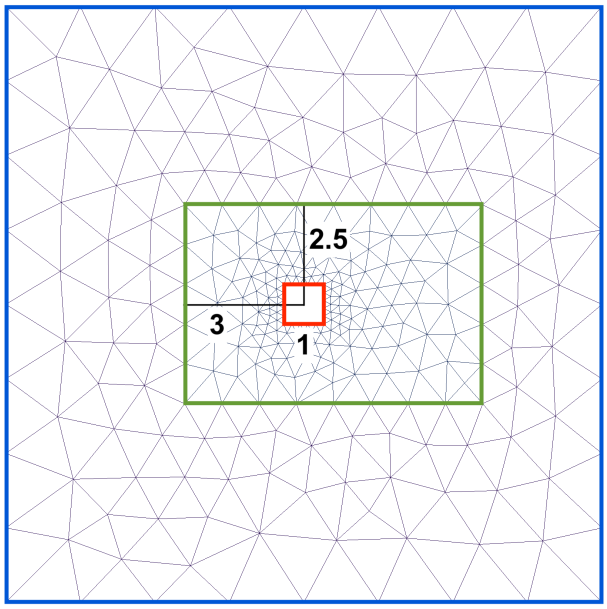


Figure 3.8: Mesh parameters

- square size: $0.1 \cdot \delta_D$, is the most refined measure around the unit square
- inner domain size: $0.5 \cdot \delta_D$, is the size on the green rectangle, which allows for greater definition around the unit square
- outer domain size: $1 \cdot \delta_D$, larger size on the outer domain to reduce complexity in a region poorly affected by the unit square

In order to test different configurations, three values were used for the two parameters respectively, as shown below:

δ_D	**Square size**	**Inner domain size**	**Outer domain size**
2	0.2	1	2
1	0.1	0.5	1
0.5	0.05	0.25	0.5

Table 3.2: Mesh size parameters

δ_L	**Inner domain length**	**Outer domain length**
1	15	30
0.75	11.25	22.5
0.5	7.5	15

Table 3.3: Mesh dimension parameters

The combinations of these parameters thus yield 9 test cases, as shown in the table 3.4 in relation to δ_D and δ_L.

The convergence analysis is then carried out by testing the 9 proposed cases by evaluating:

Test cases		**Grid dimension**		
		Coarse	***Medium***	***Fine***
Domain dimension	***Big***	$\delta_L = 1$, $\delta_D = 2$	$\delta_L = 1$, $\delta_D = 1$	$\delta_L = 1$, $\delta_D = 0.5$
	Medium	$\delta_L = 0.75$, $\delta_D = 2$	$\delta_L = 0.75$, $\delta_D = 1$	$\delta_L = 0.75$, $\delta_D = 0.5$
	Small	$\delta_L = 0.5$, $\delta_D = 2$	$\delta_L = 0.5$, $\delta_D = 1$	$\delta_L = 0.5$, $\delta_D = 0.5$

Table 3.4: Mesh convergence test cases

- the average value of the drag coefficient $\overline{C_D}$: the case in which the coefficient is constant as the complexity of the solving mesh increases (i.e. larger domain and/or more refined mesh) will then be sought. The condition found will then be the one that guarantees the same quality of result while choosing a less complex mesh (and therefore quicker to solve);
- frequency of the drag coefficient: a visual evaluation of its trend in order to assess the similarity of frequencies in the solutions obtained;
- the computational time required: final choice parameter, the case with the shortest time for the same correctness of the solution is preferred.

Table 3.5 shows the average value of the coefficient for the different test cases while Table 3.6 shows the computational time expressed as a multiple of the computational time required for the fastest case (small domain and coarse grid whose value is precisely 1.0).

The colour scale represents, from red to green, the worst and best result obtained, considering that the solution obtained with the larger and finer grid gives the best representation of the aerodynamic phenomenon. The best results in terms of drag coefficient are obtained for a medium or fine grid, and in terms of time for a coarse or medium grid. The case of domain small is excluded as it presents the worst results for each grid size.

Since the drag coefficient is of greater importance, we have focused on the four best cases (colour greenish in table 3.5): medium or fine grid for a medium or large size.

$\overline{C_D}$		**Grid dimension**		
		Coarse	***Medium***	***Fine***
Domain dimension	***Big***	1.14	1.18	1.17
	Medium	1.15	1.19	1.18
	Small	1.22	1.24	1.23

Table 3.5: Mesh convergence $\overline{C_D}$

Time scale		**Grid dimension**		
		Coarse	***Medium***	***Fine***
Domain dimension	***Big***	2.2	12.1	104.3
	Medium	1.5	8.1	62.8
	Small	1.0	5.2	36.6

Table 3.6: Mesh convergence time scales

Figure 3.9 shows the visual comparison of drag coefficient trends between selected cases in a simulation time window. It can be seen how the trends, in terms of mean value

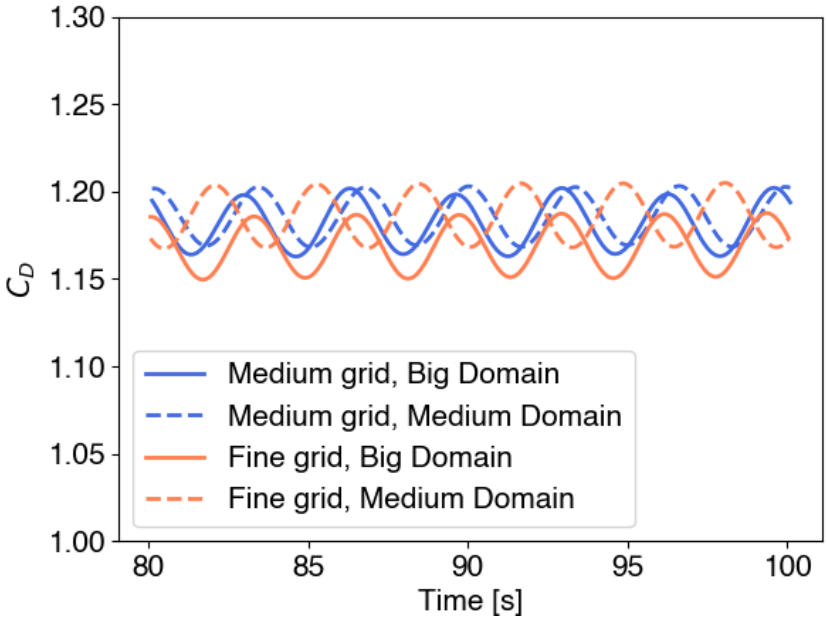


Figure 3.9: Final mesh convergence comparison

and frequency, are very similar. This comparison can therefore guarantee that the quality of the results is very similar, thus opting for the case that is quickest to solve.

Thus, the case with the medium grid and medium domain was selected as the final aerodynamic mesh. Figure 3.10 shows the final mesh, providing an outline view (a) and a zoom on the inner domain and square (b). The domain is thus found to have an outer dimension of 22.5 with a grid of 1, the inner domain rectangle has a length of 11.25 with a grid of 0.5 on the perimeter, and the square of unit side with a grid of 0.1.

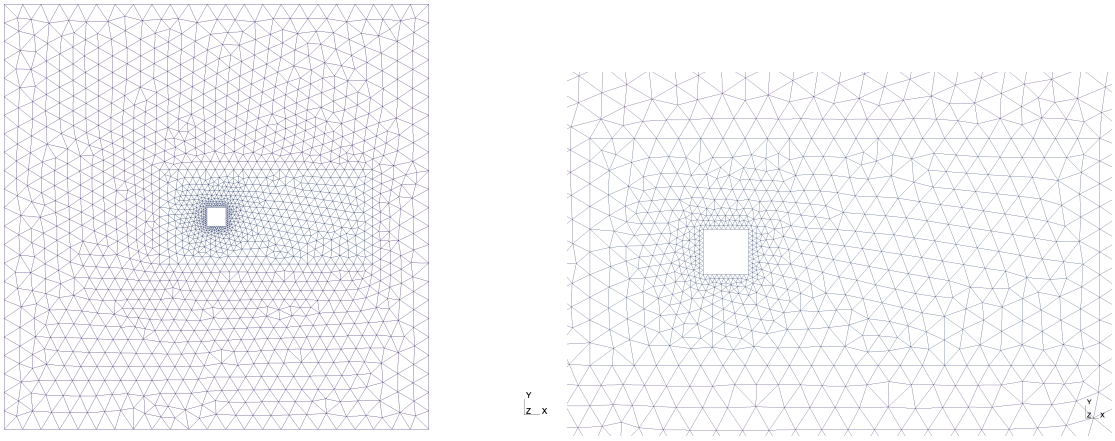
3.1.5.1 Numerical stability

In the proposed numerical resolution, time discretisation is set as a parameter by providing a time interval Δt. This interval, in order to guarantee the numerical stability of the solution, must be bound by two conditions (please refer to the relevant literature for details):

- CFL condition: for which $\Delta t \leq \frac{\Delta x}{U}$
- Von Neuman condition: for which $\Delta t \leq \frac{\Delta x^2}{2\mu}$

where Δx is the finer space discretisation (in this case the square grid dimension), U is the speed profile (in this case set to 1 as the inlet profile U_0 is unitary) and μ is computed from equation 3.1.4 given the Reynolds number set to 150.

The two conditions therefore impose a delta t as the grid size varies, i.e. as the parameter δ_D varies. It therefore turns out that: when $\delta_{D,square} = 0.2$ then $\Delta t = 0.2$, when $\delta_{D,square} = 0.1$ then $\Delta t = 0.1$ and when $\delta_{D,square} = 0.05$ then $\Delta t = 0.05$.



(a) (b)

Figure 3.10: Final aerodynamic mesh

3.2 Aerodynamic Control

This section will present the main fluid control logics available, focusing on the introduction of flaps to the mesh obtained in the discovery phase of the aerodynamic solver. In this way, it will be possible to evaluate its effects in order to choose the final design of the problem prior to the introduction of active control via deep reinforcement learning.

3.2.1 Basic principles

Flow control refers to a multitude of techniques aimed at manipulating a fluid in order to control its behaviour and the main associated phenomena. In aerodynamics and fluid dynamics, there are several types of control, the main ones of which are listed here:

- passive flow control: it involves using fixed or static devices to modify the flow of a fluid;
- active flow control: it involves the use of external energy sources to manipulate the flow. These methods often use actuators, sensors, and control systems to dynamically adjust the flow patterns;
- fluid injection or suction: it involves introducing or removing fluid mass from the flow to enhance mixing, affect the boundary layer, delay separation or reduce drag;
- boundary layer control: it involves managing the thin layer of fluid near a surface;
- morphing surfaces: it involves changing surfaces shape in response to external factors like pressure or flow conditions.

In the scenario where artificial intelligence logics are integrated with flow control, applications are particularly referred to active control, or aerodynamic design. In the context of the present work, control through the introduction of flaps, i.e. rigid or deformable structures inserted into the flow and connected to the body on which or from which active control of flow dynamics can be exercised, was used.

In particular, these structures will be aimed at reducing the drag generated by the square cylinder immersed in the flow.

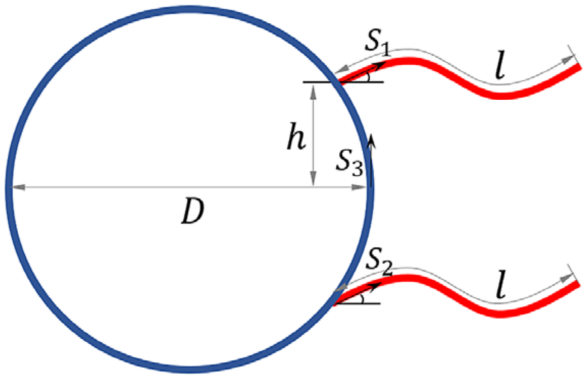


Figure 3.11: Schematic of a cylinder with a pair of flapping flexible filaments [29]

3.2.1.1 Drag reduction with flaps

In literature, studies are available aimed at evaluating drag reduction through passive engagement of deformable flaps (i.e. without control logic), understanding the underlying principles of fluid dynamics and biomimicry [28].

This technique involves attaching flexible filaments or appendages to the rear of a solid object, such as a cylinder, and oscillating them in a flapping motion in order to reduce the drag. Figure 3.11 shows the geometry of this configuration.

The mechanism behind this phenomenon is based on the interactions between the flexible filaments and the flow of the surrounding fluid. When the filaments are oscillated or flapped, they create complex vortex patterns in the fluid flow. These vortices can help to delay the separation of the boundary layer (the thin layer of fluid adjacent to the object's surface) and reduce the overall drag experienced by the object [29].

The oscillating filaments help in controlling the way the fluid flows around the object. By strategically generating vortices and altering the flow patterns, the technique can potentially reduce the drag and increase the object's overall efficiency and speed through the fluid.

It's worth noting that the effectiveness of drag reduction by flapping filaments can depend on various factors including the size, shape, and flexibility of the filaments, the frequency and amplitude of the flapping motion, and the specific characteristics of the fluid flow. As with many fluid dynamics phenomena, the interactions can be complex and might require detailed computational simulations and experimental studies to optimize the design for a given application.

3.2.2 Flaps introduction

The result of the previous section was the definition of an aerodynamic mesh whose geometric characteristics would guarantee a correct simulation of fluid-dynamic phenomena. The next phase was aimed at introducing and evaluating control in the problem at hand. To do this, two flaps were introduced on the back of the square, in a fixed position with dimensions to be set.

In particular, 3 important parameters characterise the introduced flaps:

- L_C: flap length;
- w_C: flap width;
- ρ_C: density of the constituent material, this will be the parameter used for active flow control (see section 3.3 for more details).

Even for flaps, it was necessary to introduce spatial dichretisation since their dynamics and interaction with the flow is at the core of the problem. In this case, the equations of the solid within the flap mesh elements will be solved, and the edge with the exterior will be treated as the interface between solid and fluid. Figure 3.12 shows the structure of the two blue flaps and the mesh which discretizes them.

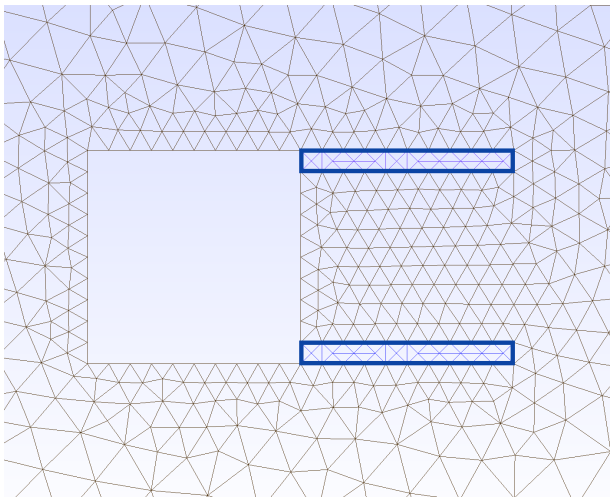


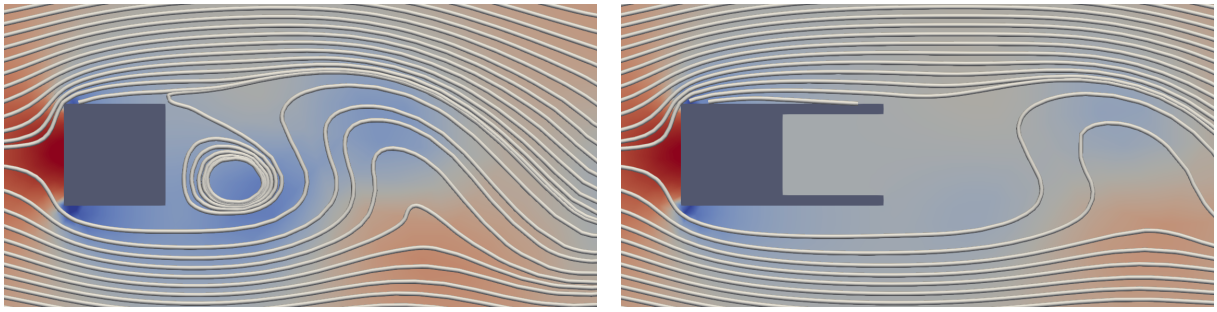
Figure 3.12: Flaps structure and mesh

As a preliminary analysis to the addition of the flaps, it is necessary to carry out a complete simulation of the new mesh and a comparison of the calculation and aerodynamic performance with the simple cylinder case without flaps. This analysis is aimed at understanding the effects of these new structures in the flow and to assess the correct mesh discretization.

For this first simulation, the focus will be on the aerodynamic plane, thus leaving aside the dynamics of deformable flaps and its interaction with the flow. As far as the geometrical parameters are concerned, flaps with unit length ($L_C = 1$) and width 0.1 ($w_C = 0.1$) are considered, while for the non-deformable flap is considered a density $\rho_C \to \infty$. As usual, a constant, unity initial speed profile is considered ($U_0 = 1$) for a simulation time of 100 seconds.

Figure 3.13 shows the flow characteristics for the case without flaps (a) and with flaps (b). In particular, the streamlines of the velocity profile are shown with the pressure field in the background.

In the scenario of a bluff body in a fluid, such as a square cylinder, the main component of the drag produced is called pressure drag (as opposed to the more relevant drag drag in aerodynamic tapered bodies). Pressure drag is, simplifying, due to the large difference in pressure generated between the two faces of the body. In fact, in this case, the front face of the cylinder is subject to much greater pressure than the depressions generated by



(a) No flaps, $\overline{C_D} = 1.186$ (b) Rigid flaps, $\overline{C_D} = 0.987$

Figure 3.13: Flow characteristics without flaps and with rigid flaps

the fluid dynamics on the back face. The pressure delta then generates the force opposed to motion, which is called drag.

In the very first analysis then, a reduction in the pressure difference therefore leads to a reduction in drag and its coefficient.

As shown in figure 3.13, the introduction of flaps alleviates to the distortion of flow lines, thus reducing the depressions caused by wake vortices. Indeed, this leads to a reduction in pressure differences and drag. As a result, the introduction of flaps decreases the coefficient of drag from 1.186 in the case without flaps to 0.987.

This first result is of great importance as it justifies that the use of flaps placed behind a square cylinder reduces the drag generated. Otherwise, stronger control techniques would have been required in order to compensate for any added drag caused by the flaps. In this case, however, the presence of the flaps only has a beneficial contribution. The subsequent introduction of active control will be aimed at further improving performance.

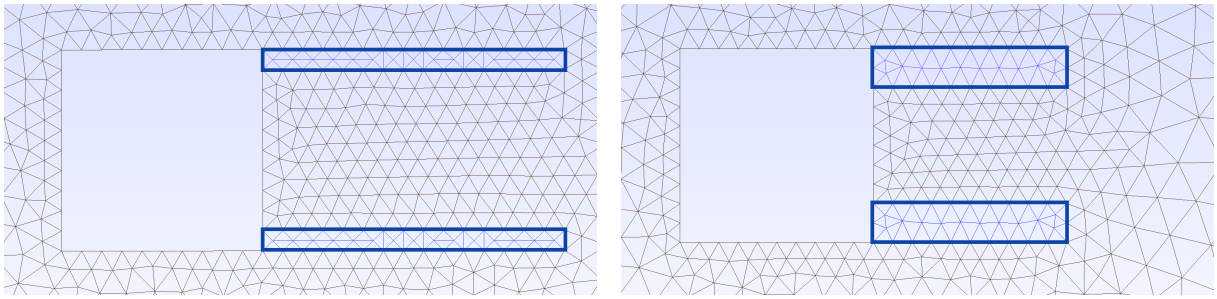
3.2.3 Flaps sensitivity analysis

The next analysis phase involved a sensitivity analysis on the geometric parameters that characterise the flaps. The purpose of this phase was to evaluate the possible contribution of these parameters to the objective of improving aerodynamic performance. The non-secondary factor of computational time will also be taken into account where it depends on the geometric parameters.

As already mentioned, flaps are defined through a length (L_C) and a thickness (w_C). The basic values used are 1 for the length and 0.1 for the thickness. The analysis was then carried out by choosing three length values (1, 1.2 and 1.5) and two width values (0.1 and 0.2), thus forming six case studies as combinations of these parameters. For this analysis, the density value is kept fixed at 2e5.

Figure 3.14 shows an example of the geometry and mesh of a case with long flaps (a) or thick flaps (b).

Simulations are carried out as usual for a time of 100 seconds, considering a uniform unit speed profile. For the various cases, the drag coefficient results are shown in table 3.7 while the time scales (indicated as the ratio of the measured time to the time required for the base case) are shown in table 3.8. Figure 3.15 shows the modulus of the velocity field and the calculation mesh in which it is possible to see behind the square the flaps and their deformation (considering, however, that a scale factor was used to increase the



(a) Long flaps, $L_C = 1.5$, $w_C = 0.1$ (b) Thick flaps, $w_C = 0.2$, $L_C = 1$

Figure 3.14: Flaps geometry examples

deformation and make it more visible).

$\overline{C_D}$		**Flaps width**	
		0.1	***0.2***
Flaps length	***1***	0.905	0.836
	1.2	0.892	0.825
	1.5	0.872	0.812

Table 3.7: $\overline{C_D}$ for flaps sensitivity analysis

Time scales		**Flaps width**	
		0.1	***0.2***
Flaps length	***1***	1	0.92
	1.2	1.12	0.94
	1.5	1.17	0.96

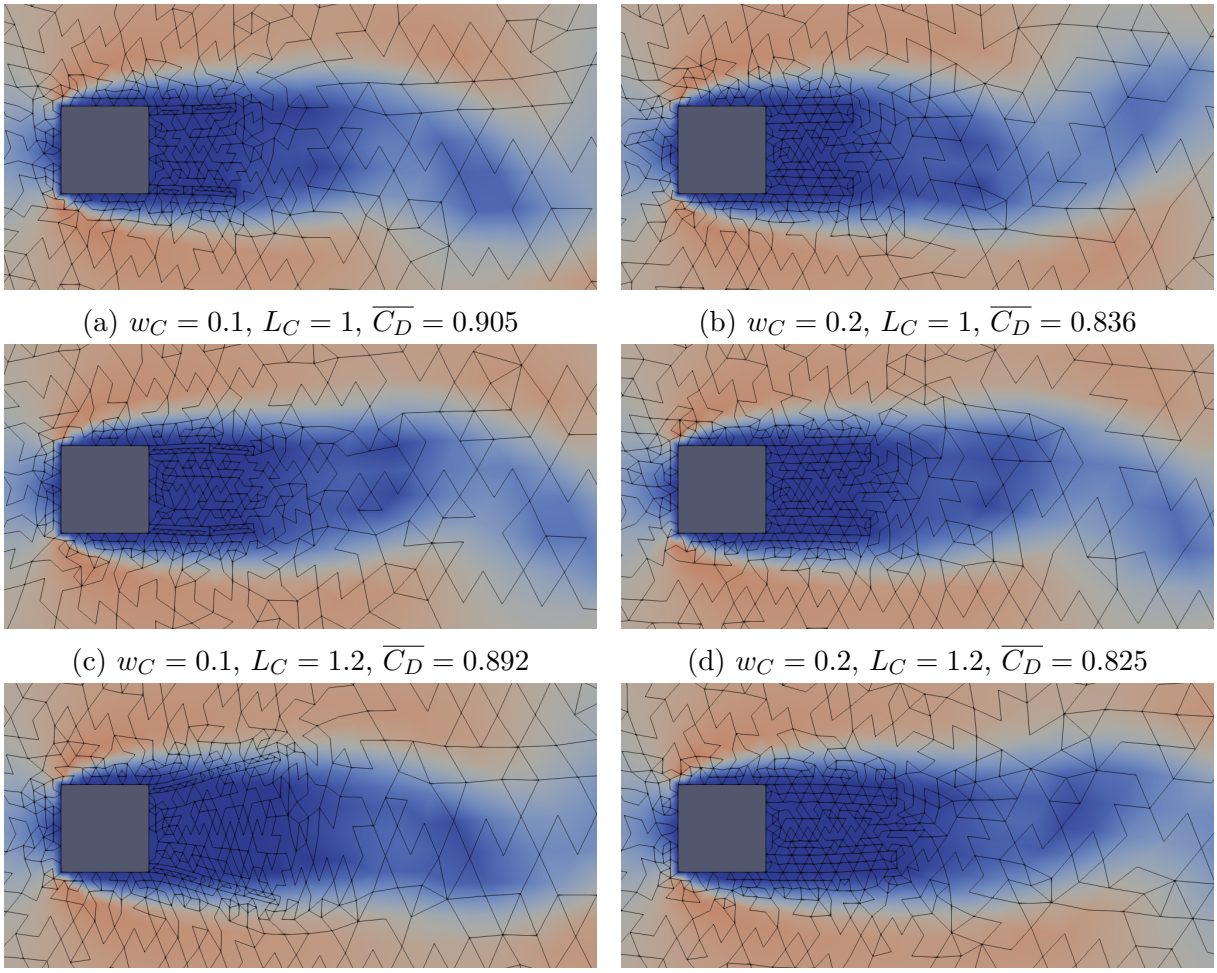
Table 3.8: Time scales for flaps sensitivity analysis

3.2.3.1 Flap length

Increasing the length of the flaps, while keeping the thickness constant, has as its first result a reduction in the drag coefficient. This result is valid for both thickness values tested. This behaviour may be due to the downstream shift of the vortex formation point. This artificial elongation of the body therefore makes it less squat and more aerodynamic, enabling better drag performance. Looking at the velocity profile in Figure 3.15, however, it can be seen that the velocity field shows similar characteristics, so there is no substantial change in aerodynamic resolution.

As far as the computational time is concerned, as shown in table 3.8, as the length increases, the time required for resolution also increases. This may be due to the fact that a greater flap extension results in an increase in the mesh of the solid, but above all in the interference zone between solid and fluid.

The final consideration is therefore that a longer flap improves aerodynamic performance (by reducing the drag coefficient) but worsens computational performance (by increasing the calculation time).



(a) $w_C = 0.1$, $L_C = 1$, $\overline{C_D} = 0.905$

(b) $w_C = 0.2$, $L_C = 1$, $\overline{C_D} = 0.836$

(c) $w_C = 0.1$, $L_C = 1.2$, $\overline{C_D} = 0.892$

(d) $w_C = 0.2$, $L_C = 1.2$, $\overline{C_D} = 0.825$

(e) $w_C = 0.1$, $L_C = 1.5$, $\overline{C_D} = 0.872$

(f) $w_C = 0.2$, $L_C = 1.5$, $\overline{C_D} = 0.812$

Figure 3.15: Velocity field and mesh deformation for different flaps configurations

3.2.3.2 Flap width

For the same length, an increase in thickness also has the beneficial effect of reducing the drag coefficient. In contrast to the previous case, thicker flaps also lead to a reduction in calculation time.

It is not easy to explain the reason for this behaviour. It is possible, however, that a thicker flap reduces the fluid area downstream of the square, thus reducing the pressure gradient between the two faces and thus positively affecting drag.

Therefore, an efficient strategy for decreasing drag could in fact be the use of thicker flaps, also benefiting computational complexity.

3.2.3.3 Final comparison

Having completed the sensitivity analysis, a design solution must be opted for on the basis of the evidence produced.

From the results presented, the best strategy would be to use a long, thick flap to improve aerodynamic performance. In this case, the drag coefficient is reduced by approximately 10% compared to the base case.

However, a longer flap could lead to problematic behaviour related to a possible extra-deformation of the flaps and a consequent clash between them. In addition, an excessive reduction in the value of the drag coefficient through the imposition of geometric parameters could reduce the range of improvement produced by the control, considering that the coefficient cannot be reduced to zero in any way. Furthermore, the gain in terms of time, for a long and thick flap, is not significant in comparison with a basic configuration with unit length and 0.1 width.

The final consideration is that there is no compelling evidence on the choice of one set of parameters over another. However, given the reasons presented here, it was decided to continue with the basic configuration.

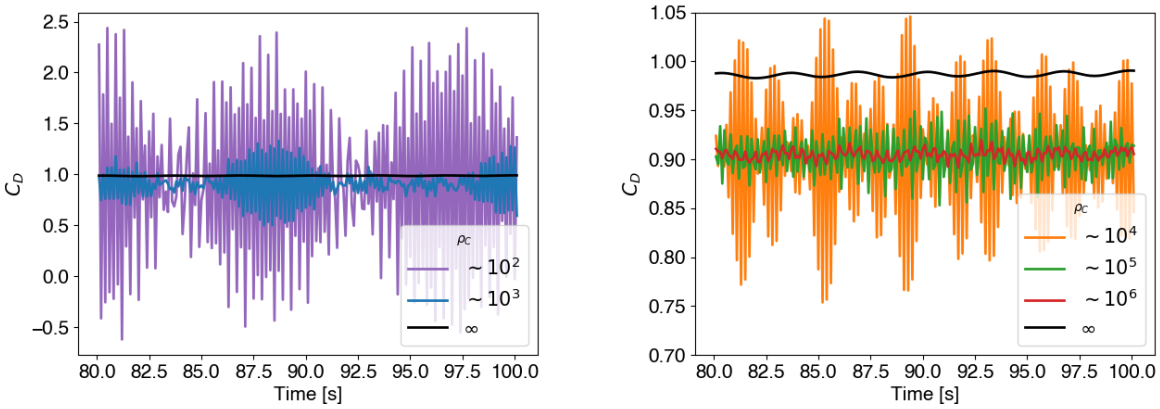
3.2.4 Control sensitivity analysis

Following the analysis aimed at understanding the main geometric parameters, we focused on the actual control parameter of the system, namely the constitutive density of the flaps.

This was therefore an exploratory phase in order to test the actual controllability of the system via this parameter. Furthermore, it is of great importance to understand how this parameter is to be varied and in what range. Different values were then tested with different orders of magnitude starting from 2 up to $2{\cdot}10^6$.

In this case, the simulations were carried out for a time of 100 seconds, setting a test value of the density ρ_C for the entire simulation. The standard values of $L_C = 1$ and $w_C = 0.1$ were used for the flap dimensions.

The drag coefficient produced with the configuration under consideration was then measured and evaluated. Figure 3.16 shows the value of the measured coefficient for a density range between 10^2 and 10^3 in (a) and 10^4, 10^5 and 10^6 in (b). The black line represents the measured value for the solid flap configuration, i.e. with infinite theoretical



(a) Range $[10^2, 10^3]$ (b) Range $[10^4, 10^5, 10^6]$

Figure 3.16: $\overline{C_D}$ for different ρ_C

density and no associated deformation.

For very small density values (such as 2 or 20), resolution presents major problems. In fact, the deformations induced are such that they induce divergence of the solver. For this reason, they will not be taken into account.

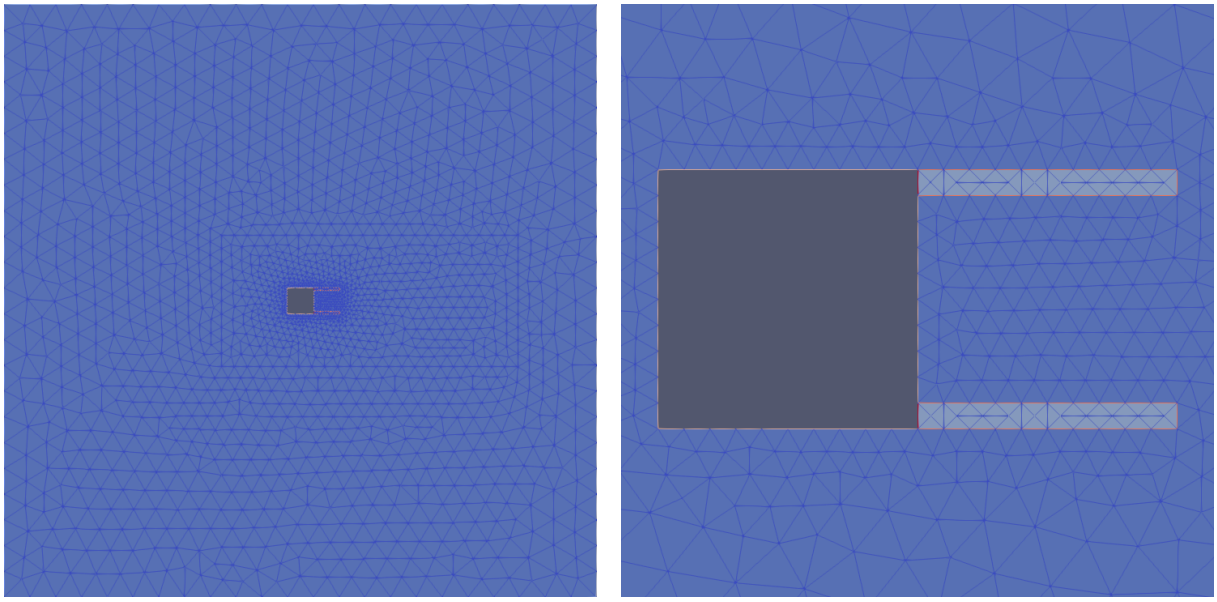
As can be seen in figure 3.16, the drag coefficient signal shows significant oscillations, which are damped as the density value increases. It will also be necessary to neglect the value of 10^2, purple line in (a), as it presents too pronounced oscillations in the signal, pushing the coefficient even to negative values, a symptom of incorrect aerodynamic resolution.

With regard to the signal of the coefficients associated with the other density values, the results are promising. Indeed, it is already important to note that a flexible flap reduces the coefficient compared to the solid case. This is another preliminary justification that the control strategy can lead to good results. Furthermore, the presence of such pronounced oscillations, in a more realistic case in which the stress and fracture dynamics of the material are taken into account, could present criticalities. The objective of the control logic will also be to dampen these oscillations while trying to reduce the drag coefficient.

3.2.5 Final configuration

The analysis carried out in this chapter therefore led to the identification of a correct mesh for the control as well, in order to discretize the flaps and resolve the deformation dynamics. The geometric parameters identified were dictated by computational efficiency and optimisation threshold requirements. The final configuration has a flap of unit length with a thickness of 0.1, shown in figure 3.17. Notwithstanding this, other configurations will be tested later for a more accurate control of the prediction. The effective controllability of the system was also investigated, acting on the flap density value. Thus, the range between 10^3 and 10^6 in which to vary the density was identified.

At this point, the aerodynamic system is functional and ready to be integrated with the control logic.



(a) Outer view (b) Inner view

Figure 3.17: Final aerodynamic mesh with control flaps

3.3 DRL and Flow Control

In this section an insight into the core of the project is given, introducing the main concepts of Deep Reinforcement Learning and how it is integrated with the aerodynamic environment developed so far. Moreover, a description of the training routine will be given.

3.3.1 Basic principles

Deep Reinforcement Learning (DRL) is a sub-field of machine learning that combines reinforcement learning (RL) techniques with deep neural networks to enable agents to learn and make decisions in complex environments. In RL, an agent interacts with an environment to learn a policy that maximises a cumulative reward signal over time. DRL extends this by utilising deep neural networks to approximate the agent's policy or value function, allowing it to handle high-dimensional state spaces and complex decision-making tasks.

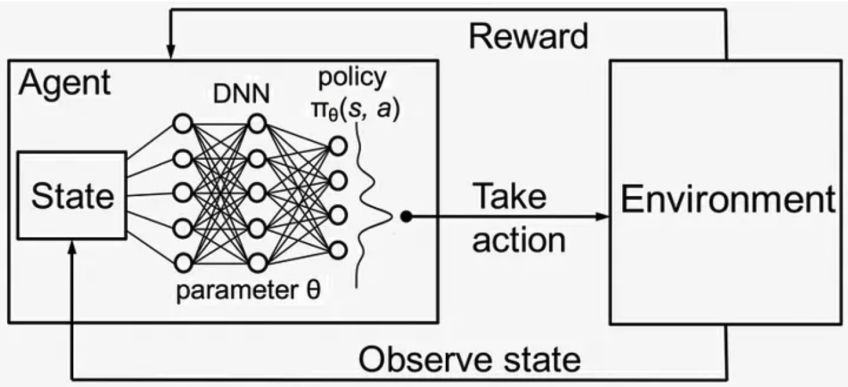


Figure 3.18: DRL overview [30]

A DRL algorithm is typically composed by the following components, also schematised in figure 3.18:

- agent: the learner or decision-maker that interacts with the environment;
- environment: the external system with which the agent interacts and learns from;
- state: a representation of the current situation or configuration of the environment;
- action: the decision or choice made by the agent to influence the environment;
- reward: a scalar value provided by the environment to the agent after taking an action, indicating the immediate desirability of the action;
- policy: a strategy or mapping from states to actions that the agent learns to maximise its cumulative reward;
- value function: An estimate of the expected cumulative reward from a given state or state-action pair.

Reinforcement learning algorithms are based on Markov formulation known as Markov Decision Process (MDP) [31].

3.3.1.1 Markov decision process

A Markov Decision Process is a mathematical framework that represents the RL problem. It consists of a tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathbb{P}$, $\mathcal{R}$) where:

- $\mathcal{S}$ is the set of states in the environment;
- $\mathcal{A}$ is the set of actions that the agent can take;
- $\mathbb{P}$ is the $\mathbb{P}(s'|s,a)$ probability of getting to state s from state s following action a;
- $\mathcal{R}$ is the reward function giving the immediate reward received when transitioning from state s to state s' by taking action a.

In the formalism of MDP, the goal is to find a policy $\pi(s)$ that maximises the expected reward. However, in the context of RL, $\mathbb{P}$ and $\mathcal{R}$ are unknown to the agent, which is expected to come up with an efficient decisional process by interacting with the environment. The way the agent induces this decisional process classifies it either in value-based or policy-based methods [2].

Value-based methods attempt to find the expected cumulative reward an agent can achieve starting from a certain state and following a specific policy. The primary objective in value-based methods is to find the optimal value function, which represents the maximum expected cumulative reward achievable from each state (or state-action pair) [32].

One of the main value-based methods in use is called Q-learning, as it relies on the learning of the Q-function to find an optimal policy. The Q-function is stored in a Q-table, which is a simple array representing the estimated value of the optimal Q-function $Q^*(s,a)$ for each pair (s,a) [2]. The Deep Learning version is called DQN.

Policy-based methods, on the other hand, focus on directly learning the optimal policy. Instead of estimating the value of states, they aim to find the best mapping from states

to actions. Policy-based methods parameterise the policy and iteratively update these parameters to improve policy performance.

The most famous method in this class is Proximal Policy Optimisation (PPO) which optimises policies by maximising expected cumulative rewards through gradient ascent.

3.3.1.2 PPO

Proximal Policy Optimization (PPO) is a popular reinforcement learning algorithm designed for training policy-based models, such as neural networks, in complex environments. PPO focuses on optimizing policies in a stable and efficient manner, by iteratively updating the policy using collected experience data. The key idea of PPO is to ensure that the policy updates are not too drastic, to prevent destabilizing learning [18].

PPO employs a surrogate objective function that encourages policy updates while keeping the change bounded. This prevents the policy from diverging too far from the previous policy, which can lead to instability. PPO has multiple variants, such as PPO-Clip and PPO-Penalty, each with slightly different approaches to enforcing the update bounds.

3.3.1.3 DQN

Deep Q-Network (DQN) is a foundational algorithm in DRL that specifically focuses on value-based reinforcement learning. DQN is designed to learn the optimal action-value function (Q-function) by using a deep neural network to approximate the Q-values for different state-action pairs [33].

DQN employs an experience replay buffer to store and sample from past experiences, which reduces the correlation between consecutive updates and improves learning stability. Additionally, DQN uses a target network to stabilise the learning process further. The target network's parameters are updated less frequently than the primary network's parameters, which helps prevent harmful value estimation feedback loops.

DQN introduces a crucial concept called Bellman equation to iteratively update the Q-values, and it uses an epsilon-greedy exploration strategy to balance between exploiting the current knowledge and exploring new actions.

3.3.2 Systems integration

The basic concepts presented in the previous section on DRL are now made explicit for the problem under analysis.

The interface between the aerodynamic system and the control system was carried out via the Gymnasium library [34]. This allows the creation of an interface that conceals a customised environment such as the one developed in this project.

A representative diagram of the interface between the two systems is shown in figure 3.19. On the right is shown the aerodynamic system, represented by the aerodynamic solver for the cylinder with flaps. On the left is the control system, which is represented as a neural network.

The interaction between the two systems is mediated through three components: action, state and reward. Details are provided in the following sections.

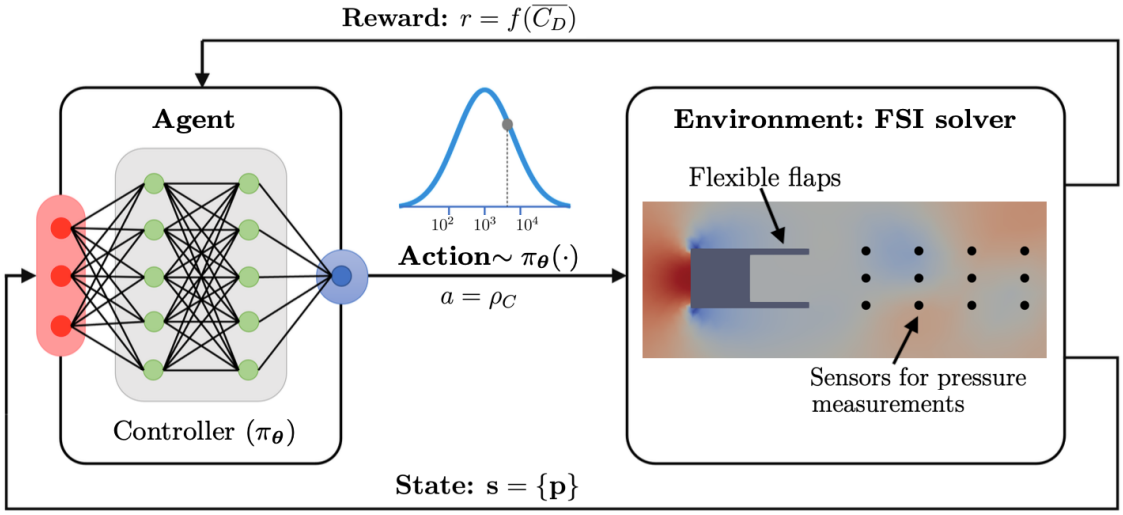


Figure 3.19: System integration overview

3.3.2.1 Action

Action represents the actual contribution by which the agent acts on the environment in order to influence its dynamics. In this case, the action a is represented by the density of the flaps ρ_C, already introduced in the section 3.2.2. In addition, the range in physical terms of variation between 10^3 and 10^6 in section 3.2.5 was also identified.

In practical terms, the actual definition of the control range is dictated by constraints related to the control algorithm. In fact, the two chosen algorithms have two different control space definitions: the PPO requires an action that is continuous within a set range, while the DQN requires a discrete action within the same range.

Two types of actions are then defined according to the algorithm used:

- PPO: the action space is continuous and it is common practice to use a range that is centred in zero, with extremes -1 and 1. At the control level, therefore, the action space is defined in $[-1, +1]$. A transformation function is then applied to obtain a density from the action value that is meaningful in physical terms. This function is shown below
$$\rho_C = 10^{(1.5 \cdot a + 4.5)} \tag{3.3.1}$$
and thus allows to go from the range $[-1, +1]$ of a to the range $[10^3, 10^6]$ of ρ_C.
- DQN: the action space is discretized into two different ranges [-500, +500] for a total of 1001 action points and [-250, +250] for a total of 501 action points. Again, it is necessary to apply the transformation of equation 3.3.1 to obtain a meaningful density value following a rescaling of the ranges between [-1, +1].

3.3.2.2 State

The state of the system is the information that is used by the agent to develop its control logic. The state is composed of the observations, i.e. a temporal extraction of the characteristics of the environment, which is then used as input for the neural networks forming

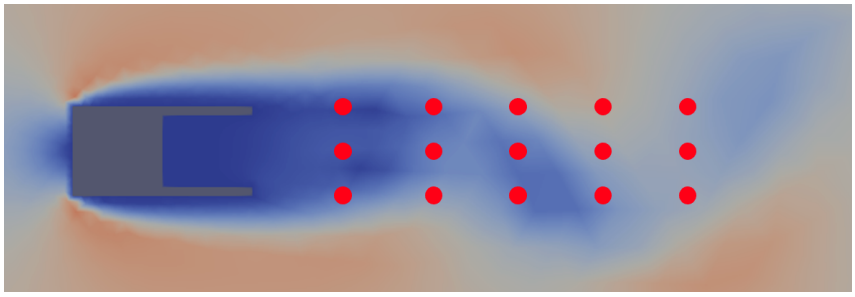
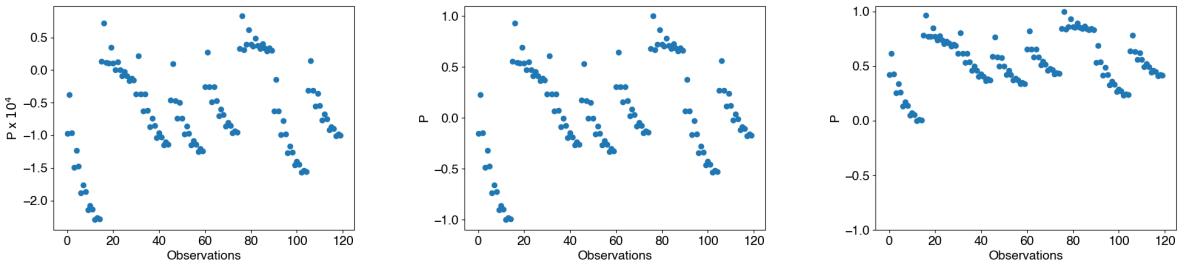


Figure 3.20: Pressure probes positions (red dots)



(a) No transformation (b) Transformation T1 (c) Transformation T2

Figure 3.21: Observations and applied transformations

the controller.

The observations are pressure measurements taken at set points in the fluid domain. The position of the pressure probes was established a priori and did not change in the course of the experiment. It should be noted, however, that the results can also be very susceptible to the choice of these positions, and thus, in a larger study, it is also possible to optimise the position itself.

In the problem under investigation, 15 pressure probes were used, placed in the wake of the square. Five pressure probes are located on the centre-line, the first downstream is placed at a distance of one from the trailing edge of the flaps, while the others are always spaced by one. The same pattern of 5 equally spaced probes is placed parallel to the first 5, but in a vertical position shifted by +0.5 and -0.5, thus being aligned with the upper and lower edge of the square. This creates a grid of pressure probes exactly in the wake of the square. The pressure is measured at each instant of the aerodynamic solver, thus at each Δt of 0.1. A schema of the distribution of pressure probes is shown in figure 3.20.

Since the state is directly the input of a neural network, it is convenient not to have an excessively large state. Indeed, if one considers the 15 probes running for a time of 10 seconds and one measurement every 0.1s, 1500 nodes are obtained. In order to avoid the use of very large neural networks that are expensive to train, a limited number of measurement points was used. For each probe, only the last 5 or last 7 measurements are therefore used as appropriate, resulting in a number of observations per control interval between 75 and 105 for 15 probes.

As with actions, it is also convenient with observations to use transformations to re-scale the values obtained. This is because neural networks are often sensitive to the

absolute value and distribution of the assigned input values.

In particular, 3 approaches were used and can be seen in figure 3.21:

- no transformation (a): in this case, the raw values of the pressures were used. Note that these values are not centred and have very small values, of the order of 10^{-4};
- transformation T1 (b): in this case, a transformation was used which re-scales the values in an interval [-1, +1];
- transformation T2 (c): similar to the previous transformation, in this case the interval is [0, +1].

Note that the transformations do not change the trend and distribution of the observations but simply re-scale the values.

3.3.2.3 Reward

The reward is the actual index of merit of how the action performed by the agent has influenced the environment in order to achieve the intended goal.

In the case of this project, the objective is to minimise the drag coefficient generated by the square cylinder. The system reward R is therefore a function of the drag coefficient, calculated as follows:

$$R = \frac{1}{2}\overline{C_{D_\tau}}^2 - \frac{p_1}{p_2}\left(1 - p_2^{u/u_{max}}\right). \tag{3.3.2}$$

The first term is a scaled measure of the drag coefficient where $\overline{C_{D_\tau}}$ is the mean drag coefficient computed in the control time scale $\Delta\tau$, which is squared to emphasise high values of the coefficient.

The second term, on the other hand, is an auxiliary term that penalises the immobility of the system. In fact, u indicates the number of successive moments in which the action of the system remains constant, without variations. As u increases towards the fixed maximum value u_{max}, the term becomes larger and larger, penalising the rewards greatly. This then turns out to be an incentive for the system to change action, not remaining stuck. In particular, the constants are $p_1 = 0.845$, $p_2 = 10000$ and $u_{max} = 20$ [22].

3.3.3 Training procedure

Once the integration of the two systems is complete, the model training procedure is considered. As a library for the Deep Reinforcement Learning algorithms, Stable Baselines 3 [3] was used, which integrates easily with the Gymnasium environment.

As mentioned above, two algorithms were employed: DQN and PPO. For both, various parameters were tested, which will be detailed in Chapter 4.

In this section, the training procedure is presented, focusing on the technical aspects necessary for the development of control agents. The basis of the training procedure is formed by an episode, i.e. the unit from the start-up of the system until a terminal state is reached. During training, many episodes are tested in order to arrive at a complete and optimal control law. An overview is given in figure 3.22, while the following sections present the details.

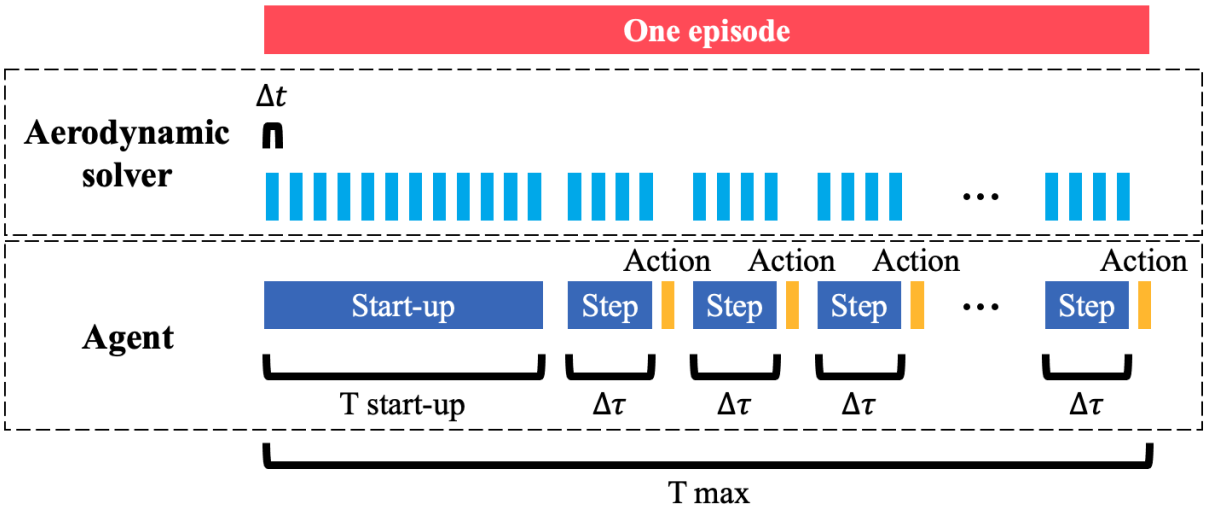


Figure 3.22: Training procedure overview

3.3.3.1 Aerodynamic vs Control timescales

The integrated system is characterised by the presence of two time scales. The aerodynamic time scale Δt is the minimum unit of time with which the time variable in the solver's resolution is discretised. In section 3.1.5.1, constraints were presented for the values that the aerodynamic time scale can take, and for reasons of stability, the value is set at 0.1. The control time scale $\Delta\tau$, on the other hand, is the time interval between two control actions performed by the agent.

In practice, the control scale will always be larger than the aerodynamic scale, which is the minimum observable time unit in the system. The control scale will therefore be a multiple of the aerodynamic scale according to the following logic: between two control actions, time $\Delta\tau$ passes, which is discretised by as many units of the aerodynamic scale. It is important that the aerodynamic system associated with the Δt scale has a transient margin between two control actions in order to resolve the aerodynamic phenomena caused by the action. For this reason, aerodynamic and control scales cannot be the same.

In this study, three different control scales were tested in order to assess their effect on optimal performance:

- $\Delta\tau_1 = 0.5$, 5 times Δt,
- $\Delta\tau_2 = 1$, 10 times Δt,
- $\Delta\tau_3 = 2$, 20 times Δt.

3.3.3.2 System start-up

Every aerodynamic solver requires a start-up phase, in which the system is initialised from the initial velocity condition imposed at the inlet. It should be noted that in this design, a constant, unity velocity profile is imposed at the inlet. An interval of time is therefore required in which the fluid injected into the domain reaches the square, interacts with it and the formation of the vortex wake structures is initiated. For this reason, a T start-up time must be considered at the beginning of each training episode in order to reach an

initiated condition of the aerodynamic environment.

Since the aerodynamic solver is deterministic, i.e. given an initial condition and a calculation time it always arrives at the same solution, a mechanism was introduced to vary the start-up time. In fact, a reference T start-up value is fixed for experimentation, but each time the system is re-started, an actual random T start-up value is chosen in a range of -10% and +10% from the fixed value. This variation is beneficial in that, considering that many episodes will be used in the training course, it allows the agent to always learn from a different initial condition, enriching variety and enhancing learning.

3.3.3.3 System running and truncation

Once the start-up phase has been completed, the system actually begins to learn. Learning is given by successive steps, which have a duration equal to the control time $\Delta\tau$. During each step, the aerodynamic solver calculates the environment with its time scale Δt and collects observations for the state. When the step is completed, the controller receives the new state of the environment and the reward of the previous action. At this point it produces a new action that triggers the next step calculated by the aerodynamic solver.

The working system is then presented as a series of episodes, each consisting of a continuous alternation of steps and actions, until the episode is restarted with a start-up phase.

The terminal state of an episode is reached when the agent reaches its goal (e.g. the end of a maze) or when the system is truncated after passing an imposed condition. For the system considered, it is not possible to identify an end state associated with the attainment of an objective. In fact, it is not possible to identify a meaningful end of the simulation, considering that the aerodynamic simulation can continue for a theoretically infinite time with the continuous action of the agent's control. Furthermore, it may be risky to identify a completion condition in this case, as the system may learn something in reference to this fictitious completion time (i.e. reaching the goal). It is more correct in fact to consider the truncation condition, i.e. the interruption in an episode once the maximum time indicated as T max has been reached. This condition is similar to that of completion, but the system does not learn upon reaching a goal, it is only interrupted and starts learning again at the start of the next episode.

Having completed the training procedure, it was possible to train several models which will be presented in the next chapter.

Chapter 4

Results

In this chapter, the results obtained from training the different models will be presented. In Appendix B are listed all the models trained with the respective parameters employed.

Evaluating a model is an intricate operation as a single index of merit is usually not sufficient. In general, the evaluation of the actual improvement in the system's aerodynamic performance, understood as a reduction in the drag coefficient, is carried out by having as a reference coefficient the one obtained with the same flap geometry but considering the rigid case, see section 3.2.2. In this way, it becomes relevant to evaluate the entity of this improvement at the trained agent's expense.

Questions arise, however, as to what this reduction means, whether this is sufficient or whether a certain type of signal trend is also necessary. In addition, it may also be correct to consider in which time interval the averaging of the coefficient is necessary.

These issues will be answered in the following sections. Presenting the models obtained and proposing a method for cross-sectional evaluation of the results obtained.

4.1 Computing facility

For each training experiment, hyper-parameters are set and the process is then launched on a node of an HPC consisting of 48 cores and 64 GB RAM. On average, this training experiment takes 60 hours. The result of this process consists in a model that is then evaluated. This evaluation involves using the trained agent to control the aerodynamic problem presented. In this way, its policy, i.e. the quality of its learning, can be evaluated.

4.2 Baseline models

It is a good idea to start the parameter tuning process by identifying a baseline model that can be taken as a first reference for subsequent comparisons.

It is known in the literature that Deep Reinforcement Learning models are very sensitive to the choice of parameters in the training phase. In this light, the number of possible parameters on which one can act is very large, and an extensive test of all possible configurations is beyond the scope of this project.

In fact, the objective posed is to identify control logics that can improve performance, being aware of the possible existence of more performing but more complicated models.

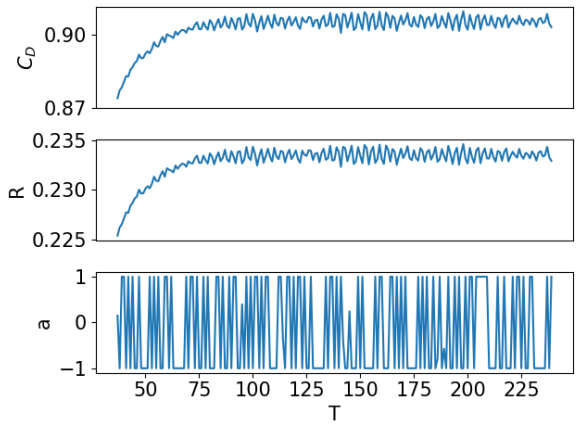


Figure 4.1: Baseline PPO performance

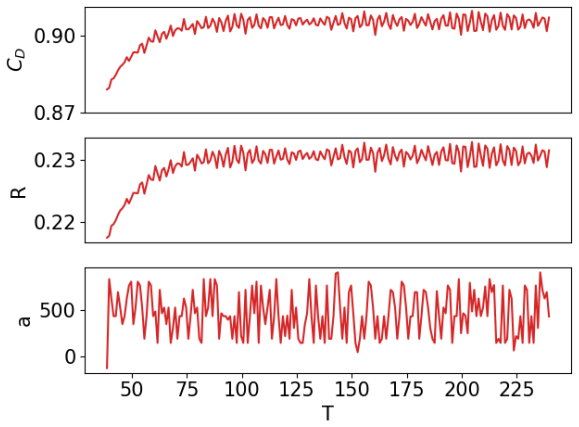


Figure 4.2: Baseline DQN performance

Based on this, the first models produced are simple but functional models that integrate well with the aerodynamic system and it is possible to observe the effective active control they exert on the environment.

Two baseline models were identified, obtained with the PPO algorithm and the DQN algorithm, respectively.

These first models were trained with a set of hyper parameters obtained following some preliminary tests. For the sake of clarity, they are listed below and please refer to section 3.3.3 for technical details:

- T start-up = 40 s, the time required to initialise the system;
- $\Delta\tau$ = 1 s, control period between two actions;
- T max = 240 s, maximum time of one episode;
- 10000 steps to be trained on, which results in 51 episodes;
- 15 probes collecting 7 pressure points for a total of 105 observations.

4.2.1 Baseline PPO

The PPO algortihm is used in MlpPolicy version which employs policy class for actor-critic algorithms.

The results for this model are shown in figure 4.1, where the drag coefficient C_D, the reward R and the action a performed by the agent as a function of time are shown.

It can be seen that the agent produces an action that triggers a consequent increase in reward. It can therefore be understood that the control is in principle effective. The first few seconds of the simulation are still the tail end of the start-up phase and towards the second 75 a full steady state is reached. The drag coefficient trend turns out to be well controlled and does not show peaks and oscillations as was the case with a flexible flap but no control (see figure 3.16).

4.2.2 Baseline DQN

The DQN algortihm is used in MlpPolicy version which employs policy class with Q-Value Net and target net for DQN. In this case, the results are shown in figure 4.2. As in the case with the PPO algorithm, the DQN algorithm shows good trends in the characteristics examined. Drag coefficient and reward show correct trends. With regard to action, however, it appears that the DQN-algorithm succeeds in creating greater nuances of control. It is not possible from here to determine which of the two controllers is acting better, but it might seem that a more varied signal could be an indication of a well-trained control.

From these two models, it was then possible to construct a more detailed analysis of the different parameters at play and assess their consequences on the problem under consideration.

4.3 Change flap geometry

During the preliminary development of the system, and in particular during the analysis on the introduction of control flaps, it was opted to maintain the standard configuration (flaps of unit length and width 0.1), as presented in section 3.2.3.3.

In order to be able to give an assessment of the choice of configuration opted for there, a model with a different configuration was also trained.

In this case, in fact, while keeping the training framework and hyper-parameters constant, only the geometry of the flaps was changed. These therefore have a length of 1.5 by a width of 0.2, as shown in figure 4.3.

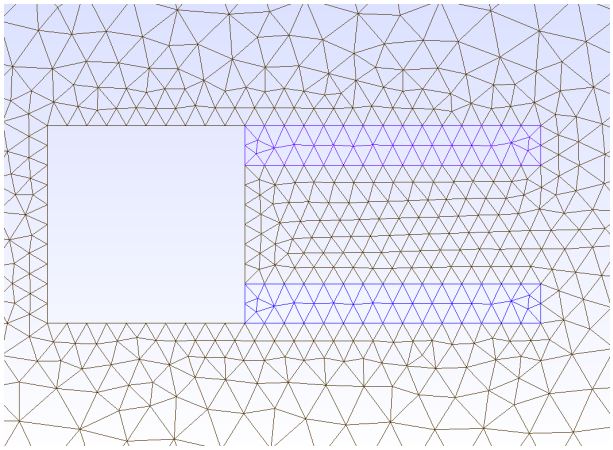


Figure 4.3: Flaps geometry with $L_C = 1.5$, $w_C = 0.2$

In this case, the DQN algorithm was used for both cases. In addition, all training parameters were kept constant: a start-up time of 40 seconds, a control $\Delta\tau$ of 0.5 seconds and a T max of 140. The models were trained for 10000 steps, resulting in 51 training episodes. The training times are comparable: 46 hours for the model with standard geometry and 47 hours for the model with modified geometry.

Figure 4.4 shows a comparison between the control performance of the two models trained using the two geometries. The upper graph shows the calculated drag coefficient, which for comparison purposes has been scaled with the average value of the drag coefficient for that same geometry but with rigid flaps. Basically, it shows the percentage gain

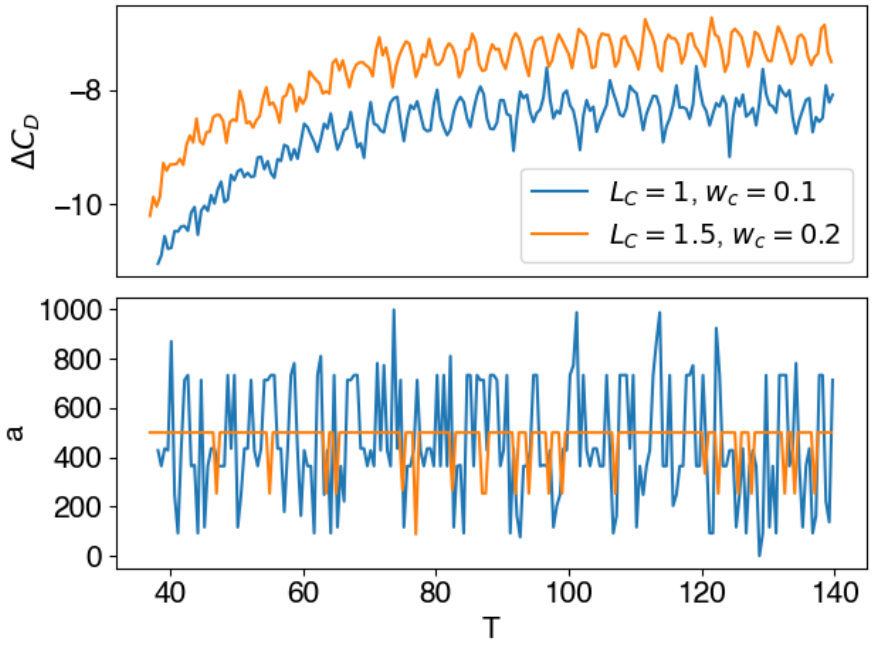


Figure 4.4: Control comparison between different geometries

on the drag coefficient using the control compared to the configuration and rigid flaps. The formula is as follows:

$$\Delta C_D = \frac{C_D - \overline{C_D}^*}{\overline{C_D}^*} \cdot 100 \tag{4.3.1}$$

where $\overline{C_D}^*$ is the mean drag coefficient, for that flap geometry, but considering the structure rigid. Specifically, for the standard geometry the average drag coefficient with rigid flaps is 0.987 while for the modified geometry it is 0.89.

The lower graph shows the signal of the action performed by the agent during the simulation.

At first analysis, it can be observed that both models achieve a reduction in drag coefficient compared to the rigid case. In particular, the standard geometry reduces the drag coefficient by approximately 8.6% while the modified geometry by 7.6%. Looking at the action signal, however, it can be seen that the action with the standard geometry is more varied, thus probably resulting in greater control and thus a greater reduction in the drag coefficient.

The cause of this signal flatness with modified geometry has not been investigated further here. However, it is possible to conclude that, although the hyper-parameters have yet to be fully optimised, the standard model will always be more receptive to control, having a slight but non-negligible coefficient gain compared to the other geometry.

It is possible that a longer and thicker geometry produces a more difficult control on the wake structures of the square. Thus causing the training of a less effective controller. This may further justify the earlier choice of selecting the standard geometry with flaps of unit length and width 0.1

It must be emphasised that the hyper-parameters for each model are certainly very sensitive to the geometric configuration. For this reason, a comparison such as the one

Model	Parameters				Results	
	T start-up	**T max**	$\Delta\tau$	**Episodes**	ΔC_D	**Total time [h]**
Baseline PPO	40	240	1	51	$\approx -9\%$	67
PPO 2	40	240	2	101	$\approx -8\%$	87
PPO 3	50	340	2	69	$\approx -8\%$	74
PPO 4	40	140	0.5	51	$\approx -9\%$	67

Table 4.1: Tested models for time scales influence

made, without an optimisation loop of the same, is not too precise. In fact, it is very possible that the parameters used for the unit flap configuration and width 0.1 are not ideal for the 1.5 flap and width 0.2. Aware of this limitation, the main intent was only to have a performance comparison, given the change in geometry used.

4.4 Time scales influence

The main parameters on which to act outside the algorithms' internal parameters are the time scales defined in section 3.3.3. Changing these scales influences the dynamics of the environment but also the way the algorithm learns, acting among other things on the number of episodes on which to train the agent.

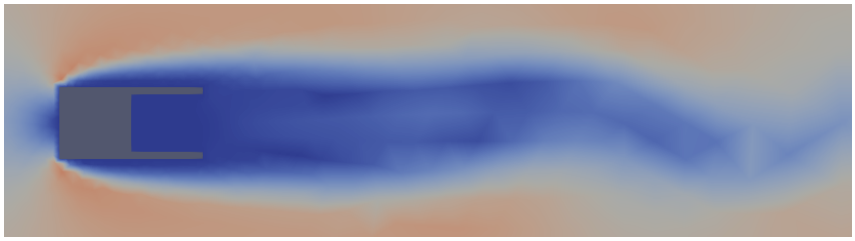
A preliminary analysis was therefore introduced to understand the influence of these scales on the performance of the models, in order to choose a set of parameters that is meaningful.

In particular, it is possible to act on:

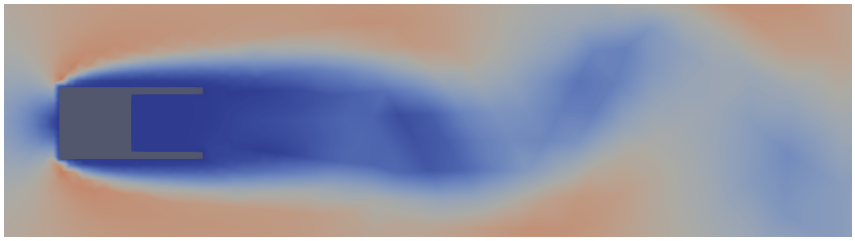
- T start-up: this parameter influences the time at which the model begins to learn. On a computational level, a longer time also increases the solving time as it is not included in the total step count for training. Therefore, it would be ideal to find the minimum time that allows the solver to complete the start-up phase correctly and does not lengthen the duration of each episode too much;
- T max: like the previous one, it influences the number of steps in each episode but has no important aerodynamic significance. Therefore its choice is more free and only aimed at having an adequate number of steps;
- $\Delta\tau$: this is the most delicate parameter and the most difficult to predict. The dynamics of the system depends heavily on this parameter (as it temporally separates two actions that will affect it) but also the learning of the agent by changing the shape of the observations and thus the state of the system. It would be ideal to find a compromise point where time is neither too short to truncate the flow dynamics nor too long to make the signal stationary between two actions.

In order to assess the influence of these parameters, 3 sets of parameters were tested for 3 models with a PPO algorithm and 10000 steps, compared to the baseline case introduced previously. Table 4.1 shows the cases tested, with the parameters used and the results obtained in terms of performance: the scaled drag coefficient as in equation 4.3.1 and the total time for training the model.

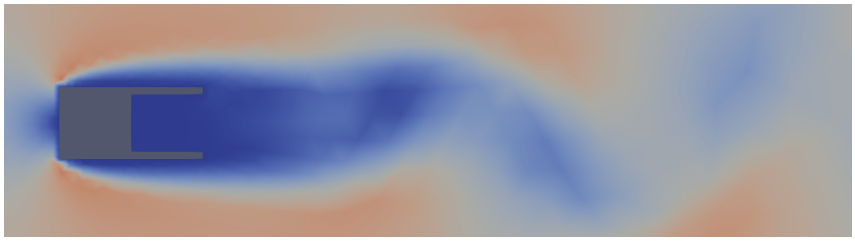
All models presented report a gain in drag coefficients. However, it can be observed in the PPO 2 and PPO 3 models that using a high $\Delta\tau$ of 2 penalises the drag coefficient



(a) T start-up = 30 s



(b) T start-up = 40 s



(c) T start-up = 50 s

Figure 4.5: Velocity fields comparison between start-up times

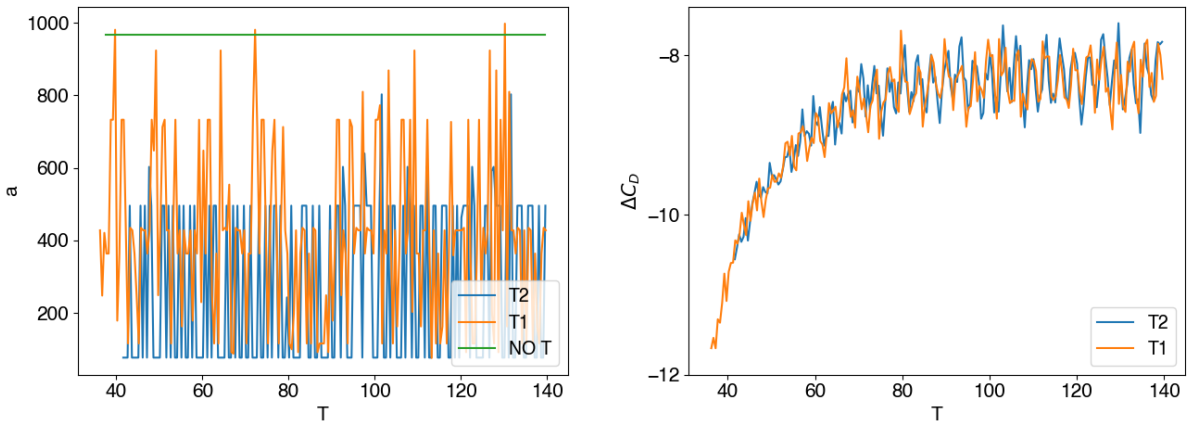
gain by approximately one percentage point. This is not a drastic worsening, but it may indicate an incorrect setting of the parameter. In fact, the baseline case and PPO 4, with lower $\Delta\tau$ of 1 and 0.5 respectively, have greater gains. It should be considered that tau 0.5 can be regarded as a limit value that allows the system to develop its dynamism correctly between two actions.

As far as the action of T start-up and T max are concerned, no particular consequences on their value have been identified except for the system's resolution time. Considering also the effect on aerodynamics of the T start-up, figure 4.5 shows the velocity field reported for different time instants. It can therefore be seen that at 30 s, the wake is still forming, while at 40 s and 50 s, alternating vortical structures can already be observed. For this reason, it is therefore convenient to opt for a start-up time of 40 s, reducing the total resolution time but maintaining the correct dynamics of the system.

4.5 Transformations of observations

An important aspect, already presented in section 3.3.2.2, concerns the management of the state of the environment and how it is manipulated and then used in the neural network for control logic.

Three approaches were tested for the management and transformation of pressure measurements made by the aerodynamic solver. Please refer to the mentioned section for details, but summarising the three approaches are:



(a) Action (b) Scaled drag coefficient

Figure 4.6: Comparison between transformation of observations

- No transformation (NO T): state is used raw;
- T1: re-scaling in [-1, +1];
- T2: re-scaling in [0, 1].

Again, the models were trained for 10000 steps with standard parameters: start-up time 40 seconds, control $\Delta\tau$ of 0.5 and T max 140. In addition, 15 probes were used for a total of 105 observations. The comparison is shown in figure 4.6, where the trend of the action and the scaled drag coefficient as presented in equation 4.3.1 is shown.

The first feature to note in the trend of the action in figure (a) is that for the case without application of the transformation function, the controller produces a constant signal. In this case, it is therefore evident that the training process has failed. This is possible since the raw pressure signals have very small values, a factor which could have a significant impact on the training process of the neural networks.

With good reason, therefore, the two transformation functions were introduced which, in the same figure, we note were able to produce a rich and active action signal.

Focusing instead on the gain alone in terms of drag coefficient (b), the trend and values for the two cases are very similar. For this reason, it can be concluded that the application of a transformation is necessary for the correct training of the models, but the system is not particularly sensitive to the choice of the same. We therefore chose to continue with the T1 transformation in the interval $[-1, +1]$ only because it is the most standard and widespread in the literature.

4.6 Reduced action space

Another way to change the characteristics of the agent's training modes is to change the action space in which the possible actions are defined.

In the case of the continuous action space of the PPO, it was preferred to keep the standard interval between -1 and +1. In the case of the DQN algorithm with a discrete action space, however, it is possible to act on the extremes and the number of points in

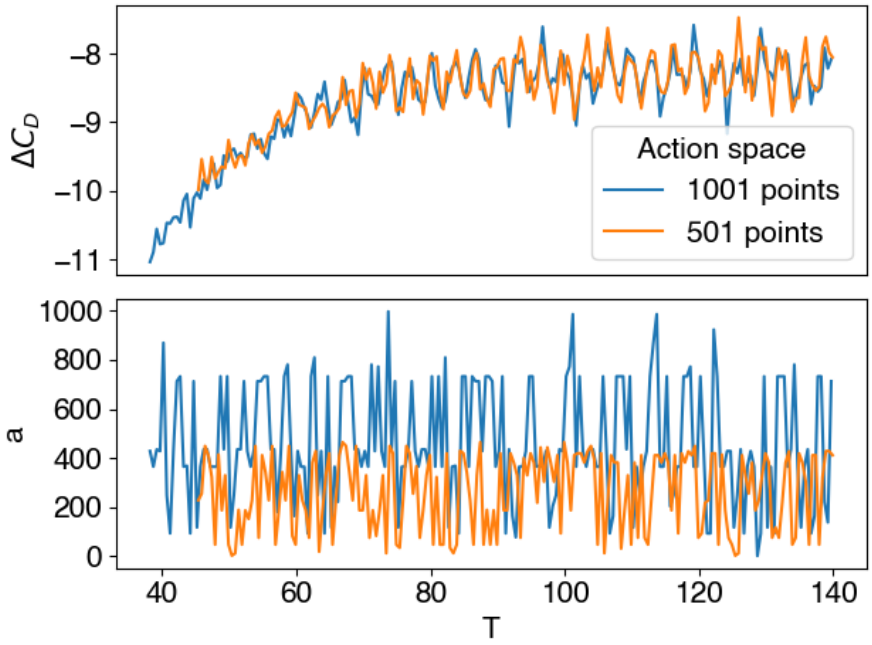


Figure 4.7: Control comparison between different action spaces

which the interval is discretized.

So far, a 1001-point action space has been used, thus having a large variety of points (i.e. actions) available to the agent. A more constrained case was then tested, halving this value to 501. In this case, the agent has the same physical range of actions between 10^3 and 10^6, but with a smaller number of points at which it is discretized.
The other model parameters used are the standard ones employed in the previous cases.

The results are shown in figure 4.7. While the action shows correct differences in terms of signal dominance, considering the different definition of the action spaces, the trend of the scaled drag coefficient is almost the same.

It was therefore not possible to evaluate a beneficial or maleficent effect on the change of the action space in an attempt to constrain the agent's choice more. It might be interesting to evaluate even greater decreases or increases in the space in order to have more extreme cases to evaluate. However, this analysis is postponed to future studies.

4.7 Controller neural networks

As already presented, the inherent peculiarity of deep reinforcement learning lies in fact that it combines deep neural networks, which are excellent at handling complex patterns and representations in data, with reinforcement learning.

The definition of the neural networks immersed in the algorithms is of utmost importance with regard to learning performance and obtaining satisfactory results.
A tuning process was also carried out in this analysis in order to assess the impact of the different neural networks on the agent's control mode.

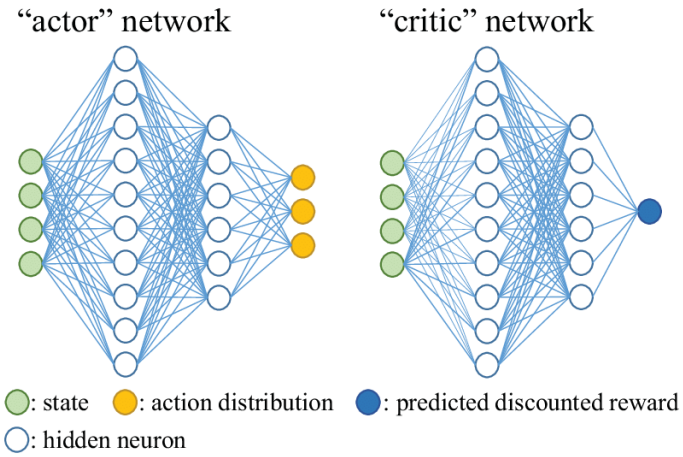


Figure 4.8: Overview on actor and critic neural networks in PPO [10]

It should be emphasised that the degrees of freedom granted by this type of analysis are many, as neural networks are essentially modular entities with several adjustable layers. Moreover, in addition to learning from the direct testing of proposed configurations, there is a great deal of expertise gained through experience in order to hypothesise better a priori configurations. In view of the great plethora of possibilities, the analysis presented here is not aimed at identifying the best possible neural network, but was limited to a rough test of these networks, modifying the main parameters.

Both proposed algorithms, PPO (section 3.3.1.2) and DQN (section 3.3.1.3), consist of two neural networks each. As far as PPO is concerned, the two neural networks are shown in figure 4.8 which are an actor network whose input is the state and output the action, and a critic network whose input is the state and output a prediction of the discounted reward.

Concerning the DQN, the two networks are the Q network and the Target networks: the Q network is the agent that is trained to produce the optimal state-action value while the Target network is a copy of the Q network and is used to approximate the Q-function to improve the stability of the training.

In particular, the structure of the neural networks for the two algorithms are presented below. For simplicity, consider the case the 15 pressure probes with 7 pressure points each, resulting in a total or 105 observations. This value can change when changing the number of probes and pressure points as stated.

For the PPO, the actor and critic networks have identical structure with the base value of the number of hidden nodes being 64:

- features extractor: simple layer to flatten passed input
- a fully connected linear layer with input the number of observations (105) and output the hidden nodes (64)
- Tanh activation function
- a fully connected linear layer with input the hidden nodes (64) and output the hidden nodes (64)
- Tanh activation function

- a fully connected linear layer with input the hidden nodes (64) and output the action space of dimension 1

For the DQN, the basic structures of the two neural networks (Q network and Target network) are shown below, with the base value of the number of hidden nodes being 64:

- features extractor: simple layer to flatten passed input
- a fully connected linear layer with input the number of observations (105) and output the hidden nodes (64)
- ReLu activation function
- a fully connected linear layer with input the hidden nodes (64) and output the hidden nodes (64)
- ReLu activation function
- a fully connected linear layer with input the hidden nodes (64) and output the action space (1001)

With regard to the experimentation carried out, it was therefore decided to modify the value of the hidden nodes for each of the layers in the networks presented here. In particular, three values were used, as shown in table 4.2, with their associated trainable parameters.

Hidden nodes	**Total trainable parameters**	
	PPO	**DQN**
64	22144	150144
128	60672	316672
256	186368	698880

Table 4.2: Neural networks parameters

Figure 4.9 shows the development of the scaled drag coefficient (as per equation 4.3.1) for the tested case studies, both for models with PPO (blue lines) and models with DQN (red lines).

In addition, table 4.3 shows the average values of the scaled coefficient.

Hidden nodes	$\overline{\Delta C_D}$	
	PPO	**DQN**
64	-8.35	-8.34
128	-8.41	-8.45
256	-8.41	-8.42

Table 4.3: Neural networks scaled drag coefficient

It can be observed that the coefficient trends show similar behaviour, although an increasing signal fluctuation can be observed as the structure of the neural network increases. Furthermore, the smallest networks, with 64 hidden nodes, are also those that achieve a lower drag coefficient gain. This could therefore mean that an increase in control oscillations, combined with an increase in flap oscillations, could result in a beneficial effect for the reduction of square drag. In order to fully justify this thesis, an in-depth

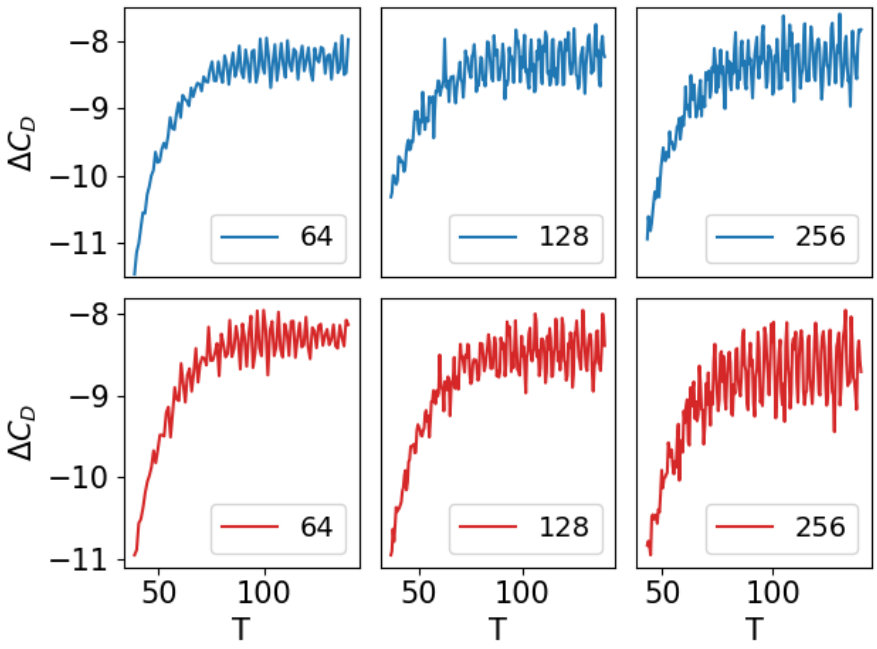


Figure 4.9: Control comparison between different neural network structures: PPO (blue) and DQN (red)

analysis of the aero-elastic dynamics of the flaps would be required, which is outside the scope of this project.

It is therefore possible to establish that a larger neural network (with 128 or 256 hidden nodes) may represent a better configuration, imposing greater oscillations resulting in improved performance.

4.8 Changing learning rates

Changing learning rates in Deep Reinforcement Learning is a common technique used to optimise the training process and improve the stability and convergence of reinforcement learning algorithms. Learning rates control the magnitude of parameter updates during training and are crucial for ensuring that the learning process converges to an optimal policy efficiently.

The choice of learning rate strategy depends on the specific DRL algorithm, the problem domain, and the available computational resources. Experimentation and tuning are necessary to find the best learning rate schedule for a particular task.

With this in mind, two models were trained with different learning rates (considering that the default value used so far is 0.0001). The two values tested are therefore an order of magnitude higher and lower than the default value, using the values 0.001 and 0.00001 respectively.

The results are shown in Figure 4.10, respectively for the two learning rates considered reward and action in a time window of interest.

The main observation that can be made concerns the trend of these signals. In terms of drag coefficient gain, the difference is not much, with both producing a gain averaging around -8.5%. But the main difference can be seen in the performance of the controller. In

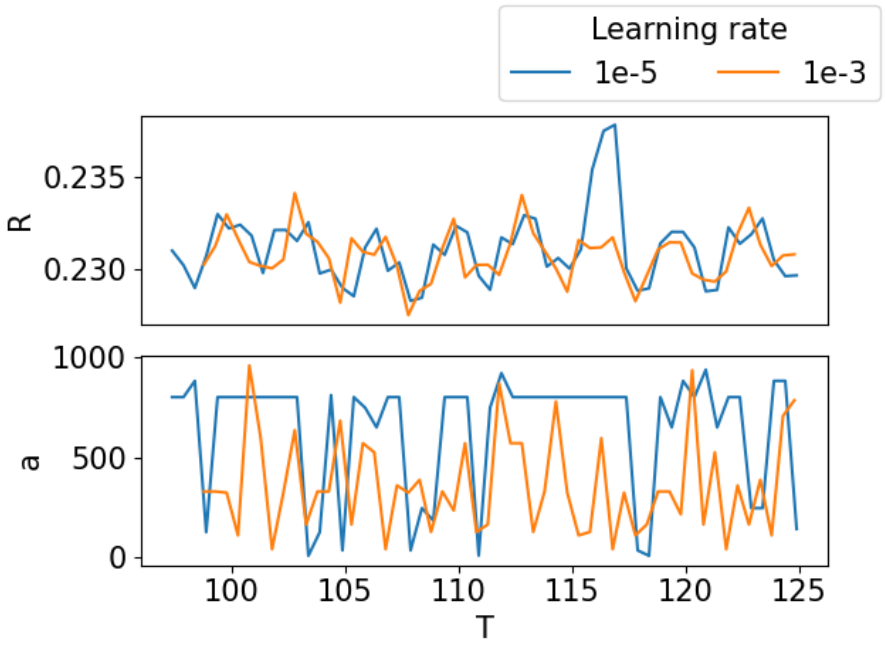


Figure 4.10: Control comparison between different learning rates

fact, it is possible that the learning rate of 0.00001 is too small for the tested configuration. In fact, it is possible to see that the signal (blue line) is flat and constant in several sections, even engaging the reward peak associated with a drag coefficient peak to force a change of action. It should also be noted that the periodicity of the reward signal is due to the intrinsic periodicity of the aerodynamic system, generated by the alternating detachment of the wake vortices from the edges of the flaps.

4.9 Final configuration

As the last configuration tested, it was opted to form a case study that combined all the evidences of improvement found so far. Furthermore, it was opted to use the DQN model as it demonstrated a better performance of the agent during experimentation, producing a rich and effective control signal. To do this, the last configuration was trained with the following characteristics:

- DQN model, 10000 steps, 0.001 learning rare and 128 hidden nodes;
- 15 pressure probes, T1 transformation and an action space of 1001 points;
- 40 seconds of start-up, 140 seconds of T max and 0.5 of $\Delta\tau$.

The results of the scaled drag coefficient are shown in figure 4.11, together with the trend of the baseline model.

With regard to the mean values of the scaled coefficient, the final model reports an improvement of approximately 0.2%, thus resulting in a scaled coefficient between 8.7 and 8.8%. This result is very promising and thus demonstrates the effectiveness of this type of control.

Evaluating in detail the trends presented in the figure. It can be observed that the pattern of the final configuration shows more marked oscillations, although the average

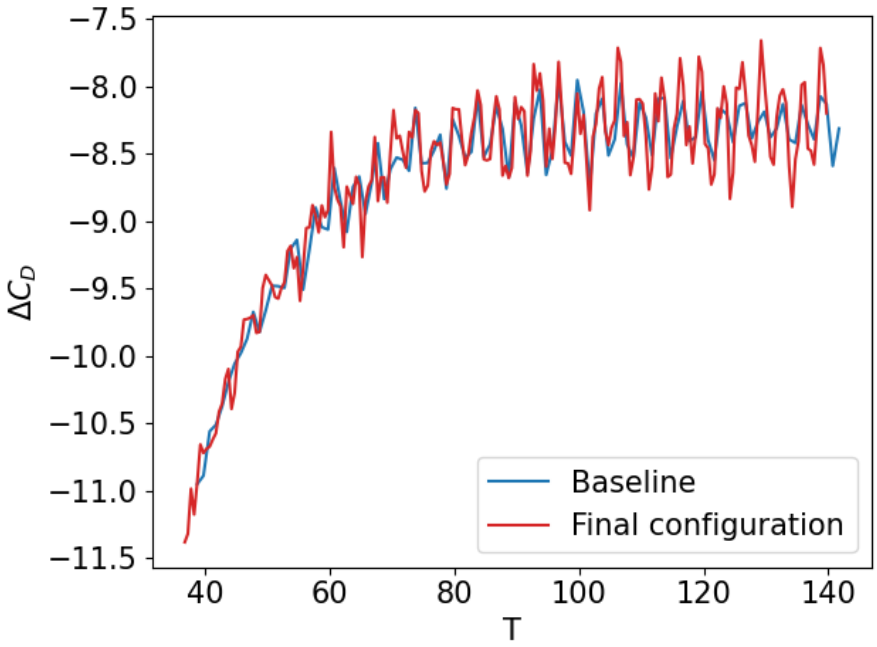
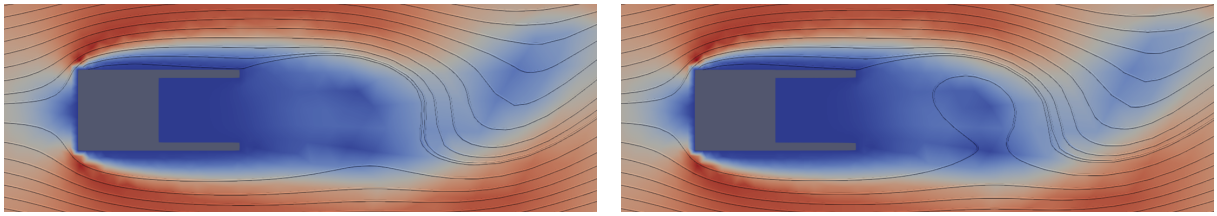


Figure 4.11: Comparison between DQN final configuration and



(a) Baseline (b) Final configuration

Figure 4.12: Comparison between velocity fields

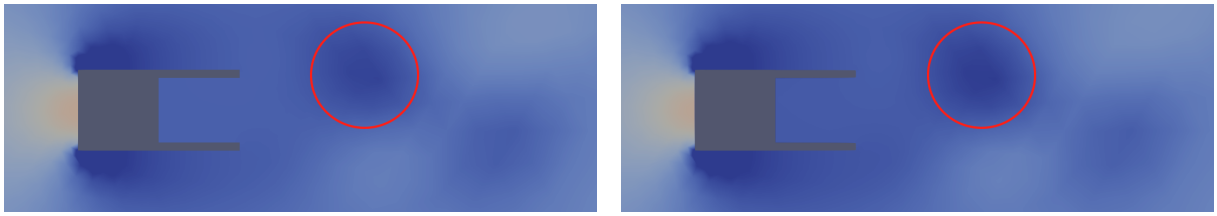
value tends to be similar. These oscillations could be the key to explaining the operation of this type of control and to justifying the improvement in the drag coefficient.

For a more detailed analysis, the velocity field (figure 4.12) and the pressure field (figure 4.13) for the baseline case and the final configuration have been reported.

The oscillation of the pressure signal is due to an increased intensity of the vortices that periodically break away in the wake of the square. Through the velocity field figures, it is possible to visualise the increased formation of a wake vortex, close to the flaps, from the reproduced semi-circular current line, in the final configuration compared to the baseline.

Although more difficult to visualise, the pressure field also shows this behaviour. Looking closely at the area of the figure inside the red circle, it's possible to see a slightly darker colouring for the case of the final configuration. This demonstrates a lower pressure in the centre of the vortex and thus a higher intensity of the vortex.

The wake vortices, like any other fluid-dynamic vortex, are regions that induce a circular motion of the fluid with a lowering of pressure in the centre of the vortex and an increasing velocity with respect to its centre. In these figures, it is not possible to see a



(a) Baseline (b) Final configuration

Figure 4.13: Comparison between pressure fields

fully formed vortex as the main velocity component remains the horizontal component of the current.

It is therefore possible that the induced flow velocity re-energises the fluid layers closer to the square and flaps, pushing high-energy air from the undisturbed current. This re-energising could be at the root of the drag reduction process, reducing the pressure imbalance between the two faces (front and rear) of the quadrant. This process is obviously on a small scale and therefore gives a reduced gain in the drag coefficient.

This concludes this section in which many models were tested in order to obtain a fine-tuning of the many available hyper-parameters. In the following section, the results presented here will be evaluated.

Chapter 5

Discussion

The developed framework provided several trained models using different parameters inherent to the aerodynamic solver and agent definitions (as listed in Appendix B). For this wide variety of hyper-parameters, the final evaluation of the results obtained can be carried out on different perspectives of analysis.

In terms of the control, the main objective of the models was to reduce the drag coefficient generated by the square in the current. The question had already been raised as to the significance of controlling an aerodynamic characteristic such as drag. With this in mind, the models not only resulted in improved aerodynamic performance through drag reduction, but also involved active control over the wake structures of the square. Indeed, the drag coefficient trends are well limited and controlled, with no oscillatory behaviour. Figure 5.1 summarises the coefficient trends for the different cases that make sense to compare. The case with rigid flaps, with flexible flaps at constant density and with flexible flaps under the action of agent control is shown.

It is evident that the use of flexible flaps already results in a good improvement over the rigid flap case. Furthermore, it is appreciable how the agent's action manages to actively control the drag coefficient signal, damping the intense oscillations of the free flexible case. Although the gain in average terms is small, but still significant, the improvement in coefficient performance due to the direct action of the use of deep reinforcement learning is important and evident.

In terms of numerical procedure, the developed framework proved to be effective in training and evaluating the models. The need for the use of a High Power Computing facility in order to be able to test these models is not insignificant, considering the large resources required for the resolution of aerodynamic models.

From the various parameters that were tested, certain improvements were evident for which an explanation or rationale could be given. By acting on the structure of the neural network and the parameters concerning the aerodynamic solver, it was possible to improve control. However, a difficulty was found in obtaining a net improvement in the drag coefficient. This behaviour could be due not so much to a problem with the control, the algorithm or the parameters used, but rather to the flap configuration itself. It is indeed possible that the proposed geometry is controllable, but only up to a certain margin of improvement.

Changing, for example, the mounting angle of the flaps or modifying the dynamics by introducing the use of rigid flaps with a single torsional spring at the root, it is possible

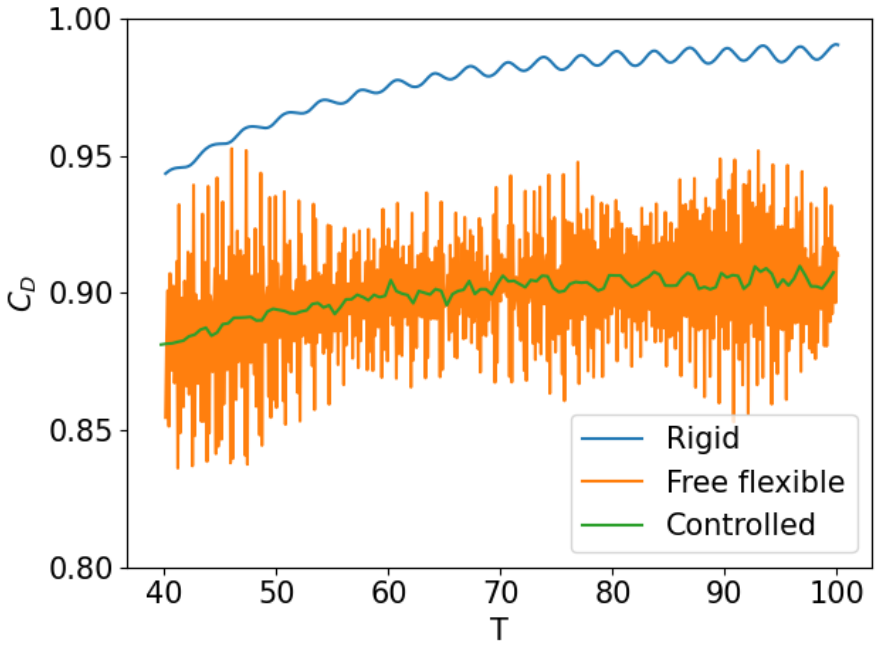


Figure 5.1: Final comparison of drag coefficient

that more sensitive systems to changing hyper-parameters would be obtained.

In terms of aerodynamics, a non-negligible element to take into account is the complexity involved in aerodynamic applications and studies. In fact, despite a rigorous and precise formal mathematical and physical definition, and despite the use of well-tested and experimentally proven numerical resolution techniques, it remains an area of study that is extremely tied to the direct experience of researchers, through their own know-how linked to the individual application. In fact, aerodynamics is not very transversal: an application that works very well for one problem may work very badly for a different but very similar problem. In this scenario, it is therefore very risky to take it for granted that the applications tested in this project can be extended without problems to other control cases.

In contrast to previous approaches, the approach applied to this problem was fundamentally different. While previous works typically involved integrating a Reinforcement Learning algorithm into an existing physical problem, here it was approached from a reverse perspective. The entire problem is viewed through the lens of RL and a comprehensive pipeline is developed to enable a complete RL-centric approach. This approach allowed for a holistic examination of the problem, fostering the creation of a flexible framework that supports replication and experimentation with various RL techniques. By taking this reverse approach, it's been possible not only to enhance the understanding of the problem but also to empower further exploration and innovation in the realm of RL for this specific domain.

However, it remains to be emphasised that the main objective of the project, in relation to research in the field, was not to obtain a ready-to-use technology but to carry out a detailed analysis on a possible application, in order to prove the actual usefulness and

interest of applying artificial intelligence to aerodynamics.

From this point of view, it is therefore possible to state that the work presented here is an original piece of research, having worked on an unexplored domain and having obtained positive results on the application tested. Without claiming to be a completely exhaustive study of the domain presented, the author is sure of the originality introduced. It therefore represents well a piece of the mosaic of studies and applications that can create the foundations for a broader recognition of the usefulness of combining aerospace engineering and artificial intelligence.

Chapter 6

Evaluation, Reflections, and Conclusions

The study carried out for this project was aimed at analysing and defining a controller based on deep reinforcement learning to optimise the aerodynamic performance of a square cylinder.

In my perspective, this project has been properly executed, introducing a important degree of innovation and adhering meticulously to a rigorous scientific methodology. The endeavour has not only provided valuable insights on a cutting-edge technology like Deep Reinforcement Learning (DRL) but has proved its formidable potential and its challenges, both in general and when applied to the intricate domain of aerodynamics. The project achieved good results but, if I were to make an improvement, I'd explore different control methods. Specifically, I would consider exploring the use of different flaps or an alternative mounting configuration, as this adjustment might offer a fresh perspective and potentially yield even more compelling outcomes in optimising aerodynamic performance.

The following sections will provide more in-depth views on the conclusions and reflections concerning this project.

6.1 Project review

This project was designed and developed at the intersection of artificial intelligence and aerodynamics. The first, fundamental part involved the creation and validation of the aerodynamic environment. This phase involved the definition of the problem under examination and the creation of a correct aerodynamic mesh, the basis of any numerical simulation. The correctness of the mesh is of fundamental importance for the subsequent production of reliable, high-performance data and simulations. The mesh study therefore focused on identifying the best compromise between the correctness of results and the lowest associated resolution time. At this stage, fundamental parameters such as Reynolds number and aerodynamic time discretization were also identified and established.

Having completed this first part of defining the mesh, it was time to introduce the control. This introduction took place gradually, to assess its contribution step by step. At first, the two rigid flaps were introduced and an initial analysis of their result was carried out. Subsequently, a sensitivity analysis was carried out on the geometric parameters of the flaps to assess the best configuration. Finally, a detailed study was carried out on the true control parameter, i.e. the density of the deformable flaps. It was only after this analysis that the possibility of controlling the system through action on the flap density

was confirmed.

The interface between the developed aerodynamic environment and the libraries with the deep reinforcement learning algorithms was then realised. This interface was realised to ensure the correct formulation of the training procedure as presented in this report. The actual training and evaluation phase of the control agents then began. This was an iterative phase in which various hyper-parameters were tested and evaluated. The aerodynamic characteristics of the solutions were also evaluated to validate them on a fluid-dynamic as well as a numerical level.

In particular, the main aerodynamic parameters of the training procedure were acted upon, varying the truncation time (and thus the number of episodes on which the agent is trained), the start-up time and the control time interval. The best values for subsequent trials were then identified by acting on the training parameters. The best structure of the neural network, the best learning rate, and state management were identified. In this way, it was possible to arrive at a final configuration, using the DQN algorithm, which resulted in an improvement of approximately 9% in the drag coefficient.

These results demonstrate the potential of DRL in optimising aerodynamic performance. The ability of DRL algorithms to adapt and learn complex control policies in high-dimensional state spaces makes them well-suited for improving the aerodynamic efficiency of various systems. DRL can learn intricate control policies that optimise aerodynamic parameters. This project highlights the importance of synergies between aerodynamics, artificial intelligence, and control theory.

Despite its great potential, studies on the application of artificial intelligence to aerodynamics and, in particular, its active control, are still limited. However, research has recently been moving in this direction and more and more articles are being published on the subject.

Of the various technologies being tested, deep reinforcement learning seems to be the most promising so far. Combining the versatility of reinforcement learning with the depth of deep learning, DRL performs well in fluid dynamics applications.

Currently, these algorithms have been applied to various forms of aerodynamic control. While the nature of the problem is very varied (the applications cover the most diverse problems, from the aerodynamics of airfoils to the swimming strategy of two fish), the goal is often unambiguous: to improve aerodynamic performance. This translates into better systems (generating less friction or producing more lift) which, by extending the analysis, could lead to a reduction in the costs and resources required.

Despite the wide variety of problems in the literature, the control of a bluff body such as a square using flexible flaps had not yet been proven. Other major studies have instead controlled the system using fluid jets or the synchronised movement of rigid flaps. This aspect makes the problem presented original, the results of which can be considered innovative in that they are produced by a configuration of the problem that has not yet been tested in the literature.

6.2 Improvements areas

Two main areas for improvement were identified during the different phases of the project. These concern the flap dynamics and the tuning of the structure of the neural networks.

The deformation of a structure is a complex process involving many degrees of freedom. In this analysis, the complexity is increased by the fact that the interaction of a solid with a fluid creates interaction phenomena that are studied in aeroservoelasticity. This is a multidisciplinary technology which deals with the interaction of flexible structures, the steady and unsteady aerodynamic forces resulting from their motion, and the control systems. This branch has complex tools aimed at evaluating and understanding these solid-fluid interaction dynamics. This interaction is the basis of the control system presented here. It is only through a greater understanding of this interaction that it is possible to have a clearer vision of how to act on the control developed. Therefore, a more in-depth study of how the flap behaves and how its movement influences the fluid field in which it deforms would have been very helpful in improving the design and the results obtained.

The study shows that DRL for aerodynamic optimisation is not without challenges. Hyper-parameter tuning is a crucial aspect of DRL for aerodynamic optimisation. The sensitivity of DRL algorithms to hyper-parameters means that extensive experimentation and tuning are often necessary to achieve optimal results. This process can be time-consuming and resource-intensive. For these reasons, an extensive search for the best hyper-parameter setting could not be carried out due to the time constraints of the project. This, although a limitation on experimentation, does not reduce the validity of the results obtained. These should therefore be understood as an initial demonstration of the effectiveness of the proposed solution. Bearing in mind the limitations presented here, future studies and applications will be able to remedy them and achieve better results and performance.

6.3 Future developments

As reported, the project aimed to extend the capabilities of applying reinforcement learning techniques to fluid dynamics. This involved the development of an original domain and the implementation of an integrated control system. The nature of this project was therefore to carry out a preliminary study to understand the feasibility and performance of the system. From these results, therefore, several lines of research for future studies and applications may arise.

Future research should explore novel flap configurations that can potentially offer improved aerodynamic performance. This involves not only variations in the geometry of the flaps but also investigating advanced computational methods and optimisation algorithms that can be employed to systematically explore the design space and identify optimal configurations for specific applications.

To enhance the accuracy of observations forming the state, it is crucial to optimise the placement and number of sensors in the flow field. Advanced sensor fusion techniques and machine learning algorithms can be applied to determine the optimal sensor configurations for obtaining critical data on aerodynamic performance, flow characteristics, and structural behaviour.

Mesh convergence analysis is a fundamental step in computational fluid dynamics simulations. Future studies should focus on refining mesh convergence analysis methodologies by incorporating control parameters to assess the convergence of not only numerical solutions but also physical quantities of interest. This will lead to more robust and reliable CFD simulations.

As already mentioned, the success of deep learning algorithms in aerodynamic simulations relies heavily on hyper-parameter tuning. Ongoing research should investigate new hyper-parameter optimisation techniques to identify optimal settings that can significantly enhance the accuracy and efficiency of control.

Finally, transitioning from 2D to 3D simulations is crucial for a more realistic representation of complex aerodynamic phenomena. Future studies should invest in the development of 3D simulations. This will enable a deeper understanding of the intricate flow patterns and interactions in three-dimensional space. This could also lead to wind tunnel experiments, which remain a vital component of aerospace research.

Appendix A

Code and Repository

All code developed in this project is public in the DELAC (DEep Learning Aerodynamic Control) repository at link `https://github.com/LaerteAdami/DELAC`.

The main script of this project is `main.py`. From this file, the entire training and evaluation procedure of the models can be controlled. It is possible to define various hyperparameters inherent to the aerodynamic solver or DRL models. In addition, there is a script `utils.py` that contains various methods useful for the operation of the process.

A.1 Mesh generation

The preliminary operation that is performed is the generation of the aerodynamic mesh. The script `mesh_generation.py` and the files relevant to this procedure are contained in the `Mesh` directory. The script contains only one function for generating a mesh from the template file `geometry_2d.template_geo`. In this file, the geometric dimension values for the square, the flaps and the mesh in general are compiled. The result of the procedure is a `geometry_2d.xml` file ready to be used by the aerodynamic solver.

A.2 Aerodynamic Solver

This functionality is based in the directory `Aero` and there are two main scripts.

`run_aero.py` is the actual configuration file for the turtleFSI solver where all parameters, mesh domains and borders, aerodynamic boundary conditions, pressure probes positions and aerodynamic simulation initiation and finalisation routines are defined.

Instead, `aero_main.py` is the control centre for all operations inherent to the aerodynamic solution. It is a collection of functions (`generate_mesh`, `aero_startup` and `aero_step`) to make low-level functionality available to the aerodynamic environment. This is where the mesh generation process is called or the aerodynamic solver `turtleFSI` is called with the appropriate configurations.

The results of the aerodynamic solver are saved inside a directory with the name of the experiment inside `Aero/Results/`.

A.3 Aerodynamic Environment

The aerodynamic environment is formed using the Gym environment framework. The reference directory is `System` where the script `aero_environment.py` is found. This file defines the `AeroEnv` class, which contains all the attributes and methods required for the integration of aerodynamic processes with the DRL library routines.

In the training phase, the output is a trained agent, which is then saved in the `System/Agents/` folder with its checkpoints. In the evaluation phase, on the other hand, the csv files of the control performance (i.e. the time, drag coefficient, reward and action) are save in the `System/Results/` folder.

Appendix B

Main trained models

Table B.1 shows the main training parameters of the main models presented in this report.

Exp name	**Geometry**	**Algorithm**			**Action and State**			**Aerodynamic Environment**			
		Policy	*Learning Rate*	*Hidden nodes*	*Action space*	*Obs Transf.*	*Pressure probes*	*T startup*	$\Delta\tau$	*T max*	*Max steps*
ppo_1	Standard	PPO	0.0001	64	-	T1	15	40	1	240	10000
dqn_1	Standard	DQN	0.0001	64	1001	T1	15	40	1	240	10000
ppo_2	Standard	PPO	0.0001	64	-	T1	15	40	2	240	10000
ppo_3	Standard	PPO	0.0001	64	-	T1	15	50	2	340	10000
dqn_2	Standard	DQN	0.0001	64	1001	T1	15	40	0.5	140	10000
dqn_3	Standard	DQN	0.0001	64	1001	-	15	40	0.5	140	10000
dqn_4	Standard	DQN	0.0001	64	1001	T2	15	40	0.5	140	10000
dqn_5	Standard	DQN	0.0001	128	1001	T1	15	40	0.5	140	10000
ppo_4	Standard	PPO	0.0001	128	-	T1	15	40	0.5	140	10000
dqn_6	Standard	DQN	0.0001	64	501	T1	15	40	0.5	140	10000
dqn_7	Big, long flaps	DQN	0.0001	64	1001	T2	15	40	0.5	140	10000
dqn_8	Standard	DQN	0.0001	256	1001	T1	15	40	0.5	140	10000
dqn_9	Standard	DQN	0.00001	64	1001	T1	15	40	0.5	140	10000
dqn_10	Standard	DQN	0.001	64	1001	T1	15	40	0.5	140	10000
ppo_5	Standard	PPO	0.0001	256	-	T1	15	40	0.5	140	10000
dqn_11	Standard	DQN	0.001	128	1001	T1	15	40	0.5	140	10000

Table B.1: Training parameters of the main models

Bibliography

[1] Aslak W Bergersen et al. "turtleFSI: A robust and monolithic FEniCS-based fluid-structure interaction solver". In: *Journal of Open Source Software* 5.50 (2020), p. 2089.

[2] Paul Garnier et al. "A review on deep reinforcement learning for fluid mechanics". In: *Computers & Fluids* 225 (2021), p. 104973.

[3] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: `http://jmlr.org/papers/v22/20-1364.html`.

[4] Nelvin Chummar Vincent et al. "Impact of artificial intelligence in the aviation and space sector". In: *Artificial Intelligence*. CRC Press, 2021, pp. 209–229.

[5] Sunil Kr Jha et al. "Renewable energy: Present research and future scope of Artificial Intelligence". In: *Renewable and Sustainable Energy Reviews* 77 (2017), pp. 297–317.

[6] Lerrel Pinto et al. "Asymmetric actor critic for image-based robot learning". In: *arXiv preprint arXiv:1710.06542* (2017).

[7] Dzmitry Bahdanau et al. "An actor-critic algorithm for sequence prediction". In: *arXiv preprint arXiv:1607.07086* (2016).

[8] David Silver et al. "Mastering the game of go without human knowledge". In: *nature* 550.7676 (2017), pp. 354–359.

[9] Amanda Lampton, Adam Niksch, and John Valasek. "Morphing airfoils with four morphing parameters". In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. 2008, p. 7282.

[10] Feng Ren, Jean Rabault, and Hui Tang. "Applying deep reinforcement learning to active flow control in weakly turbulent conditions". In: *Physics of Fluids* 33.3 (2021).

[11] Yu Zhou et al. "Artificial intelligence control of a turbulent jet". In: *Journal of Fluid Mechanics* 897 (2020), A27.

[12] Feng Ren, Hai-bao Hu, and Hui Tang. "Active flow control using machine learning: A brief review". In: *Journal of Hydrodynamics* 32 (2020), pp. 247–253.

[13] RS Matos et al. "Three-dimensional optimization of staggered finned circular and elliptic tubes in forced convection". In: *International Journal of Thermal Sciences* 43.5 (2004), pp. 477–487.

[14] Frédérique Muyl, Laurent Dumas, and Vincent Herbert. "Hybrid method for aerodynamic shape optimization in automotive industry". In: *Computers & Fluids* 33.5-6 (2004), pp. 849–858.

[15] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[16] Fangfang Xie et al. “Deep Reinforcement Learning: A New Beacon for Intelligent Active Flow Control”. In: *Aerospace Research Communications* 1 (2023), p. 11130.

[17] Jean Rabault et al. “Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control”. In: *Journal of fluid mechanics* 865 (2019), pp. 281–302.

[18] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[19] Mattia Gazzola et al. “Learning to school in the presence of hydrodynamic interactions”. In: *Journal of Fluid Mechanics* 789 (2016), pp. 726–749.

[20] Guido Novati et al. “Synchronisation through learning for two self-propelled swimmers”. In: *Bioinspiration & biomimetics* 12.3 (2017), p. 036001.

[21] Athanasios Emmanouil Giannenas, Sylvain Laizet, and Georgios Rigas. “Harmonic forcing of a laminar bluff body wake with rear pitching flaps”. In: *Journal of Fluid Mechanics* 945 (2022), A5.

[22] Nirmal J Nair and Andres Goza. “Bio-inspired variable-stiffness flaps for hybrid flow control, tuned via reinforcement learning”. In: *Journal of Fluid Mechanics* 956 (2023), R4.

[23] Chengfang Duan and Aimy Wissa. “Covert-inspired flaps for lift enhancement and stall mitigation”. In: *Bioinspiration & biomimetics* 16.4 (2021), p. 046020.

[24] Shun C Yen and Chen W Yang. “Flow patterns and vortex shedding behavior behind a square cylinder”. In: *Journal of Wind Engineering and Industrial Aerodynamics* 99.8 (2011), pp. 868–878.

[25] Sebastian Gjertsen. “Development of a verified and validated computational framework for fluid-structure interaction: Investigating lifting operators and numerical stability”. MA thesis. 2017.

[26] Andreas Strøm Slyngstad. “Verification and Validation of a Monolithic Fluid-Structure Interaction Solver in FEniCS. A comparison of mesh lifting operators.” MA thesis. 2017.

[27] Hongyi Jiang and Liang Cheng. “Hydrodynamic characteristics of flow past a square cylinder at moderate Reynolds numbers”. In: *Physics of Fluids* 30.10 (2018), p. 104107.

[28] Qian Mao et al. “Drag reduction by a rotationally oscillating cylinder with a flexible filament”. In: *Physics of Fluids* 34.4 (2022).

[29] Qian Mao, Yingzheng Liu, and Hyung Jin Sung. “Drag reduction by flapping a pair of flexible filaments behind a cylinder”. In: *Physics of Fluids* 35.3 (2023).

[30] Vishnu Vijayan. *Deep Reinforcement Learning: Artificial Intelligence, Machine Learning, and Deep Learning*. URL: `https://medium.com/@vishnuvijayanpv/deep-reinforcement-learning-artificial-intelligence-machine-learning-and-deep-learning-e52cb5974420`.

[31] Richard Bellman. “A Markovian decision process”. In: *Journal of mathematics and mechanics* (1957), pp. 679–684.

[32] Amirhosein Mosavi et al. “Comprehensive review of deep reinforcement learning methods and applications in economics”. In: *Mathematics* 8.10 (2020), p. 1640.

[33] Volodymyr Mnih et al. “Playing atari with deep reinforcement learning”. In: *arXiv preprint arXiv:1312.5602* (2013).

[34] Mark Towers et al. *Gymnasium*. URL: `https://github.com/Farama-Foundation/Gymnasium`.