Studienarbeit Nr. 794

Active flow control in simulations of fluid flows based on deep reinforcement learning

Darshan Thummar

Issued by: Prof. Dr.-Ing. R. Radespiel
Institut für Strömungsmechanik
Head of Institute Prof: Prof. Dr.-Ing. R. Radespiel
Technische Universität Braunschweig

Supervisor: Dr.-Ing. Andre Weiner (TU Braunschweig)

Publication: 05.2021

Statutory Declaration in Lieu of an Oath

Hereby I, Darshan Thummar, declare that I have authored this Studienarbeit independently and without any external help and that I have not used any aids other than those indicated.

Braunschweig, 17.05.2021

Abstract

For flow across the body, the vortices form in the wake due to unsteady separation of the flow. These vortices propagate periodically in the wake called von Kármán vortex street. The von Kármán vortex street exerts oscillatory forces on the body in terms of drag and lift. Such forces may cause aerodynamic losses and vibration in the structure. This unsteadiness may lead to heavy structural damage in the case of resonance. Hence it is requisite and goal of the project to control the unsteadiness. This von Kármán vortex street can be observed for flow across airfoil and other complex structures. Hence, controlling pressure fluctuations for complex structures introduces a multi-dimensional problem. For simplicity, flow across a 2D cylinder is taken into consideration as a generic problem which is also referred to as a bluff body.

In active flow control, the cylinder is supplied external energy to control the pressure fluctuation. The external energy to the cylinder is provided by rotating the cylinder in oscillatory motion. Active flow control may be classified as open-loop control and closed-loop control. In open-loop control, the cylinder is oscillatory rotated with a mathematical function as an Ansatz, which minimizes the pressure fluctuation. The optimal parameters of mathematical function are determined by parametric study to mitigate the drag and lift forces. On other hand, in closed-loop control, the system takes feedback to evaluate the applied rotation. In this project, the closed-loop control is achieved by the proximal policy algorithm (PPO) through deep reinforcement learning (DRL). For the PPO algorithm, the feedback from the system is taken as the cost function or in deep reinforcement learning, the reward function. The cost function is formulated with the parameter such as drag, lift, and rotational rate. The parameter of the cost function is weighted by initial weights to find an optimal control strategy. PPO algorithm is classified under Actor-Critic methods, in which the actor and critic are neural networks responsible for taking action and evaluating the taken action, respectively. After the evaluation of taken action by the cost function, the actor and critic networks are updated to achieve an optimal control strategy. In this study, the flow control is achieved by open-loop control and closed-loop control. In closed-loop control, the pressure sensors are placed on the surface of the cylinder and the actuation is subjected to the rotation of the cylinder. By using the open-loop control strategy 4.61% drag reduction is observed with increasing the amplitude and frequency of fluctuation of drag. However, by using the closed-loop control strategy 5.10% drag reduction is observed with a significant reduction in the amplitude and frequency of fluctuation of drag.

Contents

Nomenclature

Latin Symbols

A	Amplitude
a_t	Action for control time step t
c_D	Coefficient of drag
c_L	Coefficient of lift
C_{max}	Maximum courant number
$\bar{c_D}$	Mean value of drag
$\bar{c_L}$	Mean value of lift
c_{Dmax}	Minimum value of drag
c_{Lmax}	Minimum value of lift
c_{Dmin}	Maximum value of drag
c_{Lmin}	Maximum value of lift
d	Diameter of cylinder
F_D	Drag force
F_L	Lift force
f_{c_D}	Frequency of drag
f_{c_L}	Frequency of lift
G_τ	Return of the trajectory
H	Height of the domain
N_x	Number of cells in x-direction
N_y	Number of cells in y-direction
p	Pressure
Re	Reynolds number
R_t	Rewards
r_t	Reward for control time step t
S_f	Strouhal number
s_t	State observation for control time step t
r	Radius of diameter
t	Time
$\bar{U}$	Mean inlet velocity
$\mathbf{u}$	Velocity vector
u_∞	Free stream velocity
u	Velocity in x direction
v	Velocity in y direction
x, y	Cartesian coordinates
$\tilde{x}$	Normalized distance
V_ϕ	Policy network with parameter ϕ
$\hat{A}_t$	Advantage estimation

Greek Symbols

∇	Nabla operator
ν	Kinematic viscosity
ρ	Density
ω	Angular velocity
Ω	Normalized amplitude
Ω_{amx}	Peak rotational rate
Φ	Objective function for open-loop control
γ	Discount factor
τ	Trajectory length
$\dot{\theta}$	Rotational rate
μ_a	Mean of the action
σ_a	Standard deviation of the action
Δt	Time step
Δt_c	Control time step
π_θ	Policy network with parameter θ
$\mathcal{D}_k$	Number of trajectories
δ_t^V	Action-advantage function

Abbreviations

CFD	Computational Fluid Dynamics
DRL	Deep Reinforcement Learning
FVM	Finite Volume Method
KRR	Kernal Ridge Regression
NN	Neural Network
PDE	Partial Differential Equation
PPO	Proximal Policy Optimization
RL	Reinforcement Learning

Chapter 1

Introduction

In nature, birds and insects are evolved to fly very efficiently in a highly complex turbulent environment. They can precisely and efficiently maneuver, glide, land, and take off by controlling the flow across their wings. Also in a marine environment, fishes swim efficiently in the water by controlling the flow across the fins. Inspiring by the evolution, the shape of the wing of aircraft is replicated from the shape of the wing of a bird. Birds and insects change their shape of the wing to fly easily in the fluid environment or in other words, they change the shape of the wing in order to control the flow across the wings. Therefore, for real-life applications for aircraft, automobiles, etc., flow control is very crucial in order to improve the efficiency of control over fluid-induced forces. For an instance, in aircraft, the shape of the wing is changed for better efficiency with mainly changing the shape of the wing by flaps, spoiler, and slats. Throughout time, many shape optimization methods and algorithms show the improvement in control over fluid-induced forces like drag and lift [3]. But due to stagnation of scope in shape optimization, it is crucial to control flow by active flow control or passive flow control [13, 23, 6, 25].

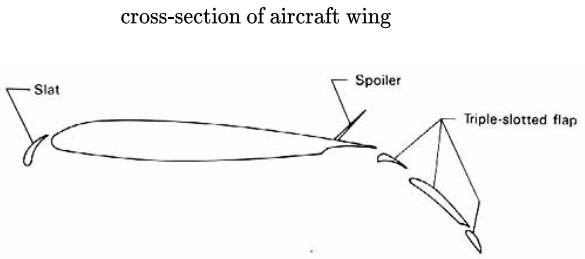

Figure 1.1: Flight of bird in complex fluid environment by changing the shape of wing to control the flow; Cross-section of aircraft wing.

Passive flow control may be achieved using vortex generators, surface smoother or porous layers between the solid body and the fluid in order to control fluid-induced forces [2, 25, 6]. For instance, riblets may be used on the surface of the wing and aircraft body in order to reduce viscous drag [25]. This is again mimic of fish skin or feather-like structures. However, for active flow control, the external energy is supplied to the system in order to control fluid-induced forces. Boundary layer control by the blowing and suction slot for an aircraft to reduce the drag is an example of active flow control [27]. The active flow control is classified as open-loop control and closed-loop control.

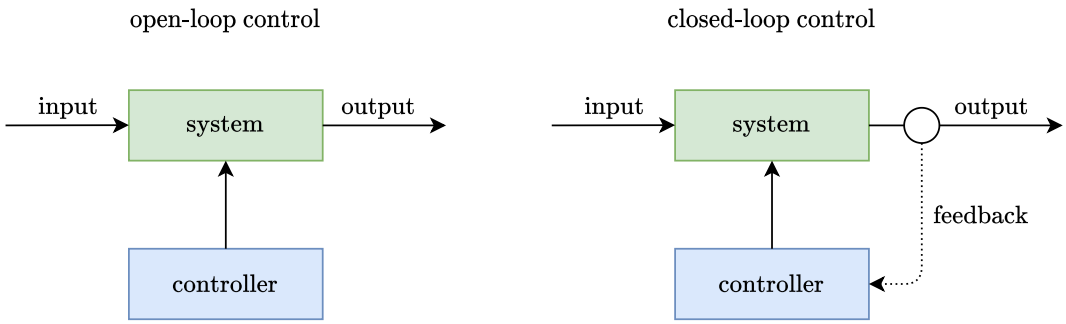

Figure 1.2: Block diagram for open-loop control and closed-loop control.

In open-loop control, the controller does not receive feedback from the system as shown in figure 1.2. In open-loop control, the control strategy is carried out and evaluated initially, and then it is fit in the system to achieve control. Therefore, the open-loop control strategy is also called offline learning.

In closed-loop flow control, the feedback from the system is utilized in order to evaluate the control strategy [4]. The feedback is frequently analyzed to improve the control. Therefore, it is also called online learning. For closed-loop control, this feedback is used to evaluate the control strategy and further, it is used to improve the control strategy during a running simulation [1].

For closed-loop learning, reinforcement learning (RL) is a proven state-of-the-art method. Reinforcement learning is one of the three categories along with supervised learning and unsupervised learning. Reinforcement learning is self-organized learning, where learning is achieved by experience and the outcome of the experience [8]. Meaningly, reinforcement learning collects data itself and learns from it. Reinforcement learning is a mimic of the human brain. In reinforcement learning, the fluid flow setting is called environment. The external energy supplied to the system is referred to as an action in reinforcement learning. The controller is referred to as an agent. The feedback from the system is obtained from the environment is formulated as a reward function which is an objective function of flow control. The reward function is utilized to evaluate the actuation. In reinforcement learning, the goal is to maximize the reward function by evaluating it and improving the actuation during the running of the system [15]. In a nutshell, for reinforcement learning the agent supplies an actuation to the environment according to the state of the system in order to maximize the returns or cumulative rewards.

In reinforcement learning, the improvement of the actuation introduced a mathematical problem for an agent (controller), where the coupling between the rewards in terms of fluid-induced forces and the actuation is not known. Hence, to approximate the coupling, the agent is replaced by neural network (NN) from deep learning, thus it is called deep reinforcement learning (DRL) [18, 20].

In fluid mechanics, when fluid flows across the object, shear stress generates between the wall of the object and the surrounding fluid. This shear stress causes the boundary layer to detach. Because of the detachment of the boundary layer and the vorticity generated by the object, vortices form in the wake. The vortices propagate periodically in the wake. These periodically propagating vortices are called von Kármán vortices [26]. The shear stress between the boundary layer and wall, and the pressure gradient due to the surface curvature of an object exerts forces on the body in form of drag and lift. The typical vortex propagation in the wake exerts periodic forces on the body in terms of lift and drag, which could potentially cause aerodynamic loss and vibration in the structure [24]. The vibration could also lead to complete structural failure by the coupling of periodicity or frequency of vortices in the wake and natural frequency of the structure [26]. Hence, it is essential to control the forces acting on the body and its periodicity in

order to mitigate the aerodynamic losses and to ensure the safety of the structure. The control plays a critical role for the structures, especially aircraft and automobiles [12]. Therefore, it is crucial to control the unsteady separation of flow by active flow control or passive flow control. But the control strategy for flow control is not yet fully understood for the complex structure like aircraft, hence as a generic example flow across 2D cylinder is considered in this study in order to evaluate the control strategy.

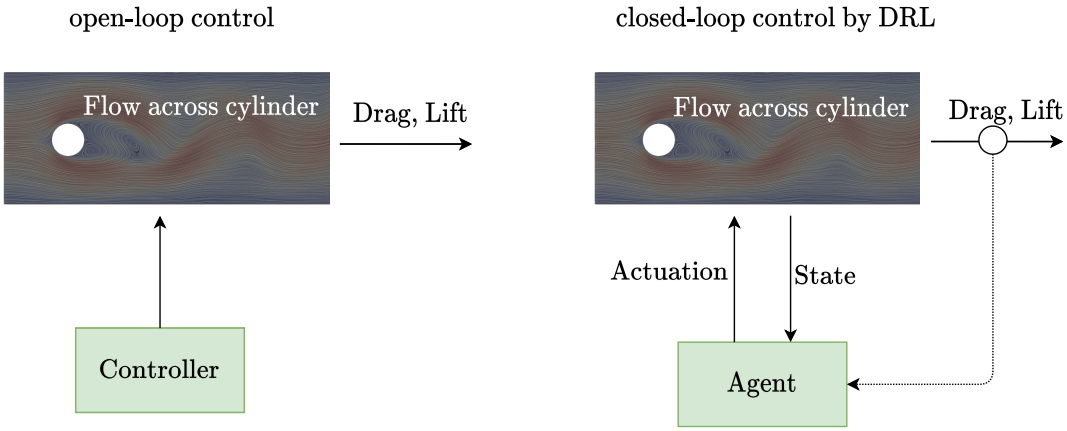

Figure 1.3: Block diagram for open-loop control and closed-loop control by DRL for flow across the cylinder.

To control the periodic drag and lift forces by an open-loop control, the cylinder is subjected to oscillatory rotation in this study. There is no direct mathematical function to carry out the optimal parameter of the oscillatory rotation to reduce drag, lift, and fluctuation of drag and lift. Therefore, the oscillatory rotation is provided with the sine wave function for angular velocity as the drag and lift fluctuate with the sine-like wave function. The parameters of oscillatory rotation are determined by parametric study for the control strategy.

In 1990 P. T. Tokumaru and P. E. Dimotakis presented the experimental study for open-loop control for flow across the cylinder [24]. In the study, the external energy to the cylinder is supplied as oscillatory rotation, flow is subjected to Reynolds number $Re = 1.5 \times 10^4$. The inflow boundary condition is formed by the GALCIT water channel in the study. The oscillatory rotation is defined with the sine wave function in the study. The optimal parameter for the oscillatory rotation is achieved by parametric study. The study presents the effect of cylinder rotation for different parameters on drag. In the study, drag reduction is achieved by factor of six, where the cylinder is rotated with optimal parameter for the oscillatory rotation.

For flow around a cylinder, in this study, DRL is considered as a closed-loop strategy. In closed-loop control for flow across the cylinder, the cylinder is subjected to rotary motion for this study, where the rotation in terms of angular velocity is determined by the agent. The agent supplies the actuation to the environment by obtaining the feedback from the system as drag and lift value based on the state of the environment. The state of the system is considered as the pressure values from the cylinder surface, which is also referred to as state observations. In recent studies, the pressure sensors are mounted in the wake of the cylinder [13, 23, 11]. However, in this study, the pressure values are obtained from the surface of the cylinder, which is more intuitive considering the real-life applications, thus a novelty of this study. In this study, the proximal policy optimization (PPO) algorithm is considered for DRL to achieve flow control.

The PPO algorithm is a modern and more effective algorithm for DRL [13, 23, 11]. For the PPO algorithm, the agent has two neural networks [20]. The one neural network approximates the coupling between rewards and the actuation, where rewards are referred to as drag and lift. This neural network supplies the actuation to the environment by feeding the state observations. This

neural network is called a policy network or actor [8]. The second neural network approximates the coupling between state observations and rewards. This neural network evaluates how valuable the supplied actuation is, hence this neural network is called value network or critic [8]. Thus, PPO is also called an actor-critic method. Flow control in DRL using the PPO algorithm is achieved by optimal policy network, which supplies an action based on state observation such that the drag and lift are minimal. The optimal policy network is determined by PPO iterations [20].

In 2019, Jean Rabault et al. presented a study for flow control by DRL [14]. In the study, the external energy to the cylinder is supplied as two air-jets mounted on the top and bottom of the cylinder surface. The pressure sensors are placed in the wake of the cylinder, where the flow is subjected to Reynolds number $Re = 100$. For closed-loop control by DRL, the PPO algorithm is used in the study. By using PPO, 8% drag reduction is observed in the study.

In 2020 Mikhail Tokarev et al. presented a study for DRL flow control by using cylinder rotation as an actuation [23]. In the study, pressure sensors are also placed in the wake of flow. For DRL flow control, the PPO algorithm is considered, where flow is subjected to Reynolds number $Re = 100$. In the study, the PPO algorithm is used for two different reward functions, where depending on the reward function, 14% and 16% drag reduction is achieved with negligible fluctuation in drag force.

In 2020 Romain Paris et al. presented a study with suction and blowing air at the top and bottom of the cylinder surface in order to control the flow [11]. In the study, a modified PPO algorithm (S-PPO-CMA) is considered for the flow with Reynolds number $Re = 120$. In this study, 18.4% drag reduction is observed. The pressure sensors are also placed in the wake of flow in the study but the study presented the performance of the DRL control strategy for the different number of pressure sensors in the wake and also the effect on the performance of the DRL by placing pressure sensors at a different distance from the cylinder in the wake.

Moreover, In 2020, Tang et al. presented a study for flow control by DRL for different values of Reynolds number by using PPO algorithm [22]. The pressure sensors are placed in the wake in this study. Four synthetic jets symmetrically located on the upper and lower sides of a cylinder are used as actuation in the study. in the study, 5.7%, 21%, 32.7%, and 38.7% drag reduction is obtained for flow with Reynolds number 100, 200, 300, and 400.

In this study, we attempt to achieve flow control by using an open-loop control strategy and a closed-loop control strategy. The implemetation and solution of the open-loop control strategy and closed-loop control strategy is obteined by using OpenFOAM as a CFD-solver. The primary objective of this study is to reduce drag force and its fluctuation for flow across the cylinder as it is higher in magnitude than lift and its fluctuation by considering flow with Reynolds number 100. The key objective of this study includes,

- Creating a 2D base flow simulation in OpenFOAM and comparing the outcome to available results from the literature, e.g. for drag coefficient and vortex shredding frequency.
- Defining and describing open-loop control strategy, actuation, and analyzing the performance of optimal open-loop control strategy.
- Defining and describing closed-loop control strategy, environment's state, reward signal, and agent's actuation.
- Implementation of PPO algorithm for multi-agent.
- Analyzing the discussing the agent's performance, sensitivity, and cost with respect to the available observations, the control strategy, and the reward signal.
- assessing the agent's ability to generalize for different flow conditions.

Chapter 2

Theoretical Foundations

2.1 Flow Problem

The Finite Volume Method (FVM) is widely used today in computational fluid dynamics to evaluate partial differential equations in the form of algebraic equations. The FVM involves the approximation of finite volume integrals. To perform FVM, the domain is discretized into finite volumes. In that small finite volume, the values of properties are averaged and these values are available at the center of the respective finite volume. These values at the centroids of finite volume are interpolated in order to calculate the flux across the control volume's surface. In other words, the value of the fluid property at the cell center represents the averaged value of that specific property within the cell. Hence FVM steps mainly involves:

- definition of the mathematical problem (PDE, initial and boundary conditions)
- discretization of domain (meshing) and equations (approximation of surface and volume integrals)
- iterative solution (linear systems of equations, pressure-velocity coupling)

2.1.1 Formulation of the Flow Problem

In a high-dimensional flow problem, for a better understanding of the effect of drag and lift forces and simplicity, the generic fluid flow problem as flow across 2D cylinder is considered. For the generic flow problem, flow around the cylinder with a circular cross-section is selected. The test case is taken from benchmark computations of laminar flow around a cylinder by Schäfer and Turek [17] as a reference. The first case from Schäfer and Turek is selected, where the domain is two dimensional and flow is incompressible for simplicity yet a very good representation of generic fluid flow problem. In the flow problem, the flow characteristic properties such as drag forces and lift are also calculated to determine system behavior.

Geometry

In the following, all distances are normalized using the cylinder diameter d.

$$\tilde{x} = \frac{x}{d} \tag{2.1}$$

The domain length and width is 22 and 4.1 respectively . The cross-section of the cylinder is located 2 far from the inlet and slightly off-center from top 1.6 and bottom side 1.5. The detailed sketch of geometry is illustrated in figure 2.1. The flow is coming from inlet, left side of figure 2.1

and exits in open environment from outlet, right side of figure 2.1. The cylinder is fixed at its center, hence the distance of the cylinder from the domains remains constant.

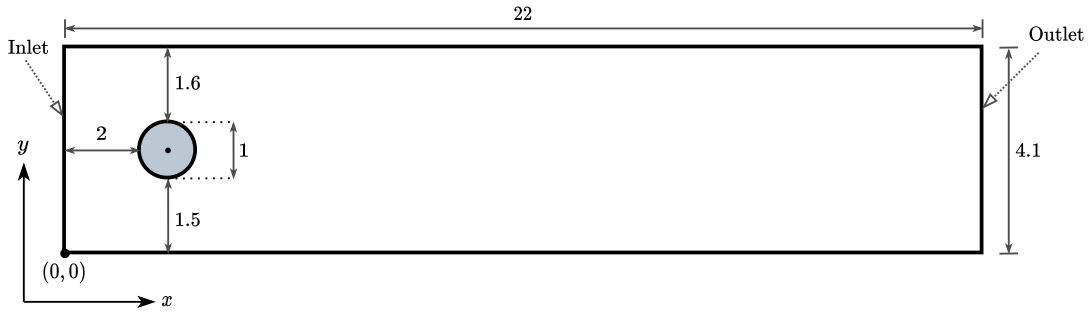

Figure 2.1: Physical domain of the simulation setup; all distances are normalized with the cylinder diameter.

Fluid Properties

The fluid flow is considered as an incompressible, unsteady, and Newtonian fluid flow, which is governed by mass conservation equation and non-dimensional momentum conservation equation. To solve this fluid flow problem, OpenFOAM is used as a CFD solver tool. The non-dimensional momentum conservation equation(Navier-Stokes momentum equation) is scaled by,

$$t^* = \frac{t}{d\mathrm{u}_\infty} \tag{2.2}$$

$$\mathbf{u}^* = \frac{\mathbf{u}}{\mathrm{u}_\infty} \tag{2.3}$$

$$\nabla^* = d\nabla \tag{2.4}$$

$$p^* = \frac{pd}{\mu \mathrm{u}_\infty} \tag{2.5}$$

Where, $\mathbf{u}^*$, p^* and ∇^* are dimensionless properties of $\mathbf{u}$, P and ∇ respectively [16], time t^* is non dimensional time of t, and u_∞ is free stream velocity. The velocity $\mathbf{u}^*$ is a non dimensional velocity vector $\mathbf{u} = (u, v)$. where u and v is a velocity in respective Cartesian coordinates x and y. Pressure p^* is non dimensionalized by taking viscous flow into consideration. The mass conservation equation and the non-dimensional Navier-Stokes momentum equation is formulated by using scaled quantities as,

$$\nabla \cdot \mathbf{u}^* = 0 \tag{2.6}$$

$$\frac{\partial \mathbf{u}^*}{\partial t} + (\mathbf{u}^* \cdot \nabla^*)\,\mathbf{u}^* = -\nabla^* p^* + \frac{1}{Re}\nabla^{*2}\mathbf{u}^* \tag{2.7}$$

The fluid flows with Reynolds number $Re = 100$ which can be calculate from the equation 2.8.

$$Re = \frac{\mathrm{u}_\infty d}{\nu} \tag{2.8}$$

Where u_∞ is a free stream velocity, d is a diameter of cylinder and ν is kinematic viscosity, where $\nu = 1.0 \times 10^{-3}\mathrm{m}^2/\mathrm{s}$.

In this flow problem, the shear stress at the wall of the cylinder cause the drag force on the cylinder in x-direction, and the pressure gradient causes the lift force on the cylinder in y direction. The drag force and lift force are calculated as shown in equation 2.9 and 2.10. From the drag and lift forces, the coefficient of lift and coefficient of drag is determined as shown in equation 2.11 and 2.12 [17].

$$F_D = F_x = \int_S \left(\rho\nu \frac{\delta v_t}{\delta n} n_y - p n_x \right) \tag{2.9}$$

$$F_L = F_y = \int_S \left(\rho\nu \frac{\delta v_t}{\delta n} n_x - p n_y \right) \tag{2.10}$$

$$c_D = \frac{2F_D}{\rho \bar{U}^2 d} \tag{2.11}$$

$$c_L = \frac{2F_L}{\rho \bar{U}^2 d} \tag{2.12}$$

$$\tag{2.13}$$

Where, F_D and F_L are referred as drag and lift force, which are integrated over surface S. The F_x and F_y are referred as forces acting on the cylinder in x and y axes. The c_D and c_L is denoted as drag and lift coefficients. The $t = (n_y, -n_x)$ represents tangent vector, where n is normal vector on S with x-component n_x and y-component n_y. The tangential velocity is denoted as v_t. The ρ is referred as density of the fluid and $\bar{U}$ is mean inlet velocity.

Modeling of Geometry and Flow Domain

The body about which flow is to be analyzed requires modeling. This generally involves modeling the geometry with a CAD software package.[1] As OpenFOAM is used as a tool, modeling of geometry and flow domain of the generic test case of flow across cylinder can be easily implemented within OpenFOAM [9].

Establishing the Boundary and Initial Conditions

In the test case, at the wall of the domain and the surface of the cylinder no-slip boundary condition implemented as shown in figure 2.2. The inflow is entering into the domain with parabolic inflow condition and at the outlet, a zero gradient flow condition is implemented.

At wall and cylinder surface : no slip,

$$u = v = 0 \tag{2.14}$$

$$\frac{\mathrm{d}p^*}{\mathrm{d}\mathbf{x}} = \mathbf{0}_{2,1} \tag{2.15}$$

Where, u and v are the velocity in x-direction and y-direction respectively, p^* is the pressure(see equation 2.5), $\mathbf{x}$ is a positional vector $\mathbf{x}(x, y)$, and $\mathbf{0}_{2,1}$ is 2×1 zero vector.

[1] `https://www.grc.nasa.gov/WWW/wind/valid/tutorial/process.html`

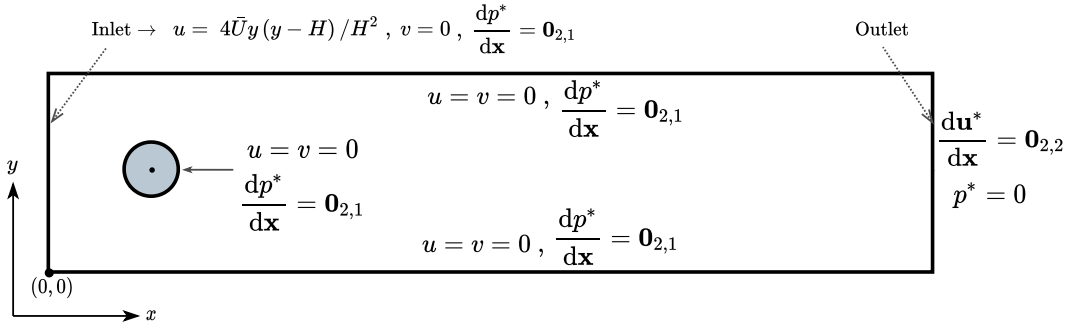

Figure 2.2: Boundary condition at domain of the flow and at the cylinder surface.

Outflow condition: zero gradient,

$$\frac{\mathrm{d}\mathbf{u}^*}{\mathrm{d}\mathbf{x}} = \mathbf{0}_{2,2} \tag{2.16}$$

$$p^* = 0 \tag{2.17}$$

Where, $\mathbf{u}^*$ is velocity vector(see equation 2.3) and $\mathbf{0}_{2,2}$ is 2×2 zero matrix.

Inflow condition: Parabolic inlet condition,

$$u = 4\bar{U}y\,(y-H)\,/H^2$$

$$v = 0 \tag{2.18}$$

$$\frac{\mathrm{d}p^*}{\mathrm{d}\mathbf{x}} = \mathbf{0}_{2,1} \tag{2.19}$$

Where, u_m is a mean velocity and H is a length the domain in y-direction. In the inlet boundary condition $\bar{U}$ is set to 1.5m/s, which leads the flow with $Re = 100$.

2.1.2 Mesh generation

The mesh generation of the domain is performed in OpenFOAM by using `blockMesh` and `snappyHexMesh` [9]. The background mesh is generated by the `blockMesh`. The background mesh contains hexahedral cells with the same size. This mesh is refined by `regionRefinement` in one third of domain from left in order to capture accurate flow dynamics. This region refinement is achieved by `snappyHexMesh`. After the region refinement, the cells in cylinder is extracted by `snappyHexmesh`, which is called snapping to surface in `snappyHexMesh`. The mesh output from the snapping stage may be suitable for the purpose, although it can produce some irregular cells along boundary surfaces[2]. Therefore, the additional layers of hexahedral cells aligned to the cylinder surface are generated around the cylinder by `snappyHexmesh` as illustrated in Figure 2.3.

The domain near to cylinder is fine-meshed, as the separation of flow occurs in this region and thus it is fine-meshed, in order to account for accurate fluid properties. At the domain near the cylinder, the thickness of the cell is half as the thickness of the cell in the wake. In OpenFOAM this mesh settings can be easily achieved by changing respective parameter of the `snappyHexMeshDict` files by `refinementRegions` [9]. As, the region refinement and the mesh around cylinder depends on background mesh, change in background mesh as fine or coarse results

[2]https://cfd.direct/openfoam/user-guide/v6-snappyhexmesh/

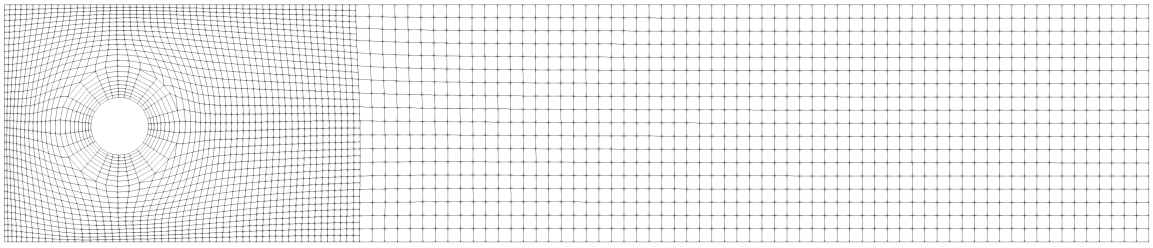

Figure 2.3: Descretized Computational domain.

also fine or coarse mesh in a region where cylinder is placed. Number of cells in x-direction and y-direction can be calculated for background mesh as shown in equation 2.20.

$$\Delta x = \frac{L_x}{N_x} \ , \ \Delta y = \frac{L_y}{N_y} \tag{2.20}$$

Where Δx and Δy is the thickness of cell in x-direction and y-direction respectively. N_x and N_y is the number of cells in x-direction and y-direction respectively. L_x and L_y is the length of domain and width of domain for background mesh. The t_x and t_y is the same for the background mesh. In figure 2.3, value of N_x and N_y is 100 and 18 respectively. These values of N_x and N_y correspond to region without `regionRefinement` in `snappyHexMesh`. In `regionRefinement` the cell numbers would be doubled as the `regionRefinement` is set to 2.

Mesh dependency study

Intuitively, the value of the fluid property at the cell center represents the averaged value of that specific property within the cell. The average value at the centroid is assumed to be equal for the entire cell. Therefore, the very small size of the cell results in an accurate and true value of fluxes that are calculated across the faces of the cell. Hence, increasing the number of cells or in other words, increasing the nodal point density leads to a rigorous representation of a physical property thence an accurate approximation of partial differential equations. The increasing number of cells or increasing nodal point density is called mesh refinement. That makes mesh refinement very important for fluid simulations. However, refining the mesh results in high computational cost so it can be only refined up to the coveted computational cost. Moreover, after a certain point of mesh refinement, the change in the desire physical property is very small. Hence optimum mesh regarding computational power refinement is obligatory, notably in FVM. Estimating this optimum mesh refinement is called mesh dependency study and it is very crucial and inevitable in most of the CFD simulations. The mesh refinement is performed by refining background mesh by `blockMesh` as the refinement in background mesh also leads refinement around the cylinder.

In the mesh of domain, from figure 2.3, for `blockMesh` $\Delta x = \Delta y = \Delta$ such that the resulting number of cells in each direction is

$$N_x = \frac{L_x}{\Delta} \tag{2.21}$$

$$N_y = \frac{L_y}{\Delta} \tag{2.22}$$

Where $\Delta x = \Delta y = \Delta$ is a thickness of individual mesh cell. Here N_x and N_y can be formulated such that change in N_x leads to relative change in N_y. Hence, value of N_y is dependent of value

of N_x and changing value of N_x results in relative change in N_y. For Mesh refinement, value of N_x is varied quadratically to 100, 200, and 400. For different mesh refinements, the number of cells, number of faces and number of points are mentioned in table 2.1.

Mesh Size(N_x)	number of cells	number of points
100	5506	9359
200	21991	36565
400	85678	141063

Table 2.1: Number of cells, number of faces and number of points for different mesh refinements(N_x).

From table 2.1, increasing mesh size quadratically leads to quadratic increment of the number of cells, and number of points. From the table 2.1, refining mesh size leads to an increase in the number of the equation that is solved at the cell and fluxes across the cells. Which suggests the increase in computational power.

In fluid simulation, Courant-Friedrichs-Lewy (CFL) condition is necessary for stability and convergence of numerical scheme. The CFL condition is stated in equation2.23.

$$\frac{u\Delta x}{\Delta t} \leq C_{max} \tag{2.23}$$

Where, in equation 2.23, u is magnitude of velocity, Δx is characteristic cell length and Δt is time step. C_{max} refers to the maximum value of the courant number. For explicit scheme, the value of C_{max} is 1. As in our simulation setup implicit scheme used, the higher value of C_{max} results in less accurate simulation of flow problem. Hence, in the simulation setup, the maximum value of C_{max} is set to 1. Therefore, Δt is adjusted according to Δx for the specific mesh.

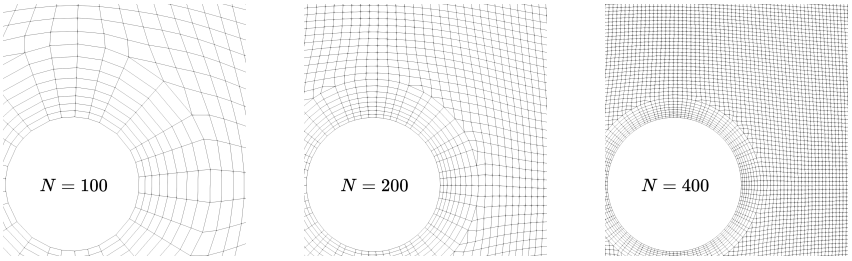

Figure 2.4: Enlarged view at the cylinder wall for mesh refinement at mesh refinement level ($N = N_x$) 100, 200 and 400.

The simulation is performed on different mesh sizes, in which the coefficient of drag(c_D) and coefficient of lift(c_L) is computed, which represent the vibrations on the body(cylinder). Computation of c_D and c_L is further described in equation 2.11 and 2.12.

In figure 2.5 and 2.6, values of coefficient of drag (c_D) and coefficient of lift c_L are plotted for different mesh size 100, 200, and 400 for time $t = 0$ to $t = 8$. From figure 2.5 and 2.6, values of c_D and c_L are converging along with mesh refinement. Further refinement leads to accurate values of c_D and c_L but the change of the values are 0.32% and 0.38% respectively with the value of c_D and c_L at mesh size 400. The relative change of c_L is higher as it deviates near to zero. For mesh size 100 in figure 2.5 and 2.6, it is clear that the results are not accurate for lift and drag due to numerical error caused by coarse mesh.

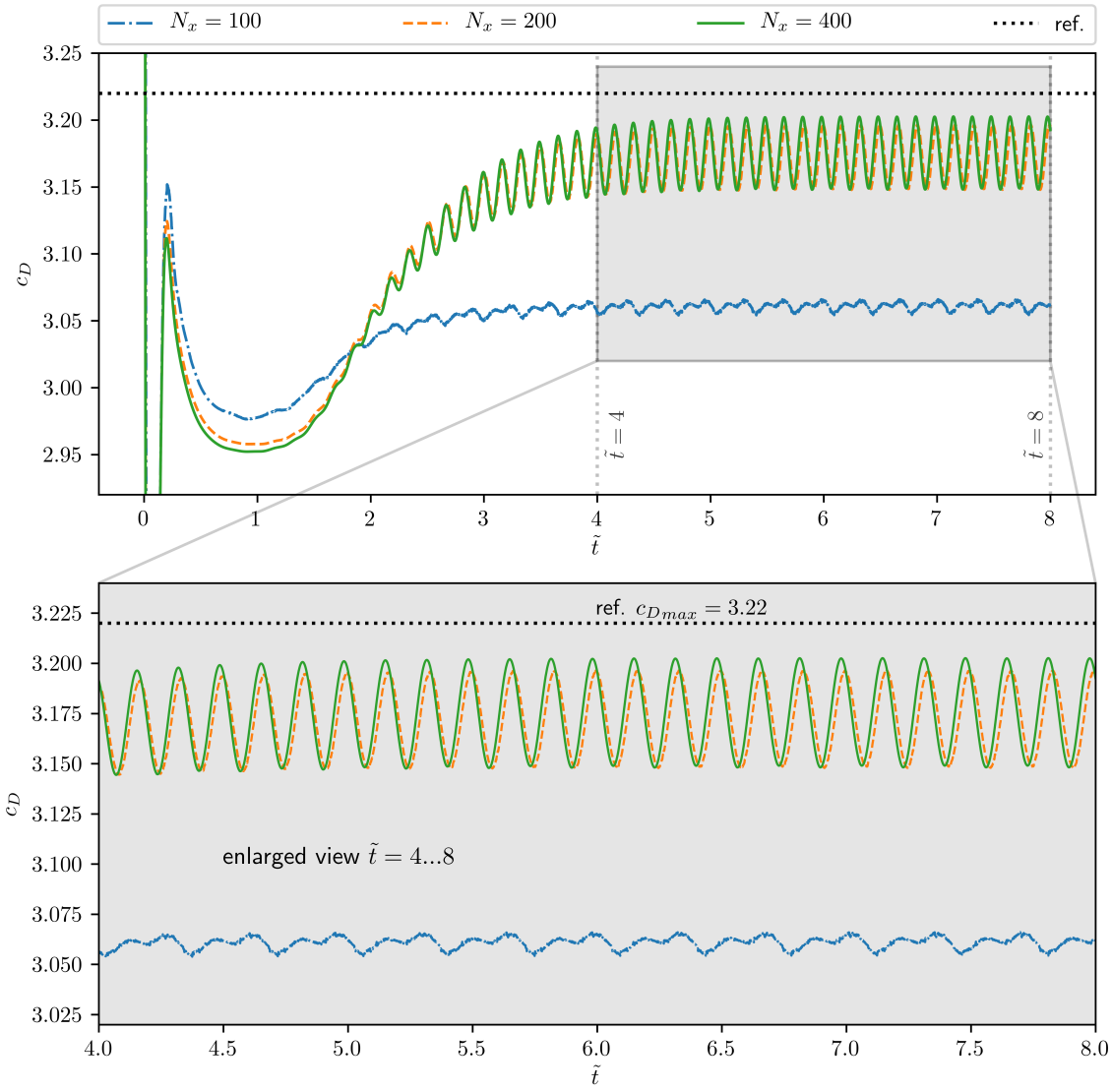

Figure 2.5: Values of c_D for different mesh refinements (N_x), where N_x varies to 100, 200 and 400.

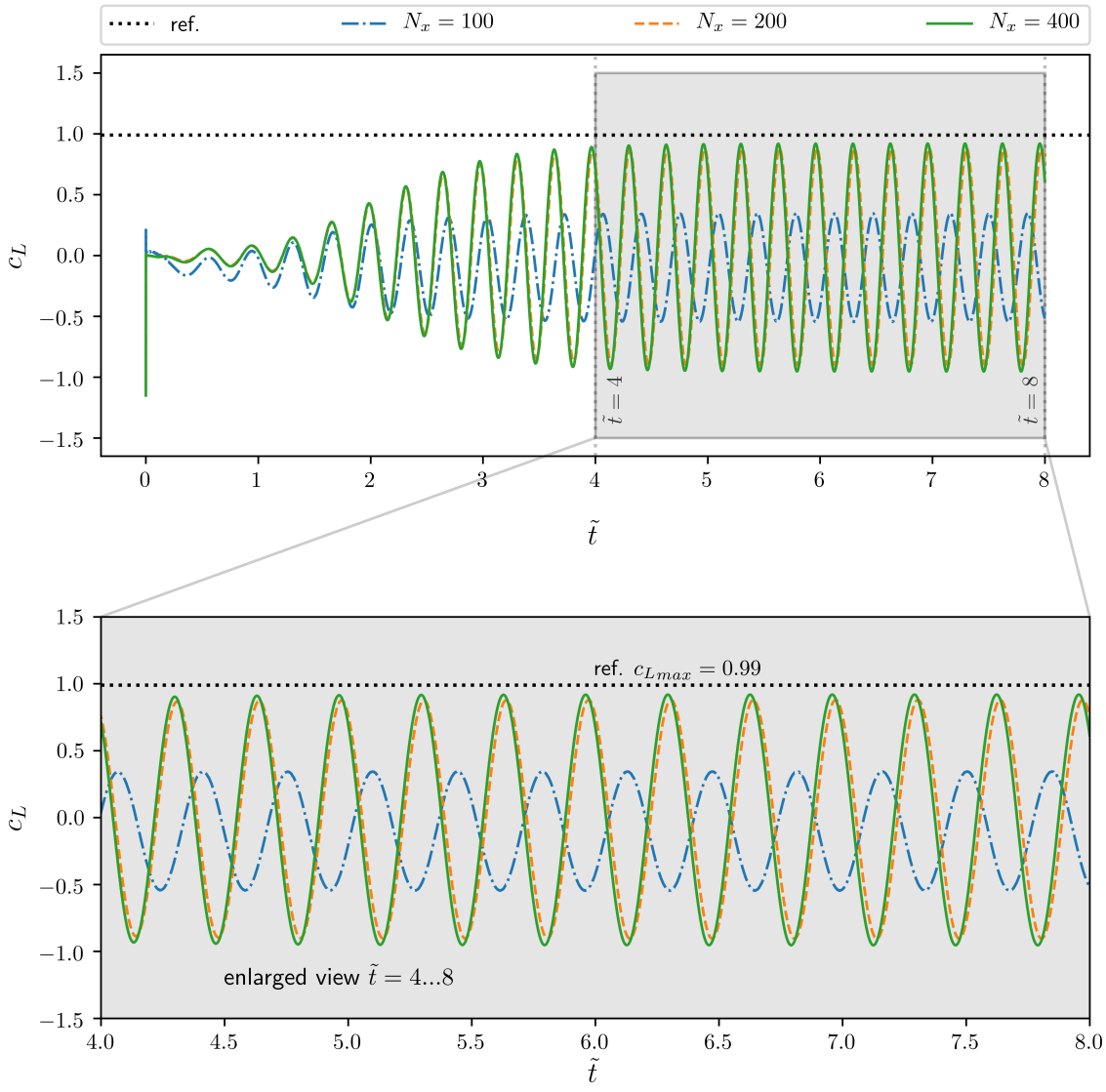

Figure 2.6: Values of c_L for different mesh refinements (N_x), where N_x varies to 100, 200 and 400.

Moreover, the mean values of drag and lift($\overline{c_D}$ and $\overline{c_L}$), maximum values of drag and lift(c_{Dmax} and c_{Lmax}) and minimum values of drag and lift(c_{Dmin} and c_{Lmin}) are converging towards the accurate mean value of drag and lift, which are shown in figure 2.7 and 2.8. The change in mean values, maximum values and minimum values for drag is 0.10%, 0.18%, and 0.006% respectively. However the change in mean values, maximum values and minimum values for lift is 23.01%, 4.31%, and 5.13% respectively, where the relative change for lift in mean value, maximum value, minimum value is slightly higher as it deviates near to zero and its lower magnitude for mesh size 200 and 400. This change can be reduced by refining mesh further.

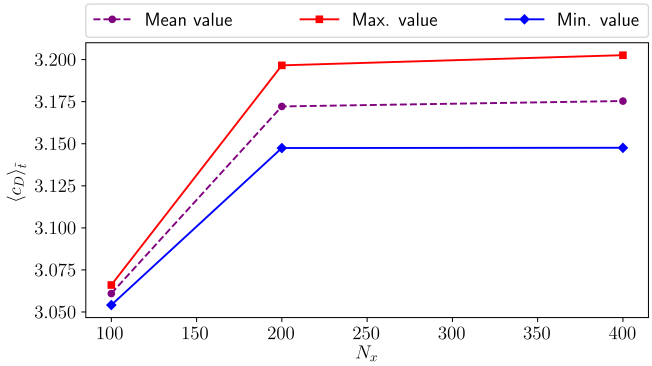

Figure 2.7: Mean value of c_D ($\overline{c_D}$), maximum value of c_D (c_{Dmax}) and minimum value of c_D (c_{Dmin}) for $t = 4s$ to $t = 8s$ with mesh size 100, 200 and 400.

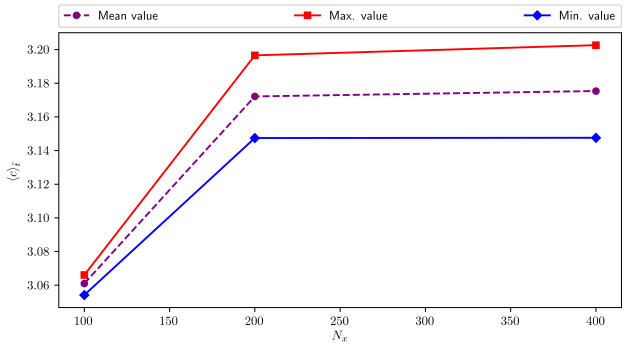

Figure 2.8: Mean value of c_L ($\overline{c_L}$) , maximum value of c_L (c_{Lmax}) and minimum value of c_L (c_{Lmin}) for $t = 4s$ to $t = 8s$ with mesh size 100, 200 and 400.

The frequency of drag force and lift force is obtained by sine curve fitting. By fitting sine curve, the frequency of the natural vortex shredding is determined. The frequency of the natural vortex shredding is equal to the frequency of the sine curve fitted on c_L. The sine curve fitting for the values of drag and lift also shows that change in frequency for mesh size 200 and mesh size 400 is 0.17%. Further refinement in mesh leads to the accurate representation of frequency but the change in frequency is very small. In figure 2.9 the sine curve fitting for drag and lift is illustrated with mesh size 200 and 400 respectively. The values of frequencies for drag and lift can be found in table 2.2.

Mesh Size(N_x)	frequency(f) of c_D	frequency(f) for c_L
200	6.0041	3.0022
400	6.0147	3.0074

Table 2.2: Frequencies for c_D and c_L for mesh size 200 and 400.

As refining mesh size, the computational time also increases. The computational time for the

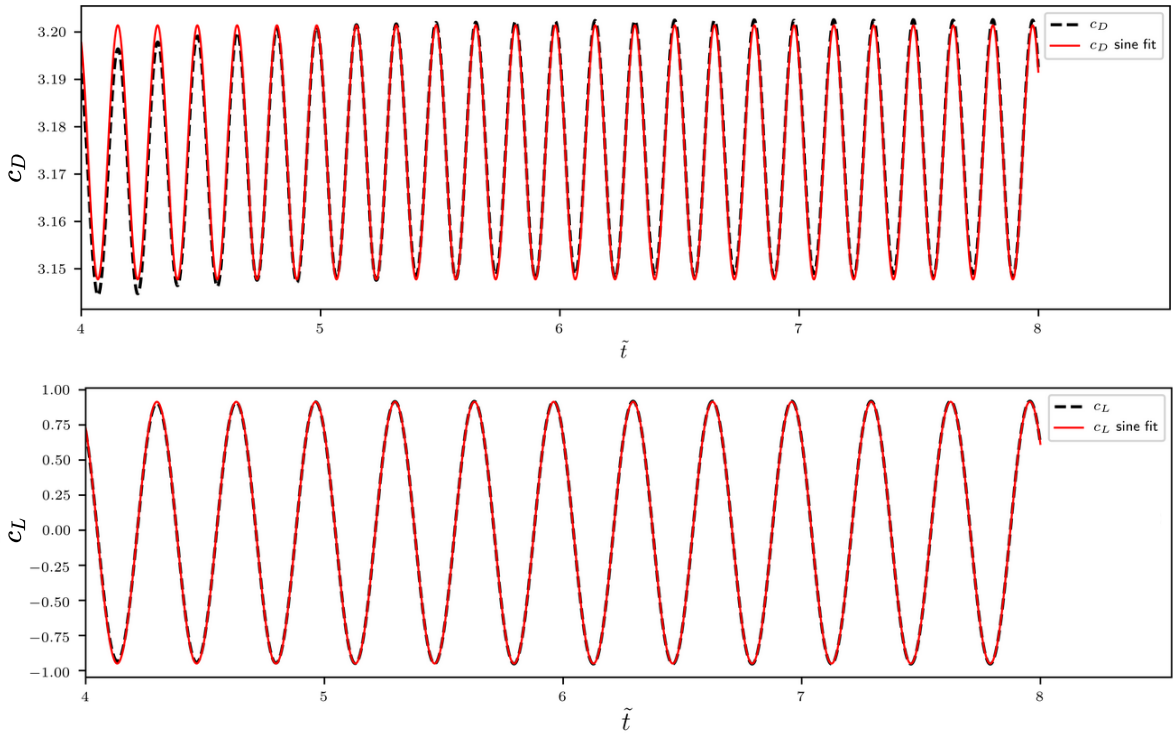

Figure 2.9: Sine curve fitting for c_D and c_L with mesh size N_x is 400.

Mesh Size(N_x)	Computational time(s)
100	12.48
200	28.95
400	325.52

Table 2.3: Computational time for different mesh sizes, where mesh sizes varies to 100, 200 and 400.

different mesh sizes, which are shown in table 2.3, is averaged for 3 test execution on specific mesh size settings on a local machine with the configuration of Intel i7-8550U CPU, 4.0GHz clock speed, and 8.0GB RAM. From the figure, it can be also seen that computation time also increases with the mesh refinement. Hence considering the plots of drag, lift and with the computational time for different mesh sizes, mesh size 200 is selected as an optimum mesh size.

2.1.3 Performing the Simulation

In FVM, the approximation of volume and surface integrals are carried out on the generated mesh. The fluxes across the cell and the fluid properties can be computed from the approximation of volume and surface integrals. Pressure(p) and velocity($\mathbf{u} = (u, v)$) can be also calculated by implementing `pimpleFoam` of pressure-velocity algorithm in OpenFOAM [10]. The results are shown in figure 2.10.

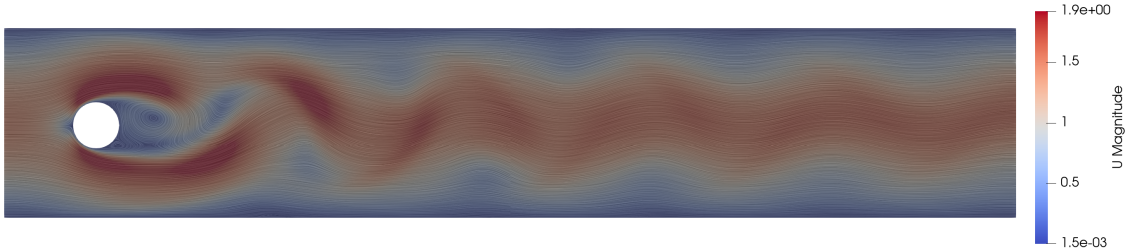

Figure 2.10: Velocity(magnitude) distribution at time $t = 5s$.

Due to the detachment of the boundary layer, the vortices are formed in the wake. The vortices propagate periodically with a certain frequency. These periodically propagating vortices are called von Kármán vortex street. The periodic behavior of von Kármán vortex street induces periodic drag and lift forces as shown in figure 2.5 and 2.6. These periodic drag and lift forces cause vibration on the object.

The frequency of drag force and lift is obtained by fitting the sine curve as shown in figure 2.9. Intuitively the frequency of lift is equal to the frequency of propagating eddies in the wake. From figure 2.11, when the eddies are forming near to the point P the lift is maximum and similarly near the point the R lift is minimum. Similarly, the drag has a higher value when the vortices are generated near the P point and R point but when the vortices are forming near to point Q the drag value is minimum. Hence, the frequency of Drag is almost two times higher than the frequency of lift $f_{c_D} \approx 2f_{c_L}$, which is also shown in table 2.3.

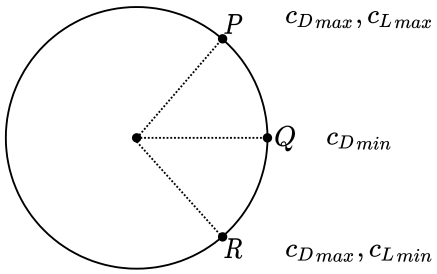

Figure 2.11: The point P, Q, and R on the surface of the cylinder. At near to point P the value of c_D is maximum and value of c_L is maximum, while at near to point R the value of c_D is maximum and value of c_L is minimum. For vortices generation at near to point Q the value of c_D is minimum and $c_L = 0$.

Chapter 3

Open Loop Control

In open-loop control, the controller is not subjected to any feedback from the system. In open-loop control, the control strategy is carried out and evaluated initially, and then it is fit in the system to achieve control. The feedback from the system in terms of evaluation of control strategy is not used during the running of the system but at the end of the running of the system. The improvement in the control strategy for open-loop control is accomplished by analyzing the strategy after the completion of the system run or simulation. Therefore the open-loop control strategy is also called offline learning.

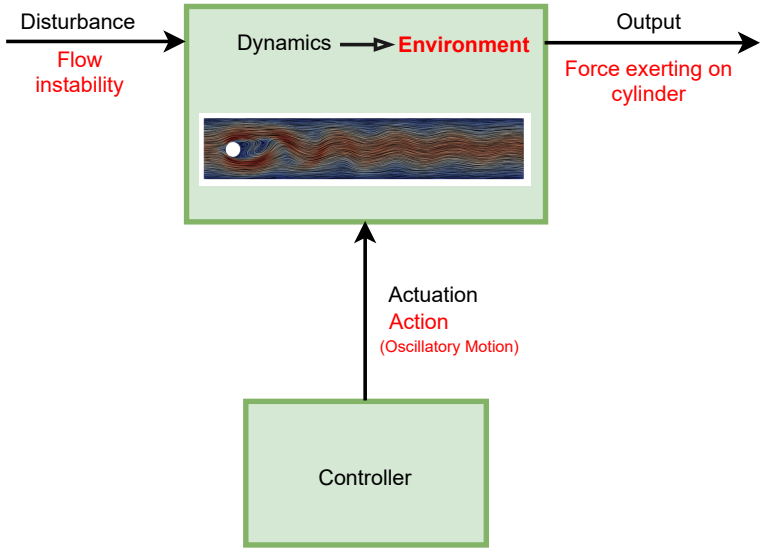

Figure 3.1: Open loop control diagram to reduce drag and lift acting on cylinder.

As shown in figure 3.1, the environment is considered as the flow problem with flow across the cylinder. The controller feeds actuation for the environment which is pre-defined. For open-loop control, in this study, the cylinder is subjected to external energy in terms of oscillatory rotations. The c_D and c_L oscillates almost exactly like sine wave (see 2.9). Therefore, the cylinder is oscillatory rotated in order to counteract the c_D and c_L fluctuations. Hence, the environment receives a pre-defined action in terms of oscillatory rotations from the controller. The oscillatory rotation of the cylinder is defined as shown in equation 3.1. The outcome from the system after the action of control is a measure of forces acting on the cylinder. The open-loop strategy is improved by evaluating the outcome from the system at the end of simulation.

$$\omega = A \sin(2\pi f t) \tag{3.1}$$

where, ω is defined as the angular velocity of the cylinder. A is the amplitude, and f is the frequency of sine oscillation. The parameter, amplitude A and frequency f controls the oscillatory motion of cylinder. The boundary condition at the wall of cylinder is applied as shown in equation 3.1. Where, r is denoted as cylinder radius, ω is angular velocity from equation and t is denoted as unit vector tangential to cylinder surface.

Boundary condition at cylinder surface wall: Oscillatory,

$$\mathbf{u} = \omega r \mathbf{t} \tag{3.2}$$

3.1 Parameter study

For open-loop control, the primary objective is to reduce the drag and lift along with its fluctuations. Moreover, for open-loop control, in order to improve the control strategy the values of oscillation parameters A and f is changed in order to reduce the drag along with its fluctuations. To solve the coupling between oscillation parameters and forces acting on the cylinder, the parameter study is performed. There is no analytical function which can describe the coupling. For parameter study, the objective function Φ is formulated as equation 3.3 in order to reduce the mean of drag and lift along with its fluctuations. The first term of equation 3.3 represents the mean of drag ($\bar{c_D}$) and lift ($\bar{c_L}$). Moreover, the second term of the equation 3.3 represents the amplitude of fluctuation of drag ($c_{Dmax} - c_{Dmin}$) and lift ($c_{Lmax} - c_{Lmin}$).

$$\Phi = w_1 \left(\sqrt{\overline{c_D}^2 + \overline{c_L}^2} \right) + w_2 \left(\sqrt{(c_{Dmax} - c_{Dmin})^2 + (c_{Lmax} - c_{Lmin})^2} \right) \tag{3.3}$$

Where, $\overline{c_D}$ and $\overline{c_L}$ is the mean values of drag and lift respectively. c_{Dmax} and c_{Dmin} is the maximum and minimum value of drag. c_{Lmax} and c_{Lmin} is the maximum and minimum value of lift. w_1 and w_2 in equation 3.3 are the weights for the mean value and amplitude of fluctuations of drag and lift respectively.

For simplicity and better understanding, the variable amplitude and frequency are normalized. The amplitude, which is also referred to as peak rotational rate is normalized with mean inlet velocity as,

$$\Omega = \frac{d}{2\bar{U}} \, \Omega_{max} \tag{3.4}$$

where, Ω_{max} is peak rotational rate of the cylinder, d is the diameter of the cylinder and U_∞ is free-stream velocity far ahead from the cylinder. The frequency is also normalized with mean inlet velocity and cylinder diameter as,

$$S_f = \frac{d}{\bar{U}} f \tag{3.5}$$

Where S_f is also called the Strouhal number. As Ω and S_f are a dimensionless quantity, these quantities are used in further study.

The parameter study is performed by sampling data for Ω and S_f, which are evaluated over the objective function. Then, the parameters which have minimum value of objective function as the optimal parameters.

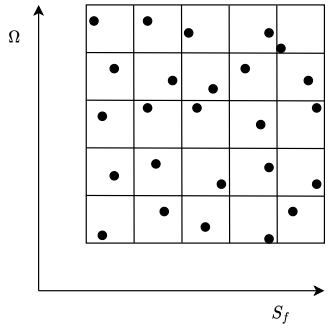

Figure 3.2: Latin Hypercube sampling; near-random sampling from finite small region.

The sampling of parameters is conducted by Latin Hypercube Sampling(LHS) [5]. In LHS, the sampling achieved from near-random sampling from finite small defined regions, which is also shown in figure 3.2. The LHS sampling method improves the exploration in the region.

For parameters Ω and S_f, data are sampled and for each data, the simulation is executed. For each data point, the value of the objective function is calculated. Total 800 data points are sampled for the range of Ω from 0.1 to 3.5 and the range of S_f from 5.5 to 14. The higher value of Ω and S_f is avoided to limit the energy supplied to the cylinder. The higher values of Ω and S_f also cause the Magnus effect, which is not desirable for this study as it increases the lift [7]. he contours plot for objective function representing each data point is shown in figure 3.3. In figure 3.3 the weights for objective function are set to $w_1 = 1$ and $w_2 = 1$.

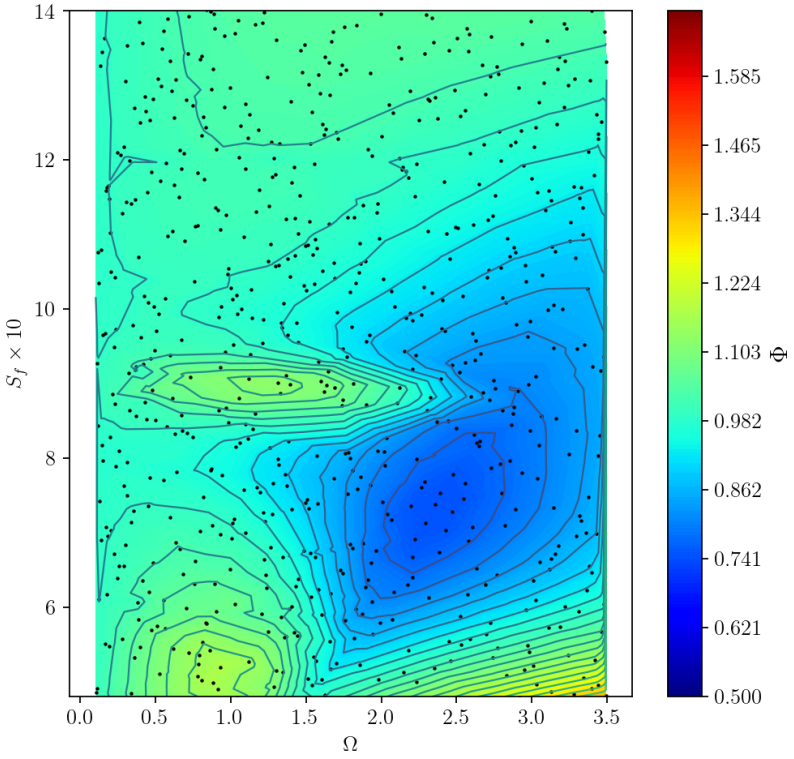

Figure 3.3: Contour plot for objective function Φ over Ω and S_f, The minimum value of plot 0.747 is at $\Omega = 2.357$ and $S_f = 0.7381$.

From figure 3.3, The minimum value of objective function Φ is 0.747 and it lies at $\Omega = 2.357$ and $S_f = 0.7381$. The amplitude and frequency of the natural vortex shredding is $\Omega = 1.7838$ and $S_f = 0.3002$ respectively. By changing the values of weights of an objective function as shown in figure 3.4, the minimum value of the objective function is the same. That is because the mean value of drag is significantly higher than the mean value of lift. Moreover, the difference between the maximum and minimum value of drag is also significantly higher than the difference between the maximum and minimum value of lift. This results in the insensitivity of objective function towards the weights because of a higher value of drag and its fluctuations. These results are helpful later to design feedback(reward) function in close loop control.

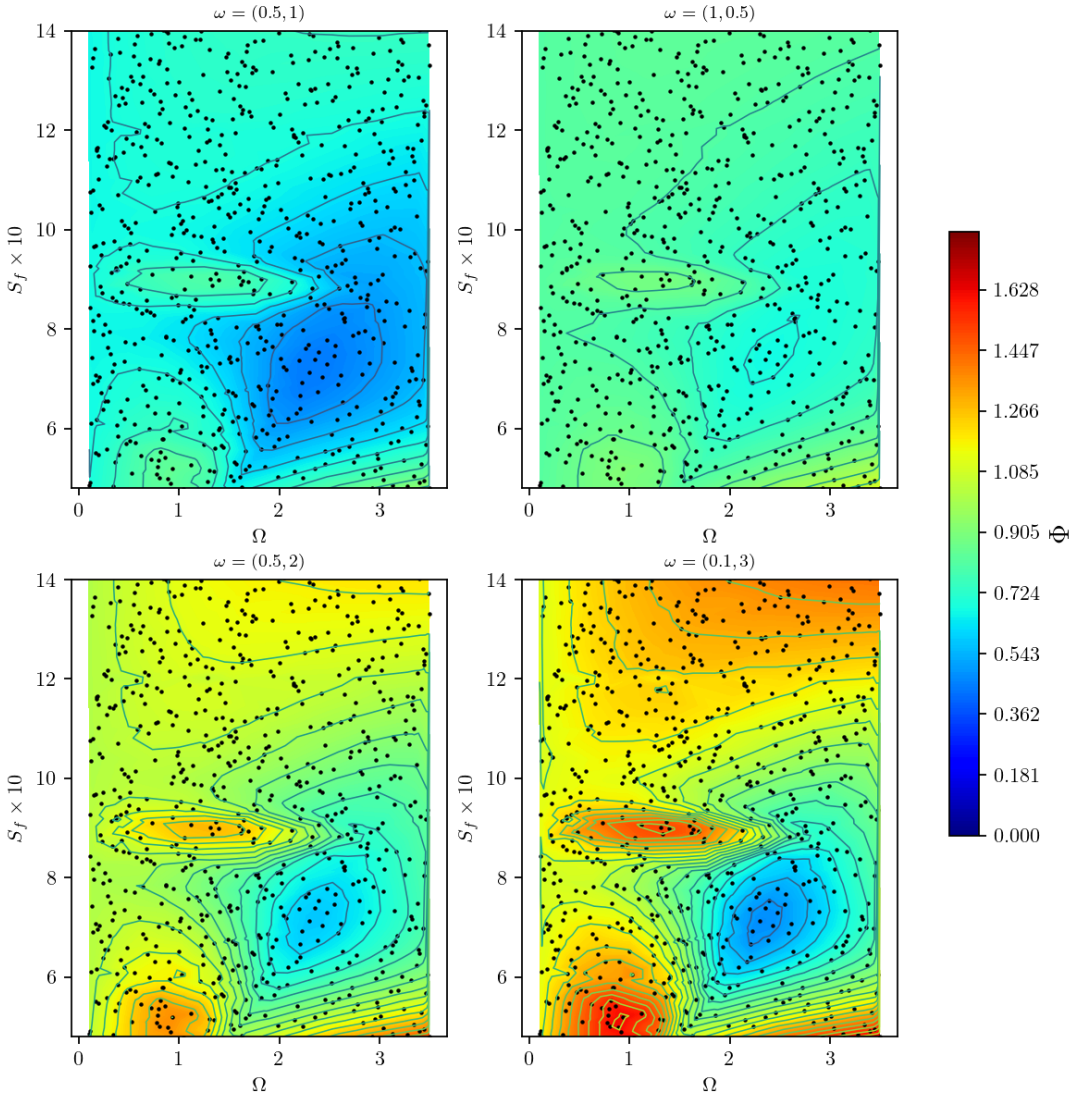

Figure 3.4: Contour plot for objective function Φ over Ω and S_f for different weights, but the minimum value of all case is at $\Omega = 2.357$ and $S_f = 0.7381$, which indicate insensitivity for weights.

3.1.1 Functional Approximation for sampled data

In parameter study, sampling the fine data results in accurate optimal parameters but it imposes a high computational cost. Whereas, sampling the coarse data results in an inaccurate representation of optimal parameters. To overcome with this problem, the functional approximation is performed over the coarse sampled data with objective function ϕ, where the minimum of this functional approximation represent the accurate optimal parameters. The function is approximated for the data sampled by the LHS method with the objective function values. This functional approximation is used to find the optimal parameter Ω and S_f. In other words, this functional approximation is also used to verify the optimal parameter obtained by parameter study. This functional approximation is achieved by Kernel ridge regression(KRR) [1]. In scikit-learn functionality for KRR, the data in terms of Ω and S_f is supplied and the regression gives the best fit function representing the values of the objective function. In figure 3.5, the values of objective function approximated by function are plotted over originally sampled LHS data. The difference between the parameter study and function approximation is plotted in figure 3.6. Hence, the figure 3.6 verifies the accurate approximation for the sampled data. Now this approximated function is used to calculate the objective function value at any sets of the value of Ω and S_f without executing the simulation. Using an approximated function for non-observed data is called generalization in machine learning. A good generalization is vital in machine learning problems.

For the generalization, in which the value of the objective function is carried out for uniformly generated data in order to verify the functional approximation with original data. The plot for generalization is shown in figure 3.7.

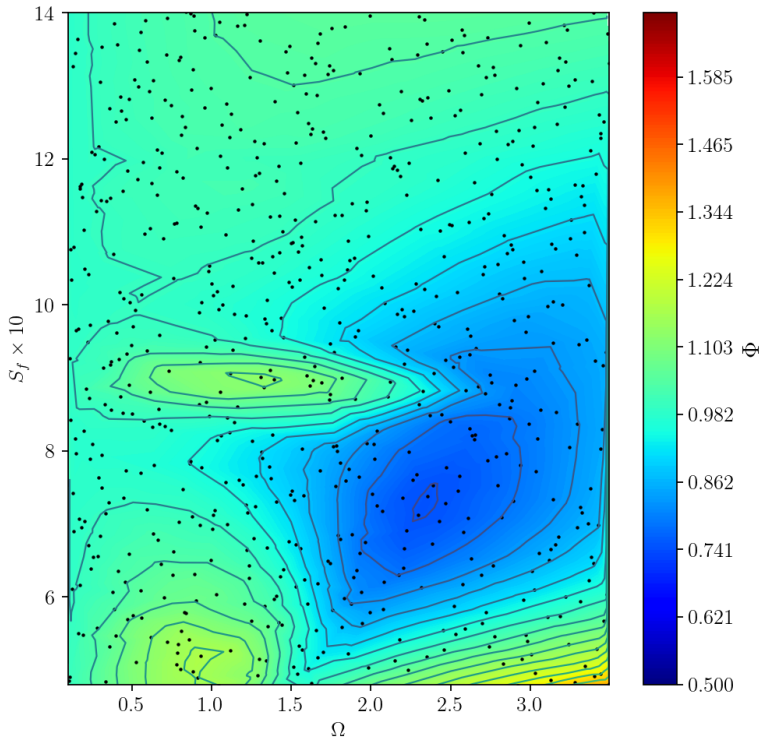

Figure 3.5: Contour plot for predicted objective function Φ by KRR over origional LHS sampled Ω and S_f data, where weight is $w = (1, 1)$. The minimun value of objective function $\Phi = 0.747$ lies at $\Omega = 2.357$ and $S_f = 0.7381$.

From figure 3.7, The functional approximation by uniformly generated data represents the accurate value of the objective function. Hence, the minimum value of the objective function is carried

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.kernel_ridge.KernelRidge.html`

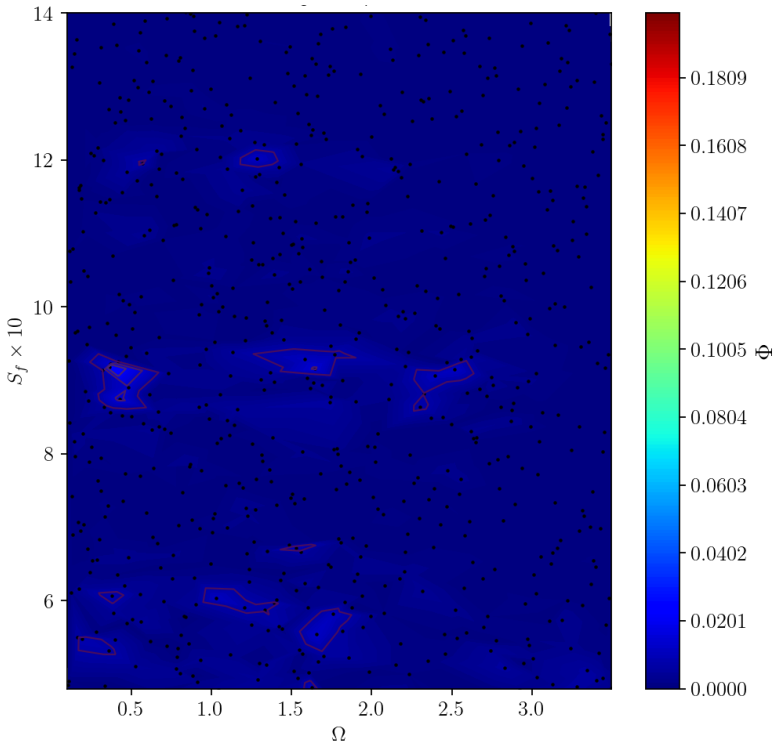

Figure 3.6: Contour plot for difference between original value of objective fucntion and predicted objective function Φ by KRR over LHS sampled Ω and S_f data, where weight is $w = (1, 1)$.

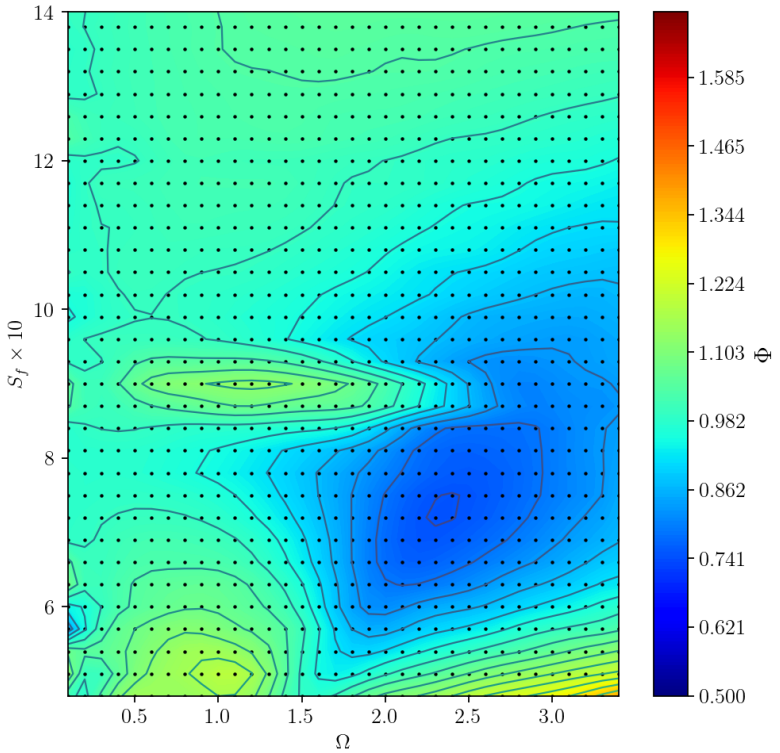

Figure 3.7: Contour plot for predicted objective function Φ by KRR over uniformly generated Ω and S_f data, where weight is $w = (1, 1)$. The minimun value of objective function $\Phi = 0.747$ lies at $\Omega = 2.294$ and $S_f = 0.7128$.

out with the approximated function. The minimum of functional approximation is $\Phi = 0.741$, which is also approximately equal to the minimum of the parameter study. Thus, the functional approximation from sampled data not only helps to verify the result from the parameter study but also ensures the best exploration in the defined region for the optimal parameter.

3.1.2 Results

The Simulation is executed at the optimal parameter values of $\Omega = 2.294$ and $S_f = 0.7128$, This simulation is referred to as a controlled case, and the simulation without open-loop control is referred to as the uncontrolled case. From figure 3.8, the propagation of vortices is decreased and in the wake, the flow is more stable than uncontrolled case.

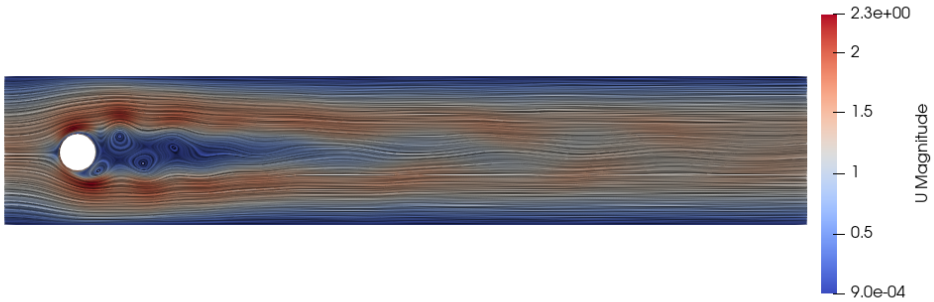

Figure 3.8: Velocity(magnitude) distribution at $t = 5s$ for flow around the cylinder, where it is oscillated by optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$.

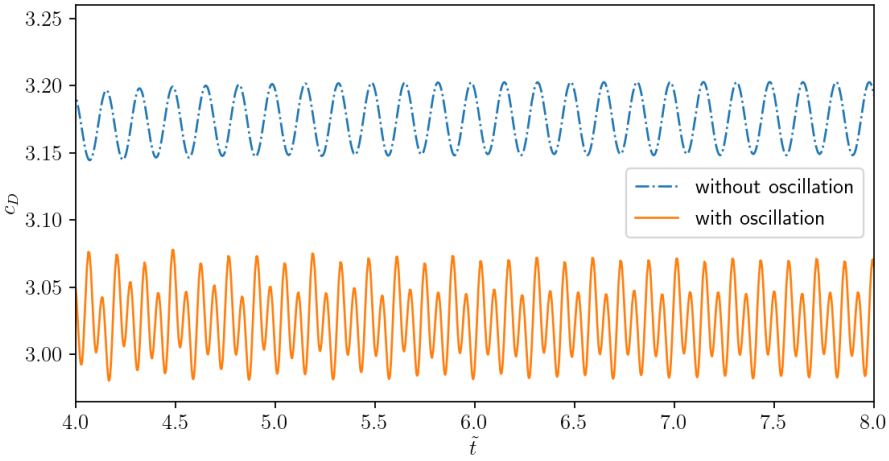

Figure 3.9: c_D values from $t = 4s$ to $t = 8s$ for controlled and uncontrolled case, where cylinder is oscillated by optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$.

case	Φ	$\overline{c_D}$	c_{Dmax}	c_{Dmin}	$\overline{c_L}$	c_{Lmax}	c_{Lmin}
Uncontrolled case	4.9566	3.1721	3.1965	3.1474	-0.0126	0.8789	-0.9049
controlled case	0.7470	3.0256	3.0778	2.9809	0.0051	0.3262	-0.3231

Table 3.1: Table containing the values for controlled case and uncontrolled case.

In figure 3.9, the values of c_D is plotted from $t = 4s$ to $t = 8s$ for both controlled and uncontrolled case. Similarly, in figure 3.10, the values of c_L is plotted from $t = 4s$ to $t = 8s$ for both controlled

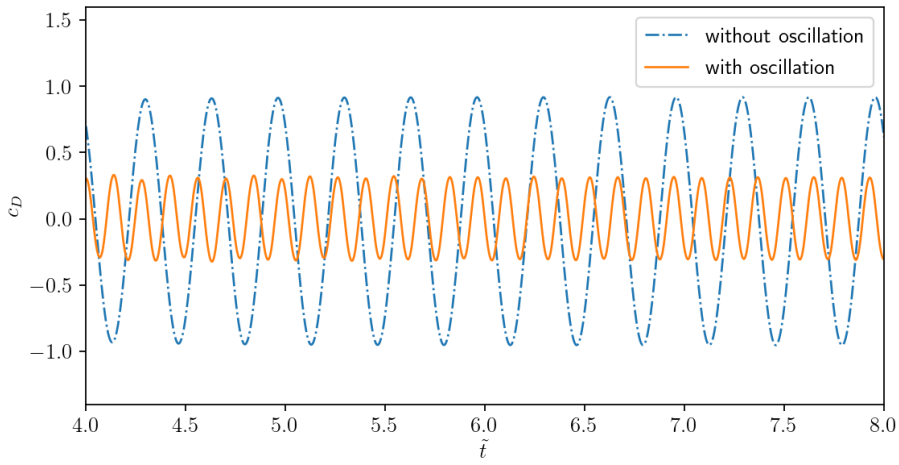

Figure 3.10: c_L values from $t = 4s$ to $t = 8s$ for controlled and uncontrolled case, where cylinder is oscillated by optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$.

and uncontrolled case. In figure 4.12, the pressure distribution over the polar coordinates at the surface of the cylinder is plotted. By using an open-loop control strategy, the pressure at the cylinder surface is changed, which results in a change in drag and lift values than an uncontrolled case. In the open-loop control strategy, at the top and bottom of the cylinder surface, the pressure value is decreased hence, the mean value of drag and lift is decreased. The mean value of drag is reduced by 4.61% and the mean value of lift is improved by 0.56% for controlled case, where the cylinder oscillates at optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$. However, In the open-loop control strategy, at the cylinder surface, the pressure fluctuation is high which results in increasing the frequency and amplitude of the drag and lift fluctuations. Therefore, the amplitude of fluctuations of drag is increased 97.35% by using the open-loop control strategy. The amplitude of lift fluctuation is decreased by 63.60%. Moreover, the objective of this study is to reduce mean values of drag and lift along with its fluctuation, where the results suggests strong correlation between mean value of drag and its fluctuation for objective function ϕ by oscillation cylinder with sine wave function. Similarly, the maximum value of drag is reduced by 5.34% and the maximum value of lift is decreased by 17.28% for the controlled case. The minimum value of drag is reduced by 3.86% and the minimum value of lift is improved by 18.48%. The values are also contained in table 3.1.

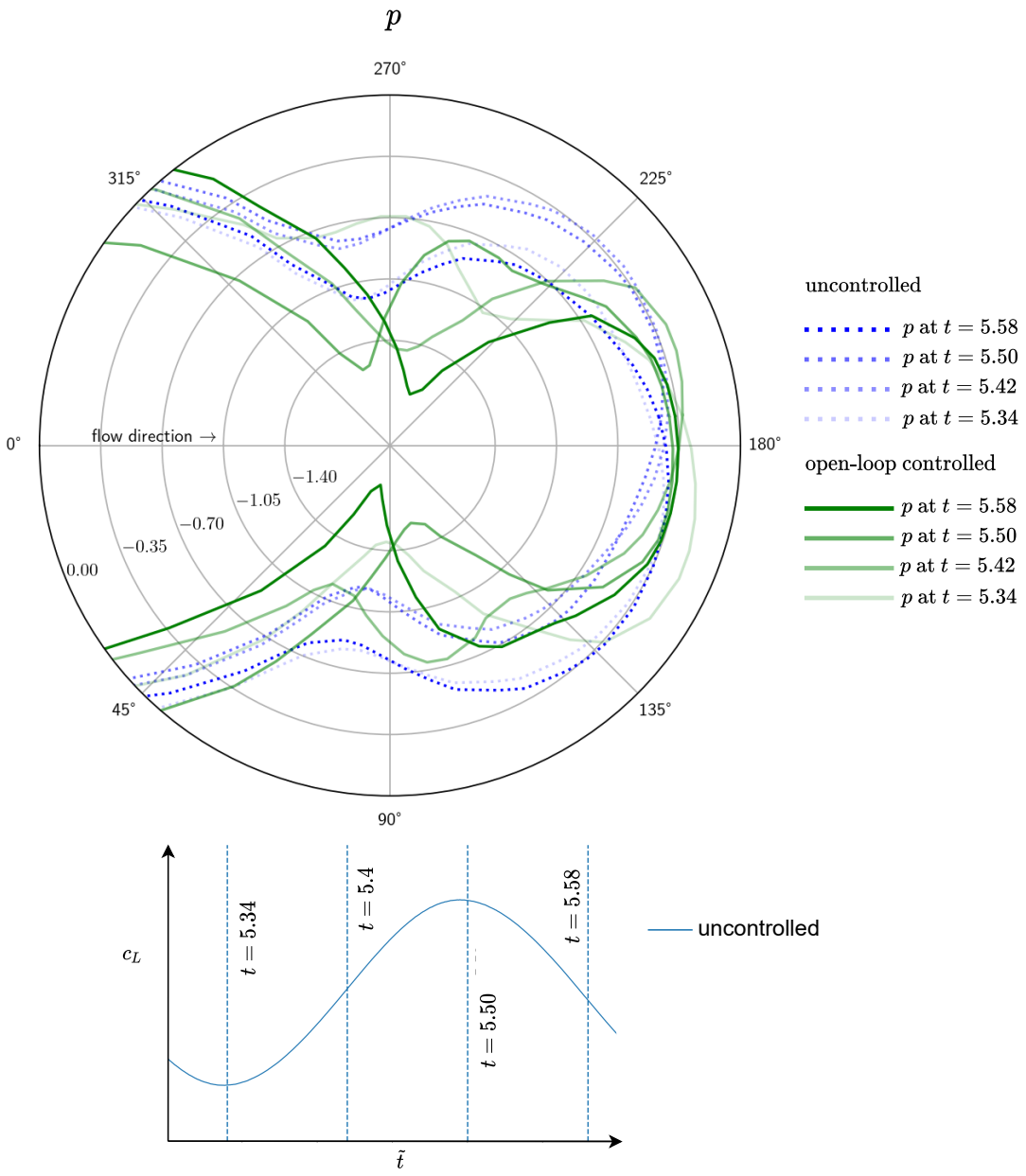

Figure 3.11: Pressure distribution in polar coordinates at the surface of the cylinder for controlled and uncontrolled case. In open-loop controlled case the cylinder is oscillated by optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$.

Chapter 4

Closed-Loop Control

The closed-loop control method is a classification of active flow control, in which the system utilizes the feedback from the system. The feedback from the system is utilized to evaluate the flow control strategy. The control strategy may be improved in order to achieve the objectives by the evaluation of the control strategy. The evaluation and improvement of strategy are performed during the running of the system. Hence, closed-loop control is also called offline control.

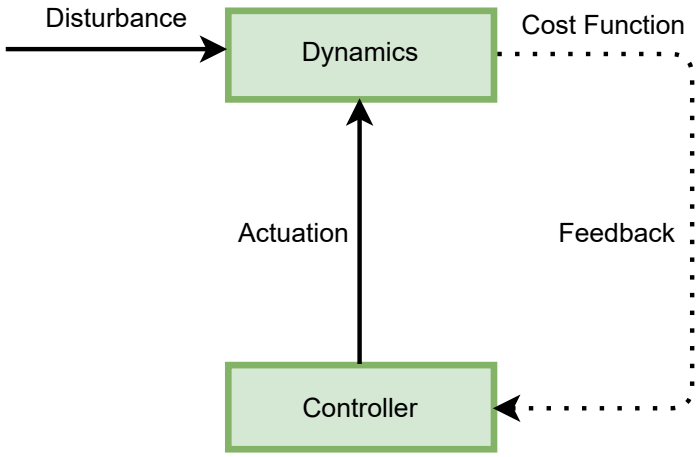

Figure 4.1: Closed-loop control for flow across cylinder.

For closed-loop control of flow around the cylinder, the cylinder is subjected to the angular velocity. The objective of the control is to reduce drag and lift with its periodicity in order to stabilize the cylinder. Hence, the objective function is formulated with drag and lift. The feedback is the evaluation of this objective function. The objective function is also referred to as the cost function since it is subjected to minimization. The controller receives feedback from the system and applies the actuation to the system. In this study, the actuation is the rotation of the cylinder, hence the controller applies angular velocity of the cylinder on the system in order to minimize the drag and lift.

4.1 Deep Reinforcement learning

Reinforcement learning is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning.[1] In reinforcement learning, the agent takes an action to maximize the cumulative reward in order to achieve the objectives. In deep reinforcement learning, the learning of taking an action based on feedback is achieved by deep learning. To design a

[1] addiai.com/reinforcement-learning/

controller for a closed-loop system, DRL is has proven to achieve state of the art results [15, 23]. In DRL, the controller is referred to as an agent. The agent is a mimic of the human brain which can understand the coupling between actuation, state of the system, and cost function. The agent takes an action such that the cost function is minimized. The pictorial representation of DRL is represented in figure 4.2.

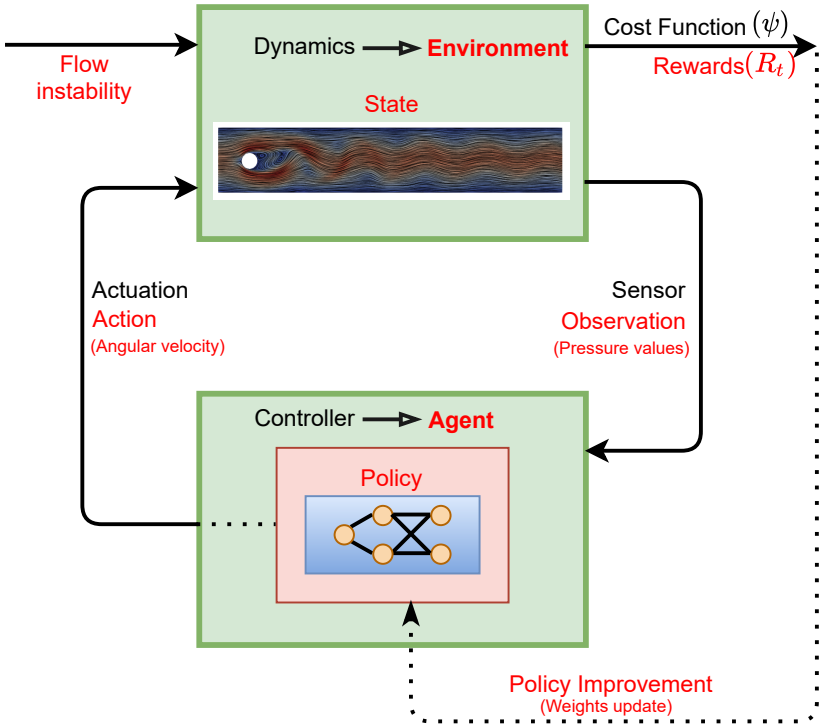

Figure 4.2: Closed-loop control by deep reinforcement learning.

4.1.1 Environment

In reinforcement learning, the system is called an environment. In this study, the environment is referred to as the flow problem. For DRL, episode is the part of simulation between the starting of control and ending of control. In DRL, a trajectory is the entirety of all state-action-reward-next state tuple of one episode, where the surface pressure is the state, the angular velocity is the action, and the resulting forces results in the reward. The complete simulation is called as a trajectory in DRL. The agent supplies the actuation in form of angular velocity based on the current state of the environment. The flow problem is formulated as described in chapter 2. The boundary condition for the cylinder is set to:

$$\mathbf{u} = \omega r \mathbf{t} \tag{4.1}$$

Where, ω is angular velocity. The r is cylinder radius and $\mathbf{t}$ is denoted as a unit vector tangential to the cylinder surface.

Horizon

In reinforcement learning, the length of the trajectory is called the horizon. Horizon is divided into two categories, finite horizon, and infinite horizon. The selection of horizon is problem-specific. For a problem such as finding the shortest path, the infinite horizon is selected as reaching at the goal by minimum state transition is essential. The state transition occurs by

taking an action in the environment. For the infinite horizon, the length of trajectory is the total state transition required in order to achieve the goal. Initially during learning, for such a problem the length of trajectory is set to infinite. After reaching the goal the information is bootstrapped and the length of the trajectory is calculated. Thus, the trajectory length varies while learning, and the main objective is to reduce the trajectory length hence, achieving the shortest path. For the infinite-horizon, the trajectory is formulated as shown in equation 4.2. The trajectory is referred to as τ and the state-action pair is represented by s and a. The state transition is denoted as $\{s_t, a_t, R_t, s_{t+1}\}$ where, by taking action a_t, the state transitions from s_t to s_{t+1} , and reward R_t is obtained.

$$\tau = (\{s_0, a_0, R_0, s_1\}, \{s_1, a_1, R_1, s_2\}, \dots) \tag{4.2}$$

For the problems, where the number of state transitions or the system dynamics, is independent of trajectory length, the finite horizon is considered. For the finite-horizon, the trajectory is formulated as shown in equation 4.3. The length of trajectory is fixed for the finite number of state transitions. In this study, the controlling flow is independent of the length of simulation or trajectory hence, the finite horizon is considered in this study.

$$\tau = (\{s_0, a_0, R_0, s_1\}, \{s_1, a_1, R_1, s_2\}, \dots, \{s_n, a_n, R_n, s_{n+1}\}) \quad , n \in N \tag{4.3}$$

4.1.2 State observations

In reinforcement learning, at each control step t, the agent receives partial state observations S_t and a reward R_t quantifying the current performance of the environment [11]. The state observation contains information about the state of the system at control time t. In this study, the state observations are considered as the pressure values at the cylinder surface. In recent studies, [13, 23, 11], for controlling flow by DRL, the pressure sensors are placed in the wake of the cylinder. In this study, the pressure values are obtained from the mesh at the cylinder surface as shown in figure 4.3, but in a real physical model, the pressure values are obtained by the pressure sensors. Mounting the pressure sensors on the cylinder surface is more intuitive and practical for the flow around the cylinder.

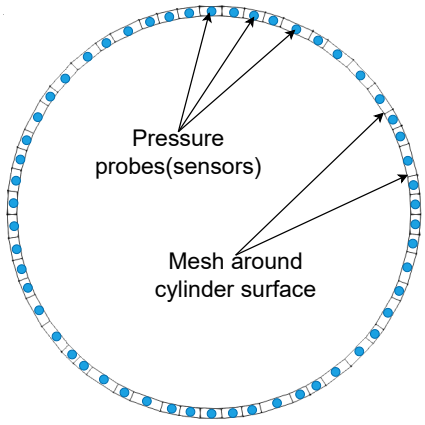

Figure 4.3: State values (pressure sensors) on the cylinder surface which are fetched from the simulation.

4.1.3 Reward and Return

The agent computes the reward (R_t) at each state transition from s_t to s_{t+1}. The reward represents the advantage of the action for the s_t after the transition. In other words, reward R_t is the gain by taking an action at control time step t.

$$R_t = R(s_t, a_t, s_{t+1}) \tag{4.4}$$

The objective of this study is to reduce c_D and c_L. Hence, the objective function is formulated with c_D and c_L. The reward function is computed from the environment as a converse of the objective function where, the reward function is formulated as a weighted sum of drag, lift, and rotational rate of cylinder. The rotational rate of cylinder is introduced in the reward function in order to mitigate the energy supplied to the cylinder. Therefore, the value of the reward function is large if the c_D and c_L are small and vice-versa. The reward function is designed as shown in 4.5. Hence, the main goal of reinforcement learning is to maximize the cumulative rewards at the end of the trajectory. The reward function is utilized to evaluate the performance of the agent and to improve it.

$$R_t = r_0 - \left(r_1 c_D + r_2 c_L + r_3 \dot{\theta} + r_4 \frac{\mathrm{d}\dot{\theta}}{\mathrm{d}t} \right), \tag{4.5}$$

where, $r_0, r_1, \ldots, r_4$ are the constants. The rotational rate of cylinder is defined as $\dot{\theta}$ and $\frac{\mathrm{d}\dot{\theta}}{\mathrm{d}t}$ denotes as the rotational acceleration of the cylinder. The mean value of drag (c_D) and its fluctuation is higher in magnitude than lift (c_L) hence, the primary objective is to reduce c_D. Therefore, the weight for the drag is larger than the weight for lift. The c_D yields minimum for the higher rotational rate but the higher rotational rate is energy inefficient and leads to Magnus effect [7]. Thus lower rotational rate is desired for the efficient control strategy. Therefore, the rotational rate and rotational acceleration are formulated in the reward function. The rotational rate ($\dot{\theta}$) penalizes the higher angular velocity and the rotational accelerations and it ensures smooth transition of angular velocity from one state to another. The total rewards at the end of the trajectory are computed as the discounted sum of rewards collected from every control time step. The total discounted sum of rewards is called return (G_τ). For the finite-horizon trajectory, the (G_τ) is computed as shown in equation 4.7,

$$G_\tau = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots \tag{4.6}$$

$$= \sum_{t=0}^{T} \gamma^t R_t \tag{4.7}$$

where, γ is discount factor and the $\gamma \in]0; 1[$. If the value of γ is close to zero then the current reward is weighted most and the future rewards are weighted less. If $\gamma = 1$ then all rewards are weighted the same. In DRL, for good policy, the early future rewards are more relevant than the later future rewards. Hence, the optimal choice for γ is close to 1, where the early future rewards are weighted more than the later future rewards. In this study, the discount factor is set to $\gamma = 0.97$.

4.1.4 Agent

The agent receives state observation s_t and outputs action a_t. The agent is considered as a brain of reinforcement learning. In DRL, the agent is a neural network (NN), which is also called a policy network. The NN is designed with an input layer, output layer, and hidden layers, in which each layer consists of a finite number of neurons. Each neuron is subjected to mathematical function with weight and bias [18]. The state observations are feed into the network and the network yields the output as an actuation, which is angular velocity. The number of hidden layers and number of neurons in each hidden layer is considered as an ansatz and tuned for better performance. In learning, the agent performs policy improvement by R_t and s_t.

Exploitation and Exploration

The agent is restricted to explore in the sub-optimal region of state-action space by taking exact output value from the policy network. Considering the exact value from the policy network is called exploitation. To explore in the sub-optimal region of state-action space for the agent, randomness is introduced in the output of the policy network. Introducing the randomness in the output of the policy network is called exploration. All learning algorithms aim at solving the exploration-exploitation dilemma, meaning achieving the best performance at a minimum learning cost [11]. Efficient exploration of the state-action subspace is a key factor in the performance of the learning algorithm [11]. The selection of exploration is crucial in DRL as too much randomness yields high computational cost and reduces the overall performance of learning. Thus, finding the right balance between exploration and exploitation is crucial in DRL. To establish the balance between exploration and exploitation, the agent is introduced exploration variance as a learning parameter. Hence, the output layer of the policy network is subjected to two parameters, mean of action μ_a and standard deviation of action σ_a. The randomness for the learning is introduced by sampling an action from the Gaussian distribution.

$$a_t \sim \mathcal{N}(\mu_a, \sigma_a^2) \tag{4.8}$$

Policy network

The policy network is designed with input layers, two hidden layers, and an output layer. The input layer consists of 54 neurons as there are 54 state observations available at the surface of the cylinder. Each hidden layer contains 64 neurons. The output layer is subjected to two neurons for μ_a and σ_a. The activation function for the hidden layer is used as ReLU (Rectified Linear Unit). The state observation s_t is fed into the neural network at each control time step, which yields output as μ_a and σ_a.

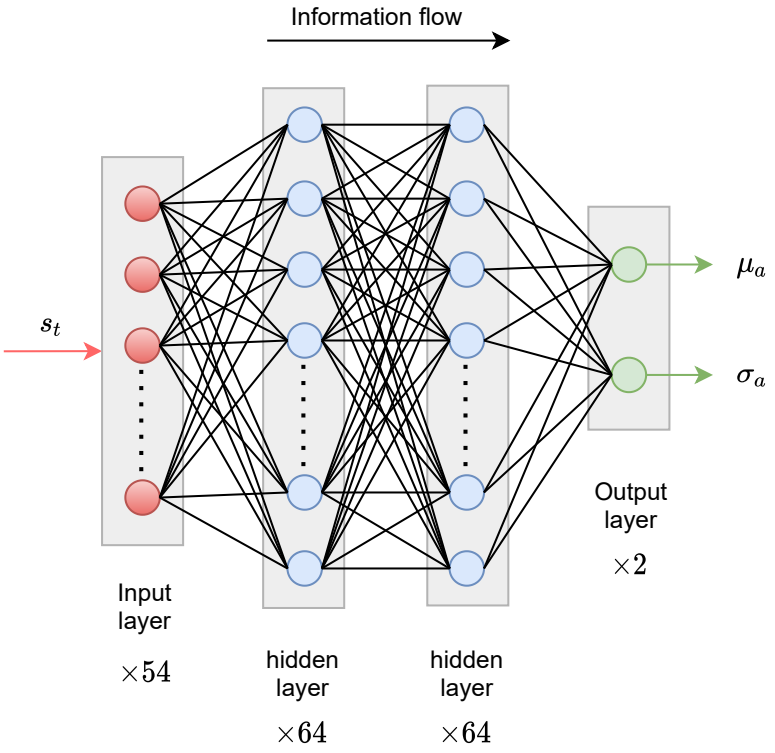

Figure 4.4: Policy network for DRL.

Control Time Step

In this study, the control time step is considered as,

$$\Delta t_c = 20 \times \Delta t_{cfd} \tag{4.9}$$

Where, Δt_c is control time step for DRL and Δt_{cfd} is simulation time step. At each control time step Δt_c, action is sampled and feed into system based on s_t at that time step. The R_t is calculated at that time step from the c_D, c_L, $\dot{\theta}$ and $\frac{d\dot{\theta}}{dt}$ available at that time step. The transition of action $a_t \rightarrow a_{t+1}$ is implemented as linear in this study.

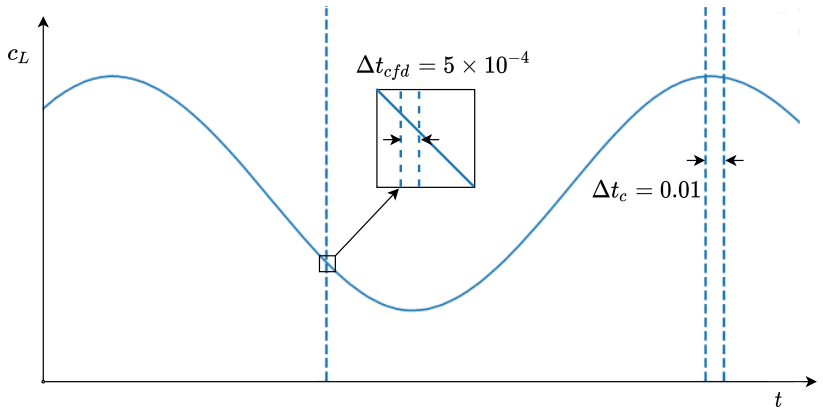

Figure 4.5: Control time step and simulation time step over lift. The cycle of lift in the figure represents natural vortex shredding frequency.

State-value function

The state-value function estimates the return at the end of the trajectory for being in the state s_t. The state-value function represents how valuable the state is. In other words, The state-value function gives information about the expected return G_τ for being in a state s by taking action from policy [8]. The state-value function is formulated as,

$$v_\pi(s) = \mathbb{E}[G_t | s_t = s] \tag{4.10}$$

Where, $v_\pi(s)$ is the value of state s by following policy π. In the present study, the policy π is a NN. The state-value function is also known as V-function or value function.

Action-value function

The action-value function estimates the return at the end of the trajectory for being in the state s_t and by taking an action a_t. The action-value function gives information about how valuable the taken action is for being in state s. The action-value function represents the expected reward at the end of the trajectory by being in state s and taking the action a by following policy π [8]. The action-value function is formulated as,

$$q_\pi(s, a) = \mathbb{E}[G_t | s_t = s, a_t = a] \tag{4.11}$$

Where, $q_\pi(s, a)$ is the q value of state s and action a by following policy π for. The action-value function is also called as Q-function.

Action-advantage function

The action-advantage function is a difference of action-value function of action a in state s and state-value function of state s under policy π. The action-advantage function represents the advantage of taking action a instead of following policy π [8]. The action-advantage function is formulate as,

$$a_\pi(s,a) = q_\pi(s,a) - v_\pi(s) \tag{4.12}$$

Where, $a_\pi(s,a)$ is the advantage of taking action a in state s. The action-advantage function is also called as A-function or advantage function.

Optimal strategy

The main objective for closed-loop control by DRL is to minimize drag, lift, and its fluctuations. The policy is called an optimal policy for possible minimum drag and lift values. For the optimal policy, the reward function value is maximum. For an optimal policy, the expected returns are greater than or equal to all other policies. An optimal state-value function is a state-value function with the maximum value across all policies for all states. Likewise, an optimal action-value function is an action-value function with the maximum value across all policies for all state-action pairs [8]. For optimal policy, the value of the action-advantage function is zero or close to zero as for optimal policy, there is no better action than the action following the optimal policy. Hence, for optimal control the main objective is to maximize the action-value function of action a in state s and state-value function of state s under policy π and to minimize the action-advantage function of action a in state s.

4.2 Proximal Policy Optimization method

Proximal policy optimization(PPO) is a method in deep reinforcement learning which is widely used in DRL. PPO method is a modern and very efficient method to approximate state-value function and action-advantage function at the same time. The block diagram of PPO is shown in figure 4.6.

For PPO, to approximate state-value function, a neural network called the value network is used. The policy network is also known as actor and value network as a critic, as the value network is utilized to approximate action-advantage function. In other words, the value network judges the action taken by the policy network in the PPO algorithm. Thus, PPO algorithm is also called an actor-critic method. In this study, the NN architecture of the value network is the same as the policy network shown in figure 4.4. However, the output layer only consists of one neuron, which represents the value of the state-value function.

In the PPO algorithm, the environment samples total M number of trajectories which is stored in the buffer memory. This buffer memory is called a reply buffer or experience in DRL. To fill the buffer memory, PPO allows sampling multiple N numbers of trajectories at the same time. This N number of trajectories is sampled by the N-number of workers. Each worker is assigned to the simulation of the flow problem and when the simulation is finished, it will be stored in a replay buffer and the worker gets the new simulation to perform until the buffer is full.

For efficient learning and better performance of PPO, randomness is introduced for starting for control in trajectories. The randomness in starting of control allows capturing all possible initial state values. In recent studies [23, 11, 13], the starting time of control is taken after the natural vortex shredding frequency stabilizes. However, in this study, the starting time of control is

considered between 0 to 4 sec, $\Delta t_c^0 \in [0, 4]$ in order to capture all possible states and also to improve the flow control for initial flow disturbance.

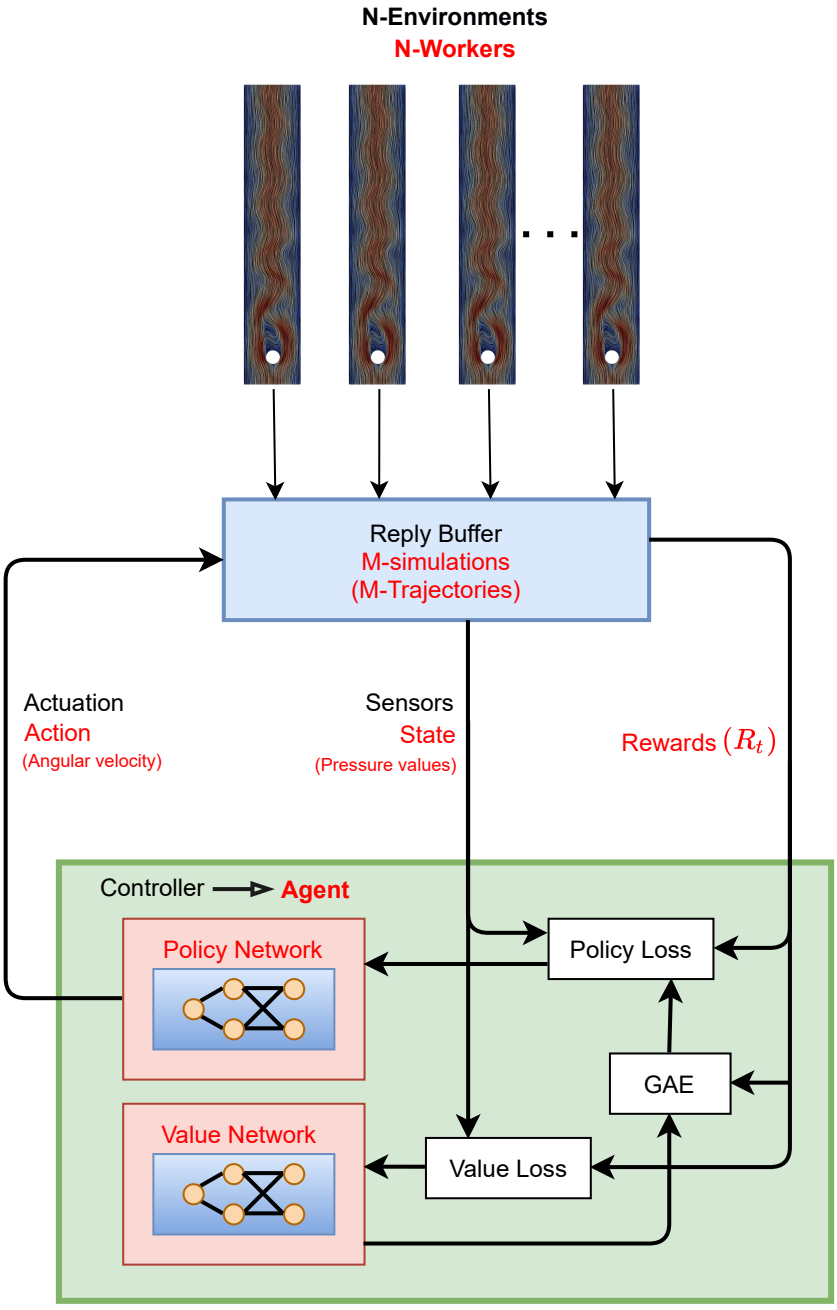

Figure 4.6: Block diagram of PPO algorithm with N numbers of workers and M number of total trajectories.

4.2.1 State-value function estimation

In the PPO algorithm, the state-value function is represented by the value network (NN). Hence, using a nonlinear function approximator to represent the value function, the simplest approach is to solve a nonlinear regression problem [19]:

$$\underset{\phi}{\text{minimize}} \sum_{n=1}^{N} \left\| V_\phi(s_n) - \hat{R}_t \right\|^2 \tag{4.13}$$

Where, $V_\phi(s_n)$ is value of the state n obtained from the value network, where the parameters of value network are ϕ. The indexes n are overall time-steps in a batch of trajectories to encounter. $\hat{R}_t$ is discounted sum of rewards from the current time step $t = l$ to end of the trajectory $t = T$. Hence, $\hat{R}_t$ is referred as rewards-to-go as it considers future rewards from the time step $t = l$.

$$\hat{R}_t = \sum_{l=0}^{T} \gamma^l r_{t+l} \tag{4.14}$$

Hence, to approximate the state-value function, the value network is updated by back-propagating value loss. For value network, by back-propagating the value loss the parameters ϕ of value network are updated from $\phi_k \rightarrow \phi_{k+1}$. The parameter updating step is subjected to a minimization problem. The parameters are updated in order to minimize the error between the collected data from the trajectory and the output of the network. Hence, the value loss is the mean-squared error averaged over the number of trajectories.

$$\text{value loss} = \frac{1}{|\mathcal{D}_k|\,T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2 \tag{4.15}$$

$$\text{Parameter updating step: } \phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|\,T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2 \tag{4.16}$$

Where, $\mathcal{D}_k$ is the number of trajectories that are taken into consideration from the reply buffer for updating the value network.

4.2.2 Action-advantage function estimation

The action-advantage function which is described in equation 4.12, measures whether or not the action is better or worse than the policy's default behavior [19]. In this study, GAE is performed over the TD-λ method. Where, TD-λ is referred to as temporal difference method [21]. The action-advantage function estimation for TD-λ method is described as,

$$\delta_t^V = R_t + \gamma V(s_{t+1}) - V(s_t) \tag{4.17}$$

Where, δ_t^V is referred as the estimation of action-advantage function for action a_t. TD-λ is a state-value function estimator where, λ is an weight parameter of the state-value function [21]. However the advantage function estimator is formulated for k-step as,

$$\hat{A}_t^{(k)} := \sum_{l=0}^{k-1} \gamma^l \delta_{t+l}^V = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) \tag{4.18}$$

Where, $\hat{A}_t^{(k)}$ is an estimator of the k-step discounted advantages [19]. The generalized advantage estimator GAE(γ, λ) is defined as the exponentially-weighted average of these k-step estimators [19].

$$\begin{aligned}\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &:= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(2)} + \lambda^2\hat{A}_t^{(3)} + \ldots\right) \\ &= \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^V\end{aligned} \tag{4.19}$$

Where, $\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}$ is denoted as GAE for action a_t. The generalized advantage estimator for $0 < \lambda < 1$ makes a compromise between bias and variance, controlled by the parameter λ [19]. Parameter λ and γ allows to trade-off between bias and variance. The γ determines the scalability of $V^{\pi,\gamma}$ hence controlling the variance, where the scalability does not depend on the λ. Moreover, $\lambda < 1$ introduces bias into the system, which is independent of the accuracy of the state-value function. on the other hand, the $\lambda < 1$ introduces bias only when the state-value function is inaccurate [19]. In this study, the values of parameter set as $\lambda = 0.97$ and $\gamma = 0.99$.

4.2.3 Policy optimization

The policy optimization is achieved by calculating the policy gradient using GAE. Hence, policy optimization imposes the minimization problem by updating the parameters of the policy network. In order to update the parameter of the policy network, the policy loss is computed and back-propagated in the network. Where, the policy optimization is formulated as,

$$\text{Policy loss}=\frac{1}{T}\sum_{t=0}^{T}\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_k}\left(a_t \mid s_t\right)}\hat{A}_t \tag{4.20}$$

Where, $\pi_\theta\left(a_t \mid s_t\right)$ is the log probability of the action a_t for state s_t by following policy π with parameter θ. In simple words, π_{θ_k} is the old policy (the old set of network parameters)

For better performance and faster convergence, the policy loss is clipped. Clipping of the policy loss controls that the new policy is not too far away from the old one. The clipping of policy loss is determined as,

$$L\left(s,a,\theta_k,\theta\right) = \min\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)}A^{\pi_{\theta_k}}(s,a),\quad g\left(\epsilon, A^{\pi_{\theta_k}}(s,a)\right)\right) \tag{4.21}$$

Where,

$$g(\epsilon,A) = \begin{cases}(1+\epsilon)A & A \geq 0\\ (1-\epsilon)A & A < 0\end{cases} \tag{4.22}$$

Where, $L\left(s,a,\theta_k,\theta\right)$ is the clipped policy loss. The ϵ is referred to as the clipping parameter and the value of the clipping parameter in this study is selected as $\epsilon = 0.1$. Hence, in this study the parameter is updated by back-propagating the clipped policy loss in the policy network. By back-propagating the clipped policy loss, the parameter of policy network θ is updated as $\theta_k \rightarrow \theta_{k+1}$ [20].

Parameter updating step:

$$\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k|\,T}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}\min\left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_k}\left(a_t \mid s_t\right)}A^{\pi_{\theta_k}}\left(s_t,a_t\right),\quad g\left(\epsilon, A^{\pi_{\theta_k}}\left(s_t,a_t\right)\right)\right) \tag{4.23}$$

The pseudo algorithm for PPO iteration is shown in figure 4.7.

Algorithm 1 PPO-Clip

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, ...$ **do**
3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4: Compute rewards-to-go $\hat{R}_t$.
5: Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based on the current value function V_{ϕ_k}.
6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.
7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.
8: **end for**

Figure 4.7: Psuedo algorithm for PPO[a]. The python code of PPO for this study is available at `github.com/darshan315/flow_past_cylinder_by_DRL`
[a] spinningup.openai.com/en/latest/algorithms/ppo.html

4.2.4 PPO iterations

The pseudo algorithm for PPO iterations is shown in figure 4.7. For iterations of the PPO algorithm, 10 workers are considered. The buffer size of the reply buffer is set to 10. Hence, 10 workers execute the 10 simulations at the same time. The weight for reward function is defined as $r_0 = 3, r_1 = 1, r_2 = 0.1, r_3 = 10^{-4}$ and $r_4 = 10^{-6}$. The weight r_3 and r_4 are set to small in order to limit the energy spending for the rotation of the cylinder. In this study, as the main objective is to reduce c_D and its fluctuations due to its higher magnitude compared to c_L, the weight for c_D is set to $r_1 = 1$ and the weight r_2 which is the weight of c_L is considered small 0.1. The higher value of r_2 also results in slower convergence for the PPO algorithm. In order to make the trajectory length equal, the total control time is set to $T = 5s$, where the control starts at $t = t_c$ and ends at $t = t_c + 5$.

Intuitively, for higher value of angular velocity, the drag is minimum. In this study, performing PPO iterations, it is observed that the agent is able to learn that physics, where policy network supplies higher values of mean and standard deviation of an action. The higher value of standard deviation may be understand as for any uncertain higher value of angular velocity, the drag is minimum. However, the higher value of angular velocity is energy inefficient and hence, not desirable. Moreover, for higher values of angular velocity, the simulation fails in OpenFOAM. Hence, the mean μ_a and standard deviation σ_a is also clamped. The values for clamping parameter for μ_a and σ_a is set as $\epsilon_\mu = 20$ and $\epsilon_\sigma = 7.39$ respectively.

$$\mu_a = c\left(\epsilon_\mu, \mu_a\right) \tag{4.24}$$

$$\sigma_a = c\left(\epsilon_\sigma, \sigma_a\right) \tag{4.25}$$

Where,

$$c(\epsilon, A) = \begin{cases} A & A \in [-\epsilon, \epsilon] \\ -\epsilon & A < 0 \text{ and } A \notin [-\epsilon, \epsilon] \\ \epsilon & A > 0 \text{ and } A \notin [-\epsilon, \epsilon] \end{cases} \tag{4.26}$$

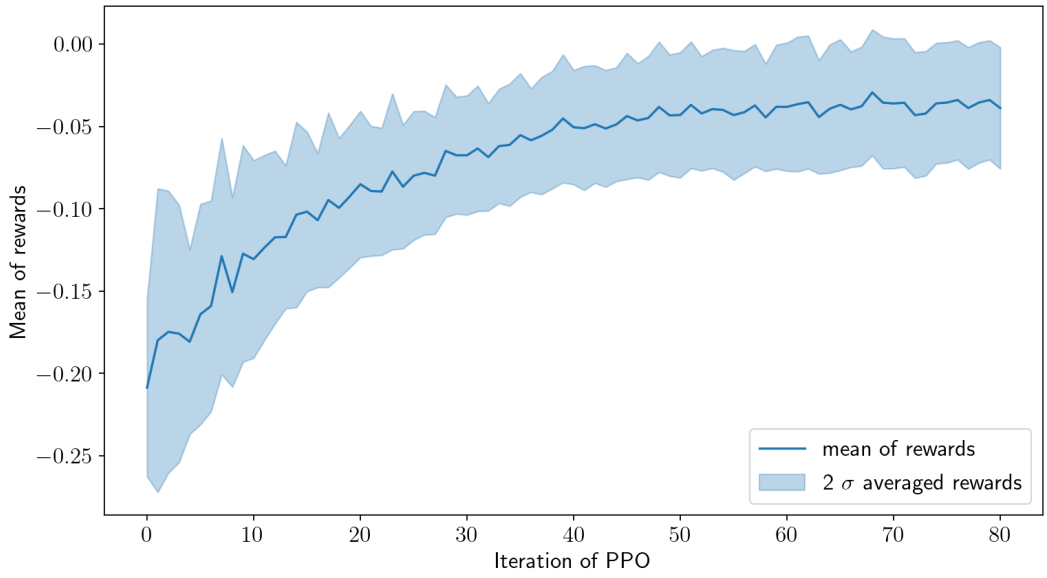

Figure 4.8: Mean of the cumulative rewards and 2 σ averaged rewards over a iterations of PPO algorithm.

The PPO algorithm is executed for 80 iterations. The average rewards and 2 σ averaged rewards at the end of the iteration is plotted in figure 4.8. From the figure, the learning converges after 60 iterations, and iterating further leads to the same results. Therefore, the policy does not improve after 80 iterations. The policy network after the 80 iterations is considered as optimal policy and the value network is considered as an optimal value network.

4.2.5 Results

To execute the simulation for optimal policy, the mean of the policy is only considered. When the state observations s_t are fed into the policy network, the mean of ω from the output layer is considered rather than sampling from the normal distribution. Hence for the rotation of the cylinder, the angular velocity is determined as,

$$\omega = \mu_a^{\pi_\theta^*} \tag{4.27}$$

Where, π_θ^* is the optimal policy with parameter θ. The simulation is executed for the optimal policy, where the DRL control starts at $t = 2.19s$. Meaningly, at $t = 2.19s$, the agent starts collecting state observation, and the cylinder is rotated by the agent in order to minimize the drag, lift, and their fluctuations. This simulation by following optimal policy in DRL is referred to as a DRL controlled case or closed-loop controlled case, and the simulation without closed-loop control is referred to as the uncontrolled case. The simulation of the closed-loop controlled case and simulation of the uncontrolled case is shown in figure 4.9. From the figure, the formation of the vortices in the wake occurs far from the cylinder in a closed-loop controlled case compared

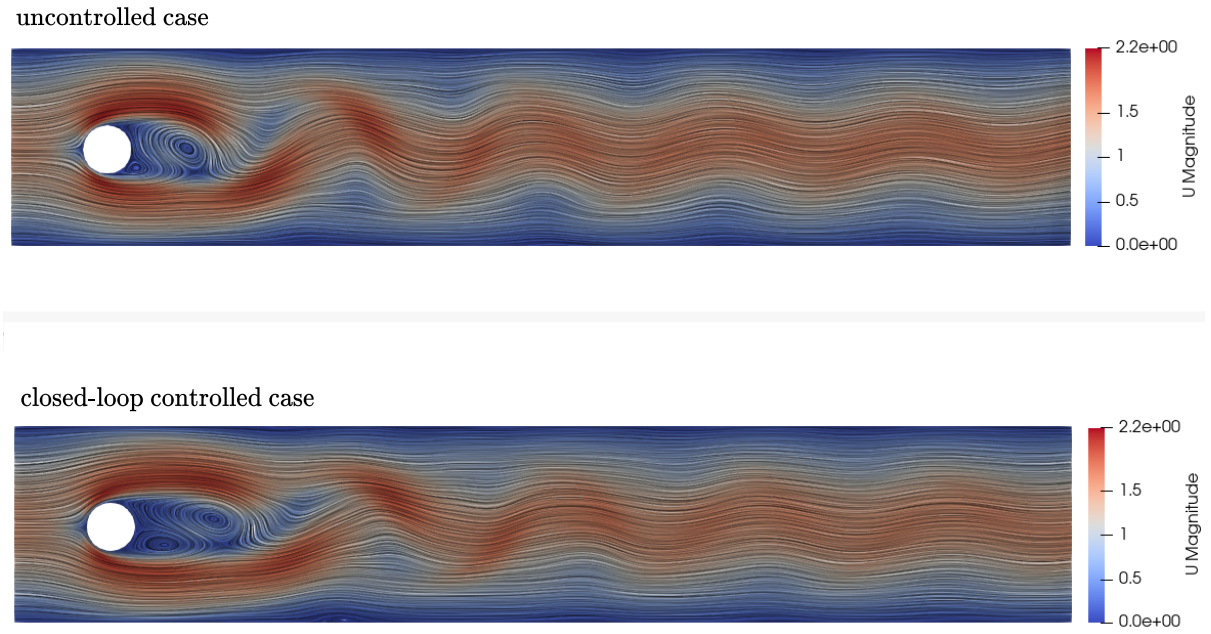

Figure 4.9: Velocity(magnitude) distribution at $t = 5.34s$ for closed-loop(DRL) controlled case and uncontrolled case. In closed-loop controlled case the control starts at $t = 2.19s$.

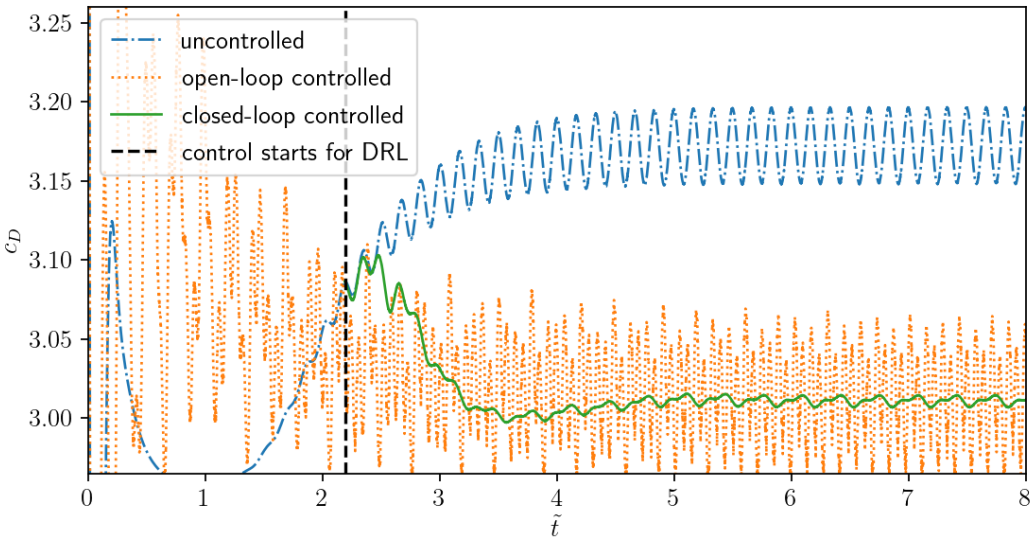

Figure 4.10: Values of c_D for closed-loop(DRL) controlled, open-loop controlled case and uncontrolled case, where in closed-loop controlled case cylinder is rotated by an agent which follows optimal policy, where the control starts at $t = 2.19s$. In open-loop controlled case the cylinder is oscillated by optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$.

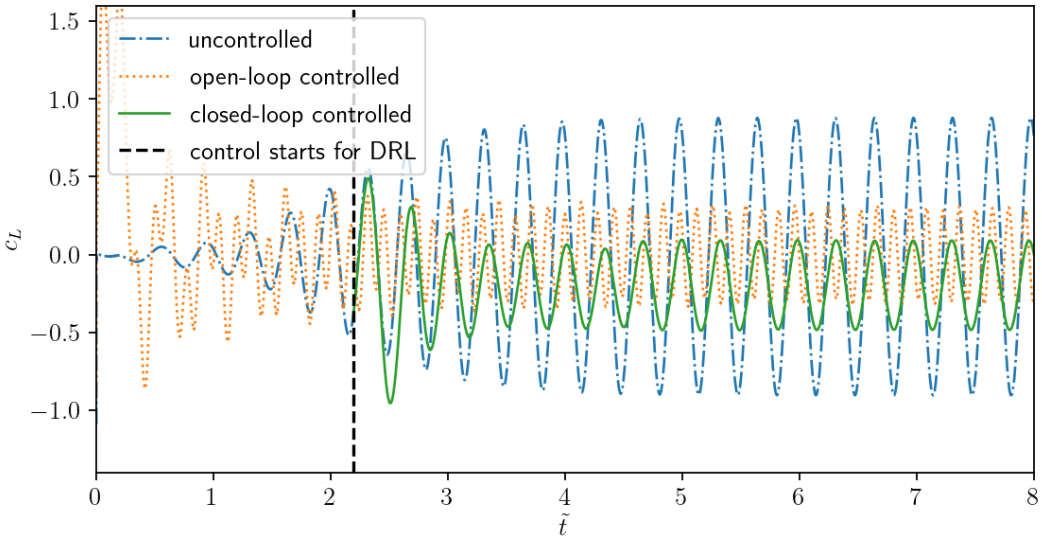

Figure 4.11: Values of c_L for closed-loop(DRL) controlled, open-loop controlled case and uncontrolled case, where in closed-loop controlled case cylinder is rotated by an agent which follows optimal policy, where the control starts at $t = 2.19s$. In open-loop controlled case the cylinder is oscillated by optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$.

case	$\overline{c_D}$	c_{Dmax}	c_{Dmin}	$\overline{c_L}$	c_{Lmax}	c_{Lmin}
uncontrolled case	3.1721	3.1965	3.1474	-0.0126	0.8789	-0.9049
open-loop controlled case	3.0256	3.0778	2.9809	0.0051	0.3262	-0.3231
closed-loop controlled case	3.0102	3.0153	3.0017	-0.1957	0.0956	-0.4897

Table 4.1: Table containing the mean, maximum, and minimum values of c_D and c_L for controlled case and open-loop controlled case and closed-loop controlled case, where the values are considered from $t = 4$ to $t = 8$.

case	$(c_{Dmax} - c_{Dmin})$	$(c_{Lmax} - c_{Lmin})$
uncontrolled case	0.0491	1.7838
open-loop controlled case	0.0969	0.6493
closed-loop controlled case	0.0136	0.5853

Table 4.2: Table containing the values of amplitude of fluctuations for controlled case and open-loop controlled case and closed-loop controlled case, where the values are considered from $t = 4$ to $t = 8$.

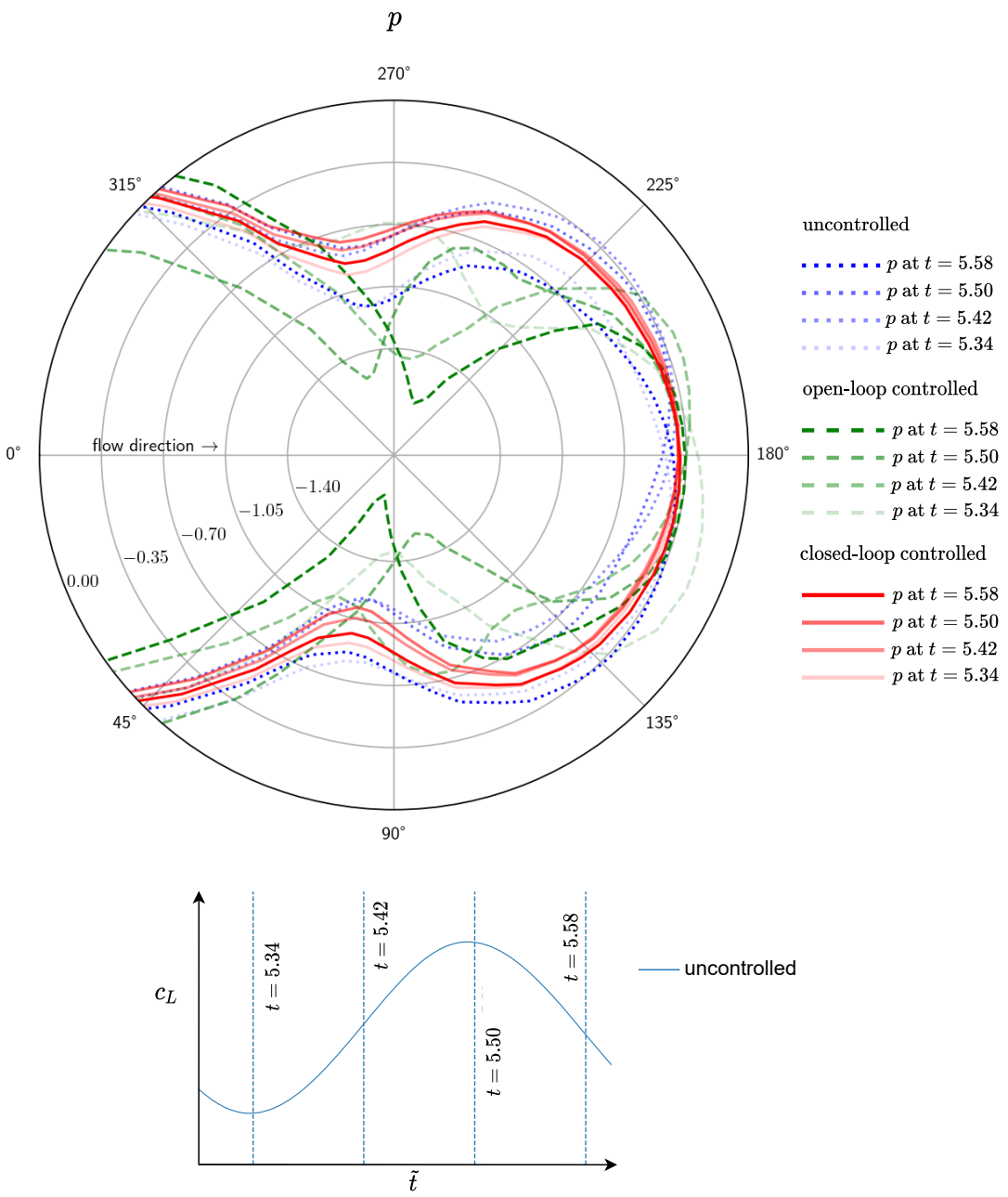

Figure 4.12: Pressure distribution over the polar coordinates at the surface of the cylinder for closed-loop(DRL) controlled, open-loop controlled case and uncontrolled case. In closed-loop controlled case cylinder is rotated by an agent which follows optimal policy, where the control starts at $t = 2.19s$. In open-loop controlled case the cylinder is oscillated by optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$.

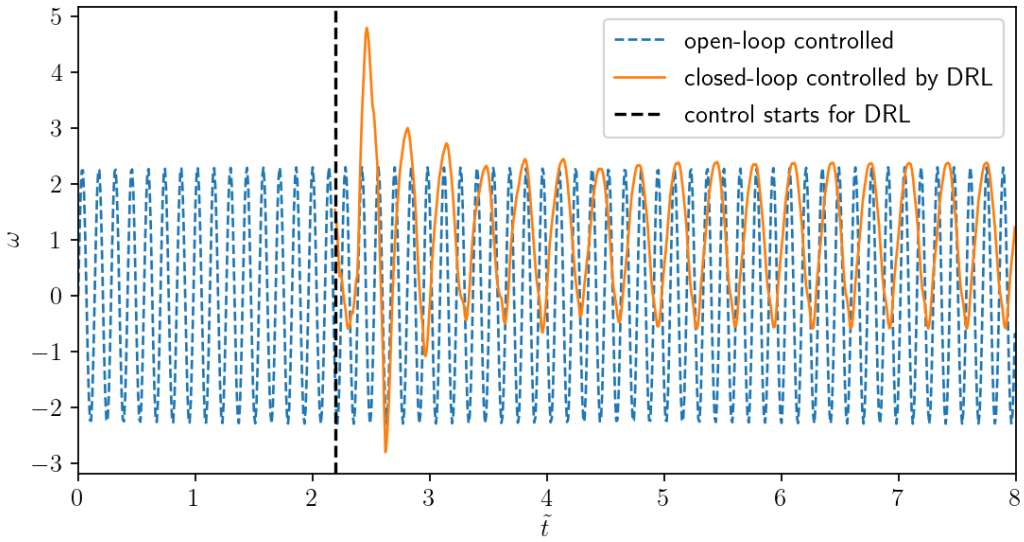

Figure 4.13: Angular velocity for closed-loop(DRL) controlled and open-loop controlled case.

to an uncontrolled case. The vortices propagation in the wake of the cylinder is weaker in closed-loop controlled case with comparing uncontrolled case.

In figure 4.12, the pressure distribution over the polar coordinates at the surface of cylinder is plotted. By using open-loop control strategy and closed-loop control strategy, the pressure at the cylinder surface is changed, which results reduction in drag and lift. The mean value of drag and lift is also improved by using closed-loop control, as the pressure fluctuation at the cylinder surface is decreased. Hence, in closed-loop control the agent learns to rotate the cylinder in order to decrease the pressure fluctuation at the cylinder surface.

The table 4.1 contains mean, maximum and minimum values of c_D and c_L. The amplitude of fluctuation of c_D and the amplitude of fluctuation of c_L are contained in table 4.2. In figure 4.10, the values of c_D is plotted for uncontrolled case, open-loop controlled case and closed-loop controlled case. From figure, the mean value of c_D is reduced with the significant reduction in the fluctuation of c_D compared to open-loop control. In closed-loop control, the frequency of the fluctuation is also significantly reduced compared to open-loop control. From table 4.1 and table 4.2, The mean value of drag is reduced by 5.10% and the amplitude of the fluctuation is damped by 72.30% by closed-loop control. However, in open-loop control the amplitude of the fluctuation of c_D is increased by 97.35%. Similarly, in figure 4.11, the values of c_L is plotted for uncontrolled case, open-loop controlled case and closed-loop controlled case. In closed-loop control the amplitude of fluctuation of c_L is decreased by 3.58% with compared to open-loop control. From figure 4.13, the relative change in the amplitude of angular velocity ω is 35.36% for closed-loop control compared to open-loop control. The ordinary frequency of rotation for closed-loop is also reduced by factor of 2. Hence, closed-loop control is energy efficient compared to open-loop control.

Chapter 5

Assessment

An assessment study provides information about the performance control strategy by changing the variable of the flow problem. The assessment determines the agent's ability to generalize for different flow conditions. In this study, for the assessment, the simulation is performed for $Re = 200$. For open-loop control strategy, the assessment is performed for optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$ which are achieved for $Re = 100$. Similarly, in the closed-loop strategy, the assessment is performed for optimal policy π^*_θ, which is obtained by $Re = 100$. The simulations assessment for the uncontrolled case, open-loop controlled case and closed-loop controlled case are shown in figure 5.1.

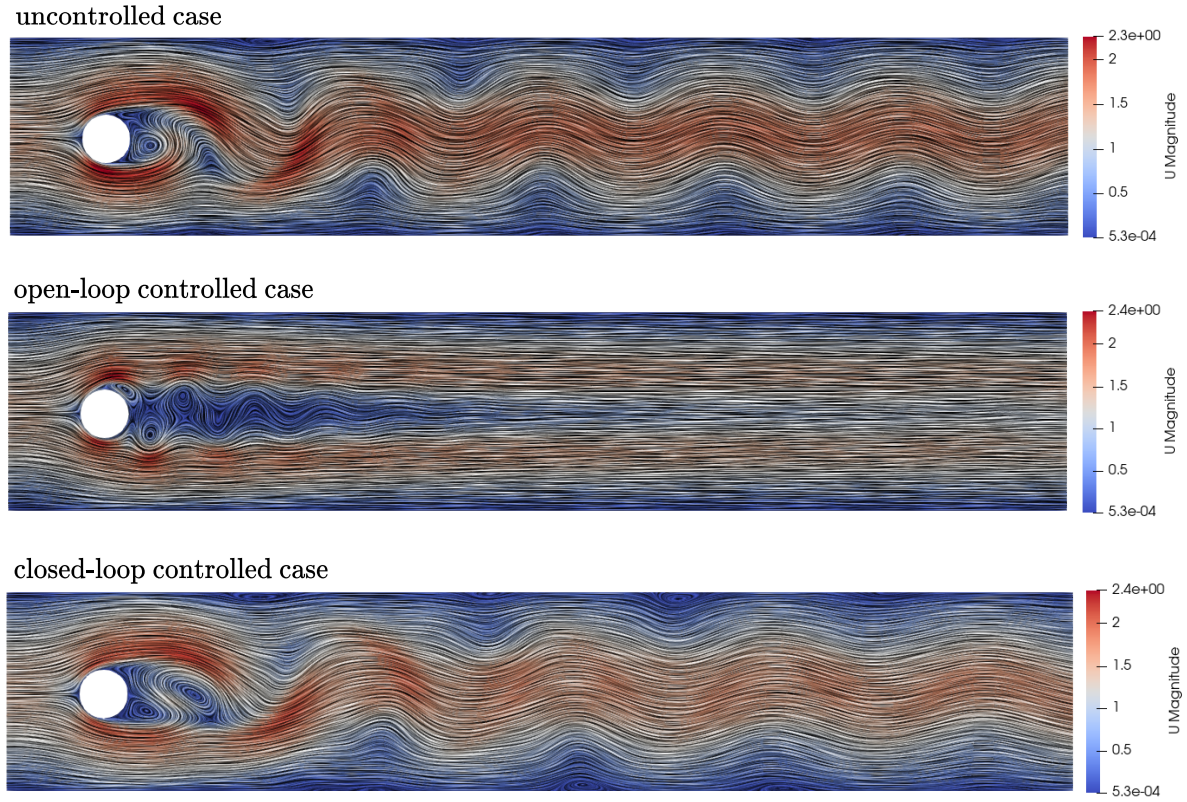

Figure 5.1: Velocity(magnitude) distribution at $t = 5.34s$ and $Re = 200$ for closed-loop(DRL) controlled case and uncontrolled case, where with control referred as closed-loop controlled case and without control referred as uncontrolled case. In closed-loop controlled case the control starts at $t = 2.19s$.

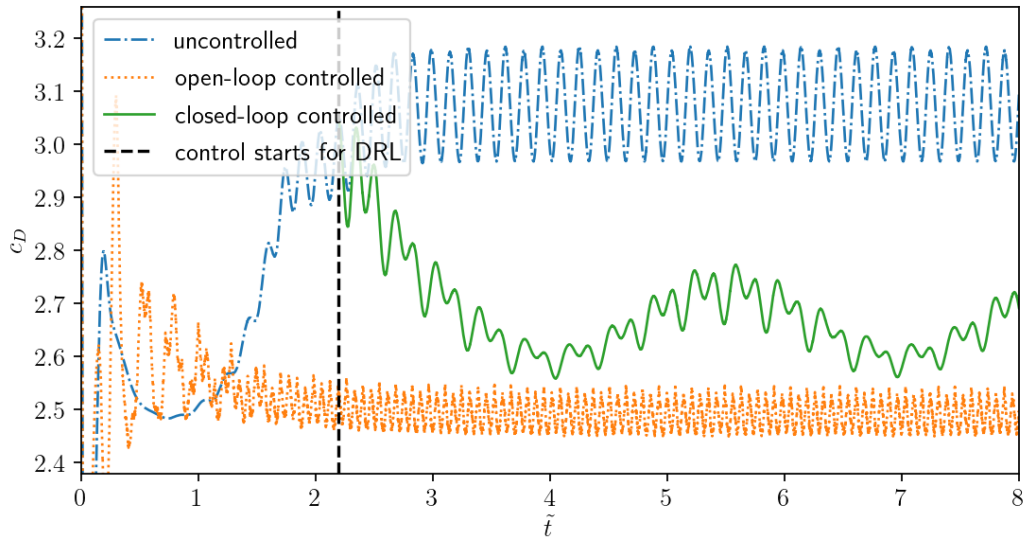

Figure 5.2: Values of c_D for closed-loop(DRL) controlled, open-loop controlled case and uncontrolled case for $Re = 200$, where in closed-loop controlled case cylinder is rotated by an agent which follows optimal policy, where the control starts at $t = 2.19s$. In open-loop controlled case the cylinder is oscillated by optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$.

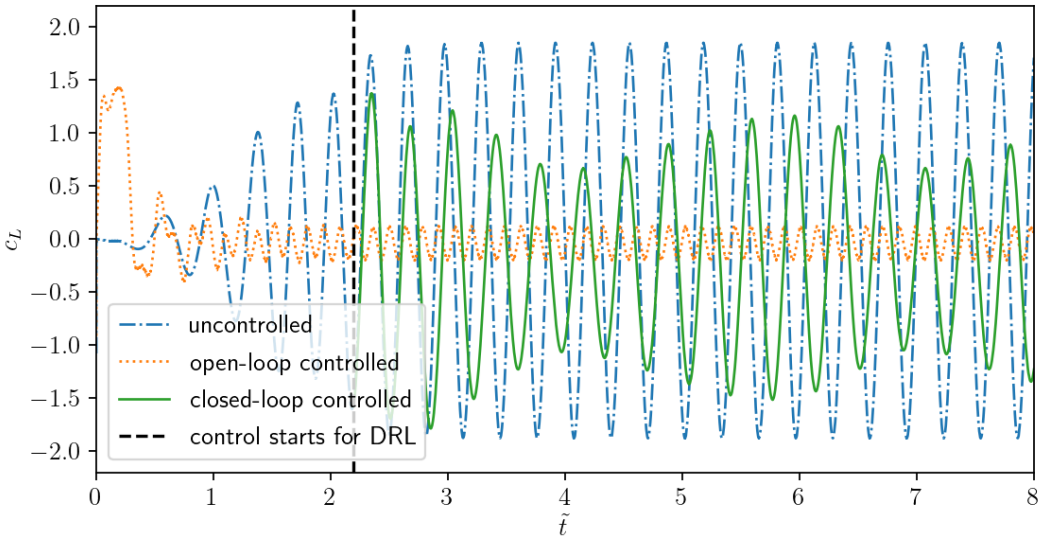

Figure 5.3: Values of c_L for closed-loop(DRL) controlled, open-loop controlled case and uncontrolled case for $Re = 200$, where in closed-loop controlled case cylinder is rotated by an agent which follows optimal policy, where the control starts at $t = 2.19s$. In open-loop controlled case the cylinder is oscillated by optimal parameter $\Omega = 2.294$ and $S_f = 0.7128$.

case	$\overline{c_D}$	c_{Dmax}	c_{Dmin}	$\overline{c_L}$	c_{Lmax}	c_{Lmin}
uncontrolled case	3.0737	3.1848	2.9681	-0.0436	1.8512	-1.8789
open-loop controlled case	2.4900	2.5459	2.4492	-0.0389	0.1213	-0.2016
closed-loop controlled case	2.6550	2.7729	2.5589	-0.2014	1.1644	-1.5159

Table 5.1: Table containing the mean, maximum, and minimum values of c_D and c_L for controlled case and open-loop controlled case and closed-loop controlled case for $Re = 200$, where the values are considered from $t = 4$ to $t = 8$.

case	$(c_{Dmax} - c_{Dmin})$	$(c_{Lmax} - c_{Lmin})$
uncontrolled case	0.2167	3.7301
open-loop controlled case	0.0967	0.3227
closed-loop controlled case	0.214	2.6803

Table 5.2: Table containing the values of amplitude of fluctuations for controlled case and open-loop controlled case and closed-loop controlled case for $Re = 200$), where the values are considered from $t = 4$ to $t = 8$.

From figure 5.2, the mean drag value is reduced for both open-loop controlled case and closed-loop controlled case. However, the frequency of fluctuation of c_D is increased for the open-loop control strategy. For open-loop control, the amplitude of fluctuation of c_L is damped significantly compared to other cases. On other hand, for a closed-loop controlled case, the agent is able to reduce $\bar{c_D}$ by 13.62% with significantly reducing amplitude and frequency of fluctuation of c_D. Moreover, The agent is also able to reduce $\bar{c_L}$ by 19.32% with a reduction in amplitude and frequency of fluctuation of c_D. The relative drag reduction achieved by the agent is higher at $Re = 200$ compared to $Re = 100$.

Chapter 6

Summary

When fluid flows across the object, due to the surface geometry and the shear stress between the surface of the object and boundary layer, the vortices form in the wake of the object. These vortices propagate in the wake of the object. These vortices are called von Kármán vortex street. The periodically propagating von Kármán vortex imposes periodic drag and lift forces on the object. Pressure fluctuations due to the surface curvature and shear stress between the boundary layer and boundary wall of the object. These periodic drag and lift forces are considered unwanted system dynamics in aircraft and automobiles. Hence, these unwanted system dynamics need to be controlled which imposes high dimensional optimization problem. As a generic example, flow across the 2D cylinder is considered in order to control the unwanted system dynamics in terms of drag, lift, and its fluctuations. The parabolic inflow boundary condition is considered. The fluid flows with Reynolds number 100. The flow problem is solved by using an OpenFOAM open-source solver. The meshing is achieved In OpenFOAM by creating the background mesh for the whole domain and subtracting the cylinder from it and the mesh around the cylinder is refined by region refinement in order to accurately capture the flow instability. The mesh dependency study is performed on a different level of mesh refinement 100, 200, and 400. The mesh refinement level 200 is considered as there is no significant change in drag and lift after the refinement level 200. The execution of the cylinder is performed parallel to minimize the computation time. The execution of the flow simulation provides information about drag, lift, and its fluctuations. The fluctuations and the mean value of drag are much higher than the fluctuations and the mean value of the lift. Hence, the primary objective is to control the drag and its fluctuations.

In open-loop control, the system is not subjected to feedback in order to evaluate the control strategy. In open-loop control, the control strategy is carried out and evaluated initially, and then it is fit in the system to achieve control. The evaluation of the open-loop control strategy occurs after the running of the system. For the open-loop control strategy, the cylinder wall boundary condition is subjected to the oscillatory sine wave function with parameter as amplitude A and f. The parameter A and f can be normalized as Ω and S_f. The optimal parameter Ω and S_f is determined by parameter study, where the cost function is formulated with the mean values of drag and lift along with its fluctuations. The range of parameter study is restricted for smaller values of Ω and S_f in order to limit the energy for rotating the cylinder. A fine sampling of Ω and S_f for parameter study results in high computational cost. A coarse sampling of Ω and S_f for parameter study results in inaccurate optimal parameters. To overcome this problem, the functional approximation is achieved on the coarse sampled data with the cost function, where the minimum value of the functional approximation yields the optimal parameters Ω and S_f.

In closed-loop flow control, the feedback from the system is utilized in order to evaluate the control strategy. The feedback is frequently analyzed to improve the control strategy in closed-

loop control during the running of the system. For closed-loop control, reinforcement learning is a proven state-of-the-art method. The environment in reinforcement learning is subjected to flow problems. The reinforcement learning problem is formulated as a finite horizon. The reward function is formulated as the converse of the cost function, where the cost function is formulated as drag, lift, and its fluctuations. The agent receives feedback from the environment in terms of state and rewards. The state of the environment is referred to as the pressure values at the cylinder surface. In reinforcement learning, the agent takes an action based on the state of the environment to maximize the reward function in order to achieve the objectives. The action taken by the agent is in form of rotating the cylinder. The agent determines the action by using a neural network(NN) architect from deep learning, which is called a policy network or actor. Therefore it is called deep reinforcement learning (DRL).

For DRL, the proximal policy optimization (PPO) algorithm is considered in this study. The PPO algorithm approximates state-value function and action-advantage value function. For PPO algorithm, the approximation of state-value function and action-advantage function is achieved by using value network and policy network respectively. The learning of a PPO algorithm is performed by PPO iterations. The optimal policy is obtained by the end of sufficient PPO iterations. For optimal policy, the policy network determines an action a_t based on the state of the environment s_t, such that the reward function $R_t$is maximum. In other words, optimal policy determines the angular velocity for the cylinder based on the pressure values at the cylinder surface in order to minimize the drag and lift along with its fluctuation.

The optimal parameters for the open-loop strategy are achieved as $\Omega = 2.294$ and $S_f = 0.7128$. For optimal parameters $\Omega = 2.294$ and $S_f = 0.7128$, the reduction in pressure gradient at the top and bottom of the cylinder surface results in a reduction in $\bar{c_D}$ by 4.61% and $\bar{c_L}$ by 0.56%. However, due to the high fluctuation of the pressure at the cylinder surface, the frequency of the fluctuation of c_D is increased and the amplitude of the fluctuation of c_D is increased by 97.35%. The c_{Dmax} and c_{Lmax} is reduced by 5.34% and 17.28% respectively for the open-loop controlled case. Similarly, the c_{Dmin} and c_{Lmin} is reduced by 3.86% and 18.48% respectively.

In closed-loop control, for the optimal policy, the $\bar{c_D}$ is decreased with significant damping its fluctuation. By optimal policy, the pressure fluctuation at the cylinder surface is significantly decreased, which results in the reduction of $\bar{c_D}$ and its fluctuation. The $\bar{c_D}$ is decreased by 5.10% with 72.30% reduction in amplitude of fluctuation of c_D. The $\bar{c_L}$ is also reduced by 5.77% and the amplitude of fluctuation of c_L is decreased 67.18%. The c_{Dmax} and c_{Lmax} is reduced by 5.66% and 24.50% respectively for the closed-loop controlled case. Similarly, the c_{Dmin} is reduced by 4.62% and c_{Lmin} is increased by and 18.48%.

The agent is assessed for $Re = 200$ in order to determine the agent's ability to generalize for different flow conditions. In the assessment, the agent is able to reduce $\bar{c_D}$ by 13.62% with significantly reducing amplitude and frequency of fluctuation of c_D and $\bar{c_L}$ by 19.32% with a reduction in amplitude and frequency of fluctuation of c_D. The assessment study suggests that the the relative drag reduction achieved by the agent is higher at $Re = 200$ and it has high potential to reduce the drag, lift, and its fluctuations.

The performance of an agent can be also improved by taking the sensible pressure values from the surface of the cylinder. At the top and bottom of the cylinder, where the pressure fluctuates more can be considered as the most sensible pressure values in the training of PPO. The pressure values, which do not fluctuate much can be avoided for better performance of the agent. Paris et al. [11] presented the study, which shows the sensitivity of the agent's learning ability with the number of pressure sensors placed in the wake of the cylinder and its distance from the cylinder. The performance of the agent can be also evaluated for the variable inflow conditions.

Bibliography

[1] Brandt Belson, Onofrio Semeraro, Clarence Rowley, and Dan Henningson. Feedback control of instabilities in the two-dimensional blasius boundary layer: The role of sensors and actuators. *Physics of Fluids*, 25, 05 2013.

[2] Charles-Henri Bruneau and Iraj Mortazavi. Passive control of incompressible bluff-body flows using porous media. *Comptes Rendus de l'Academie des Sciences Series IIB Mechanics*, 329, 01 2001.

[3] Haecheon Choi, Woo-Pyung Jeon, and Jinsung Kim. Control of flow over a bluff body. *Annual Review of Fluid Mechanics*, 40(1):113–139, 2008.

[4] N. Gautier, J.-L. Aider, T. Duriez, B. R. Noack, M. Segond, and M. Abel. Closed-loop separation control using machine learning. *Journal of Fluid Mechanics*, 770:442–457, 2015.

[5] R L Iman, J M Davenport, and D K Zeigler. Latin hypercube sampling (program user's guide). [lhc, in fortran].

[6] J. C. Lin, F. G. Howard, D. M. Bushnell, and G. V. Selby. Comparative study of control techniques for two-dimensional low-speed turbulent flow separation. pages 429–474, 1991.

[7] Sanjay Mittal and Bhaskar Kumar. Flow past a rotating cylinder. *Journal of Fluid Mechanics*, 476:303–334, 2003.

[8] Miguel Morales. *Grokking Deep Reinforcement Learning*, volume 1. Manning Publication, 1 edition.

[9] OpenFOAM: User Guide v2006 : Flow around a cylinder OpenFOAM documentations. `https://openfoam.com/documentation/tutorial-guide/tutorialse3.php`.

[10] OpenFOAM: User Guide v2006 : Pressure-velocity algorithms OpenFOAM documentations. `https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-pressure-velocity-intro.html`.

[11] Romain Paris, Samir Beneddine, and Julien Dandois. Robust flow control and optimal sensor placement using deep reinforcement learning. *Journal of Fluid Mechanics*, 913, 04 2021.

[12] Mark Pastoor, Lars Henning, Bernd R. Noack, Rudibert King, and Gilead Tadmor. Feedback shear layer control for bluff body drag reduction. *Journal of Fluid Mechanics*, 608:161–196, 2008.

[13] Jean Rabault, Miroslav Kuchta, Atle Jensen, Ulysse Réglade, and Nicolas Cerardi. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865:281–302, 2019.

[14] Jean Rabault and Alexander Kuhnle. Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach. *Physics of Fluids*, 31:094105, 09 2019.

[15] Jean Rabault, Feng Ren, Wei Zhang, Hui Tang, and Hui Xu. Deep reinforcement learning in fluid mechanics: a promising method for both active flow control and shape optimization, 2020.

[16] R. Radespiel. Vorlesungsmanuskript: Strömungsmechanik. Technical report, Institut für Strömungsmechanik, TU Braunschweig, 2008.

[17] M. Schäfer and S. Turek. Benchmark Computations of Laminar Flow Around a Cylinder. *Springer Publishing*, 1996.

[18] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, Jan 2015.

[19] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[21] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning*, volume 1. The MIT Press, Massachusetts, 2 edition, 2015.

[22] Hongwei Tang, Jean Rabault, Alexander Kuhnle, Yan Wang, and Tongguang Wang. Robust active flow control over a range of reynolds numbers using an artificial neural network trained through deep reinforcement learning. *Physics of Fluids*, 32(5):053605, May 2020.

[23] Mikhail Tokarev, Egor Palkin, and Rustam Mullyadzhanov. Deep reinforcement learning control of cylinder flow using rotary oscillations at low reynolds number. *Energies*, 13(22), 2020.

[24] P. T. Tokumaru and P. E. Dimotakis. Rotary oscillation control of a cylinder wake. *Journal of Fluid Mechanics*, 224(1):77–90, 1991.

[25] P.R Viswanath. Aircraft viscous drag reduction using riblets. *Progress in Aerospace Sciences*, 38(6):571–600, 2002.

[26] C H K Williamson. Vortex dynamics in the cylinder wake. *Annual Review of Fluid Mechanics*, 28(1):477–539, 1996.

[27] Kianoosh Yousefi, Reza Saleh, and Peyman Zahedi. Numerical study of blowing and suction slot geometry optimization on naca 0012 airfoil. *Journal of Mechanical Science and Technology*, 28:1297–1310, 05 2014.

List of Figures

List of Tables