SOLVING NAVIER STOKES EQUATIONS WITH PHYSICS INFORMED NEURAL NETWORK FOR CALCULATION OF AERODYNAMIC FORCES

by
SILA AKPINAR

Submitted to the Graduate School of Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Science

Sabancı University
July 2022

SOLVING NAVIER STOKES EQUATIONS WITH PHYSICS INFORMED NEURAL NETWORK FOR CALCULATION OF AERODYNAMIC FORCES

Approved by:

Prof. Serhat Yeşilyurt .....
(Thesis Supervisor)

Assoc. Prof. Kamer Kaya .....

Assoc. Prof. Ahmet Fatih Tabak .....

Date of Approval: July 27, 2022

Sila Akpinar 2022 ©

All Rights Reserved

ABSTRACT

SOLVING NAVIER STOKES EQUATIONS WITH PHYSICS INFORMED NEURAL NETWORK FOR CALCULATION OF AERODYNAMIC FORCES

SILA AKPINAR

Mechatronics Engineering, Master's Thesis, July 2022

Thesis Supervisor: Prof. Serhat Yeşilyurt

Keywords: scientific machine learning, physics-informed machine learning, fluid dynamics, aerodynamics, laminar flow, turbulence

Understanding of flow dynamics is crucial in a comprehensive set of scientific disciplines, such as astrophysics, chemistry, biology, meteorology, biomedical engineering, and mechanical engineering. Nevertheless, fluid dynamics properties cannot be well-understood in a case of complex geometry, high Mach number flow, turbulence, stall, and complex reactions. Experiments can provide some insights to study these complicated phenomena. Yet, certain information may not be obtained accurately because of low fidelity and experimental limitations. On the other hand, Navier-Stokes as a governing equation of viscous flow of an incompressible fluid can be solved numerically to obtain flow properties. However, such numerical analysis relies heavily on computational power which requires long duration to conclude and modelling ability.

As an alternative approach, we deal with this problem by implementing a physics-informed neural network (PINN). As a scientific machine learning algorithm, PINNs are developed to solve partial differential equations approximately. In this thesis, we first implement PINNs for solving Navier-Stokes equations for laminar flow over a cylinder. Then, we apply PINN for turbulent flow over a stationary NACA0018 airfoil with a high angle of attack. We implement the PINN approach with sparse data from the numerical CFD study. Our results reveal that the PINN is able to recover missing data with excellent accuracy for both laminar and turbulent flow problems. The PINN model is also used to calculate aerodynamic forces acting on the cylinder and on the airfoil. For force calculations, two different methods are applied to find the optimum application with less error from the PINN approach. Results show that gradient-based stress integration method ends up with more accurate results than integral-based control volume approach.

ÖZET

AERODİNAMİK KUVVETLERİ HESAPLAMAK İÇİN NAVIER-STOKES DENKLEMLERİNİN FİZİK BİLGİLİ NÖRAL AĞ İLE ÇÖZÜMÜ

SILA AKPINAR

Mekatronik Mühendisliği, Yüksek Lisans Tezi, Temmuz 2022

Tez Danışmanı: Prof. Dr. Serhat Yeşilyurt

Anahtar Kelimeler: bilimsel makine öğrenmesi, fizik entegre edilmiş makine öğrenmesi, akışkanlar dinamiği, aerodinamik, laminer akış, türbülans

Akış dinamiğinin anlaşılması astrofizik, kimya, biyoloji, meteoroloji, biyomedikal mühendisliği ve makine mühendisliği gibi kapsamlı bir dizi bilimsel disiplinde çok önemlidir. Bunun yanı sıra, karmaşık geometri, yüksek Mach sayılı akış, türbülans, tutunma kaybı ve karmaşık reaksiyonlar durumunda akışkanlar dinamiği özellikleri iyi anlaşılamaz. Deneysel çalışmalar, bu karmaşık akış davranışlarını incelemek için bazı bilgiler sağlayabilir. Ancak, düşük doğruluk ve deneysel sınırlamalar nedeniyle bazı bilgiler doğru olarak elde edilemeyebilir. Öte yandan, Navier-Stokes sıkıştırılamaz bir akışkanın viskoz akışını yöneten bir denklem olarak akış özelliklerini elde etmek için sayısal olarak çözülebilir. Bu tür sayısal analizler, sonuçlandırmak için uzun süre ve modelleme yeteneği gerektiren hesaplama gücüne büyük ölçüde ihtiyaç duyar.

Alternatif bir yaklaşım olarak, bu sorunu fizik bilgili nöral ağ uygulayarak ele alıyoruz. Bilimsel bir makine öğrenme algoritması olarak, fizik bilgili nöral ağ, kısmi diferansiyel denklemleri yaklaşık olarak çözmek için geliştirilmiştir. Bu tezde, ilk önce, Navier-Stokes denklemlerini silindir üzerindeki laminer akış için fizik bilgili nöral ağ ile çözdük. Ardından, yüksek hücum açısına sahip sabit NACA0018 kanat profili üzerinde türbülanslı akışı çözebilmek için fizik biligili nöral ağ uyguladık. Algoritmayı sayısal hesaplamalı akışkanlar dinamiği çalışmasından elde edilen seyrek verilere uyguladık. Sonuçlarımız, fizik bilgili nöral ağ modelinin hem laminer hem de türbülanslı akış problemleri için mükemmel bir doğrulukla eksik verileri kurtarabildiğini ortaya koymaktadır. Modeli ayrıca silindir ve kanat üzerine etki eden aerodinamik kuvvetleri hesaplamak için de kullandık. Kuvvet hesaplamaları için, iki farklı yöntem uyguladık. Sonuçlar, eğim tabanlı stres integrali yönteminin, integral tabanlı kontrol hacmi yaklaşımına göre daha doğru çalıştığını göstermektedir.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Serhat Yeşilyurt for his patience, invaluable guidance, and for always being supportive of me. It was a privilege to work under his supervision and I am honored that I had this opportunity. Also, I would like to express my deepest thanks to Assoc. Prof. Kamer Kaya. His help and comments contributed essentially to the completion of this thesis. In addition, I would like to thank Assoc. Prof. Ahmet Fatih Tabak for his careful evaluation of my thesis and useful remarks.

I thank Sahar Dadashi Farkhandi for her funniest friendship, Asal Saeidfar, Fatemeh Malekabadi, Saina Farrokhpour Sani, and Mervenaz Şahin for being the best lab mates. I am grateful to Alperen Kenan, Bilal Çatkın, Çağatay Irmak, Harun Tolasa, Melike Cezayirlioğlu, Ömer Burak Aladağ, Selim Ahmet İz, and Özgür Taylan Kenanoğlu for their invaluable friendship and for making my graduate life more enjoyable. My special thanks go to Celal Umut Kenanoğlu for motivating me whenever I feel overwhelmed, for his all jokes (even for very bad ones), and his sincerest friendship.

Last but not least, I would like to express my warmest thanks to my family, my beloved twin sister Çağla Akpınar, my greatest companion Derin Karadeniz, and my favorite person Atakan Güven. They are the source of happiness in my life and I am forever indebted to have their endless love and support with me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Approach in this thesis, and necessary background are summarized here; solution methods of partial differential equations in the context of fluid dynamics with relatively newly proposed machine learning techniques are described. Especially, background is provided for application of machine learning methods in fluid dynamics. The discussion is followed by the motivation of the thesis and the outline of the thesis.

1.1 Literature Review

Real-life problems are modeled mathematically for better understanding and developing solutions. In physics and many other disciplines, solving partial differential equations (PDEs) is key to simulating the problem domain. Some important PDEs are Laplace's Equation, Poisson's, heat equation and the wave equation. For decades, PDEs have been approximated by numerical methods such as the finite difference method, the finite element method, and the finite volume method[3]. Application of those techniques is becoming challenging by the curse of dimensionality that leads to exponential growth in computational operations and the lack of a mathematical theory for turbulent flow [4]. Besides, some assumptions and specific conditions are needed for numerical solving, that is tough for complex geometries.

Apart from conventional techniques, there has been growing interest in deep learning and machine learning algorithms to find surrogate models for physical problems and tackle PDEs in fluid mechanics. Data-driven artificial neural network studies have focused on fluid flow [5] and turbulence modeling [6, 7] due to the accumulation of a huge amount of data from numerical simulations and experiments. Similarly, the closure term of the Spalart-Allmaras turbulence model has been rebuilt by a supervised learning algorithm [8]. A convolutional neural network (CNN) that makes

use of shared-encoding and decoding has been studied to predict velocity and pressure fields around an airfoil by information extraction from inputs of the Reynolds number, angle of attack and the airfoil geometry [9]. PDE functional identification of nonlinear dynamics (PDE-FIND) algorithm has been developed to factor in spatial derivatives in exploring Navier Stokes equations (NSEs) as a data-driven tool [10]. Deep neural networks (DNNs) have been constructed for discovering turbulent shear stress [11], observing flow in heterogeneous media [12], and Reynolds stress prediction of channel flow [13]. Those techniques perform accurately and relatively efficiently in computation. However, the training process requires a huge collection of data. Acquisition of the data can be challenging from experiments, or it can be computationally expensive from numerical analysis. In this study, Navier Stoke Equations (NSEs) and continuity equations are approximated by physics-informed neural networks (PINNs) [14, 15, 16]. PINNs allow to create a model in absence of sufficient data to investigate fluid velocity and pressure field for different types of problems such as laminar flow over a cylinder, and turbulent flow around a stationary airfoil with a high angle of attack.

In the late nineties, multi-layer perceptron (MLP) consisting of constraint loss function for problems governed by either ODE or PDE has been studied [17]. More recently, physics-informed neural networks (PINNs) have been introduced to solve nonlinear PDEs such as the advection-diffusion equation, Burger's equation, and Korteweg-de Vries (KdV) equation by Raissi et. al in two parts [14, 15]. Physics informed neural network is a tool that combines scientific computing that focuses on differential equations and machine learning. The approach is applicable for both forward and inverse type of problems. In a forward problem, model approximates solution of PDE based governing equations. In an inverse problem, physical quantities of the problem are extracted from the data. NSEs in both velocity-pressure form and vorticity-velocity form have been solved for laminar and turbulence flows via PINNs, specifically called as NSFnet (Navier-Stokes flow nets) [18]. In order to ensure low numerical stiffness in high Reynolds number flow regimes, NSFnet has been expanded [19] with a learning rate algorithm that utilizes gradient statistics [20].

Hidden fluid mechanics (HFM) that encodes Navier Stokes and conservation laws, i.e., continuity, momentum, and energy, to deduce hidden velocity and pressure quantities from passive scalar visualization is presented by Raissi et al. [16]. In addition to being agnostic of geometry, the algorithm is also independent of the initial and boundary conditions, which allows feasibility in the problem domain to be chosen. In [16], HFM has been used to predict information such as aerodynamic forces and wall shear stresses in arteries where measurement is almost impossible.

The aerodynamic force predictive capacity of the CNN-based deep learning model, that is fed by force coefficients using a full-order NS solver, has been demonstrated for several types of bluff bodies [21]. Similarly, several CNN with competitive results compared to MLP have been performed for lift force coefficient prediction of different airfoil geometries in a diverse flow characteristic [22].

1.2 Motivation

Scientific machine learning (SciML) is a discipline that is looking for new innovative techniques to deal with scientific data sets [23]. It benefits from machine learning and scientific computing. Machine learning data-driven models can be perfectly fitted on training data; however, model may not be able to extract physically meaningful predictions. Therefore, there is a need to teach physical laws and domain knowledge to the machine learning model. It can be considered as informative priors that are hard theoretical constraints. Observational, empirical, or physical understanding of knowledge as a prior information can be advantageous to improving the training performance of the network [24].

Generally, useful data is usually very limited for data-driven techniques as it is not practical to obtain experimental data from complex systems. On the other hand, while it is possible to have a huge amount of data, for example in the form of continuous time series, governing equations behind the problem may not be known. In most cases at best, experimental data is sparse, noisy and only partially known when the physics based model is available. It is possible to infer missing or even hidden quantities by recovering the solution of partial differential equation based governing equations behind the problem from the sparse data set and partially known physics [24].

Physics-informed Neural Networks (PINNs) [14] are an example of SciML that reflects the integration of data and governing equations of the physics that is driven by generally partial differential equations. These machine learning methods can stay robust by having governing equations as hard constraints on even imperfect data, i.e., missing data, noisy data, or data with having outliers [24]. Such a data-efficient physics-informed machine learning algorithm occurs as an alternative approach to conventional numerical solvers and makes possible to deal with several problems in computational science.

In this thesis, PINNs are applied to two-dimensional fluid flow problems that include laminar and turbulent flows to obtain drag and lift forces on objects inside the flow. For the laminar flow problem, flow around a cylinder is investigated by visualization

of passive scalar, similar to [16]. Our study expands on the work of Raissi's [16] that extracted lift and drag forces by taking integral of stress field over the object. On the other hand, we highlight the application of the integral approach, i.e., control volume approach to extract aerodynamic forces. Furthermore, turbulent flow is considered as a second problem on a stationary airfoil with a high angle of attack. PINN model for turbulent flow problems has only information on eddy viscosity and y component of the fluid velocity. The pressure field and x component of the velocity are inferred from the Navier Stokes and continuity equation. The main objective behind the study is to explore the capability of PINN that is an immature algorithm over different fluid flow motions. Discovering the feasibility and limitations of PINN can lead to combining the power of machine learning and numerical methods in future studies.

1.3 Thesis Outline

In Chapter 2, background information from aerodynamics and PINN is provided. In the aerodynamics part, fundamental fluid properties, physical laws that govern fluid dynamics problems, aerodynamic forces, airfoil characteristics, and stall phenomena are mentioned briefly. Under the physics-informed neural network section, algorithm structure is explained.

Chapter 3 presents an investigation of the laminar flow around a cylinder. For this purpose, firstly, a 2D time-dependent numerical CFD study is developed by COMSOL. Synthetic sparse data set is created from a numerical solution for the PINN model, that is built to recover missing data and further used to extract hidden unknown quantities from the governing equations.

In Chapter 4, the study is extended to turbulent flow. As a first step, a 2D time-dependent turbulence model is conducted for modeling the flow around a stationary airfoil. Then, sparse data set is obtained from numerical solution for PINN model that recovers the missing data and infer hidden quantities from the governing equations.

In Chapter 5, the conclusion of the thesis is discussed and the direction to future studies is proposed.

Chapter 2

Background

The second chapter provides background information for aerodynamics and physics-informed neural networks for a better understanding of thesis motivation and methodology. In the first part, aerodynamics fundamentals along with flow properties and physical laws are discussed to describe fluid motion. The importance of aerodynamic forces and a couple of calculation ways are explained. Lastly, airfoil geometry, characteristics, and aerodynamics are expressed. In the second part, physics-informed neural networks are expressed in detail.

2.1 Aerodynamics

2.1.1 Fundamentals

Aerodynamics is addressed the motion of air, especially when the airfoil interacts with a solid object. One of the main concerns is the prediction of aerodynamic forces, moments, and heat transfer rate over the object. The flow pattern around the object should be obtained accurately in order to achieve useful results for practical applications. Fluid behavior depends on geometry, position with respect to the free stream, altitude, and speed of the object. In general, some assumptions on fluid properties are needed for analysing flow types [2].

For studying different flow types, some fluid property assumptions are typically required. For instance, since the temperature gradient is so small in some cases, the temperature is ineffective in the flow field. Density is typically taken to be constant when temperature change is neglected. However, since density depends on both temperature and pressure, such an assumption is invalid for high-speed flows [2]. Another example is the continuum assumption. In contrast to liquids and solids, gases are made of discrete molecules. Therefore, gas atoms occupy the volume

but with a relatively low percentage. The sparse distribution of gas molecules is neglected in many aerodynamics studies to assume the continuum behavior of the flow when the characteristic length is sufficiently greater than the mean-free path between the collision of gas molecules [25].

Physical laws are used to describe the fluid motion. Fundamental physical laws are conservation of mass, conservation of momentum, and conservation of energy. They are also known as a continuity equation, Newton's second law of motion, and the first law of thermodynamics respectively. For the solution of complex fluid behaviors, generally, theoretical studies rely on some key assumptions that are valid under specific circumstances.

The continuity equation indicates that mass is conserved and provided in the partial differential form as follows

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) + \frac{\partial}{\partial z}(\rho w) = 0 \tag{2.1}$$

The above equation is valid for all steady/unsteady, viscous/inviscid, compressible/incompressible flow characteristics. It forms for a steady flow. For an incompressible flow where a change in the density is neglected, the continuity equation is simplified to

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{2.2}$$

Conservation of linear momentum is written in the closed as shown in Equation 2.3

$$\rho\vec{g} - \nabla p + \nabla \cdot \tau_{ij} = \rho\frac{d\vec{V}}{dt} \tag{2.3}$$

where terms represent gravity force, pressure force, and viscous force per unit volume respectively on the left-hand side, and density times acceleration on the right-hand side for the velocity vector, $V = [u, v, w]$. If a frictionless flow is assumed, then τ_{ij} is dropped which ends up with Euler's equation for inviscid flow. On the other hand, viscous term τ_{ij} is expressed in terms of element strain rate and viscosity for Newtonian fluids. For an incompressible, Newtonian fluid with constant density and viscosity Equation takes the following form in three-dimensional space, which is known as Navier-Stokes equations [26].

$$\begin{aligned}
\rho g_x - \frac{\partial p}{\partial x} + \mu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}) &= \rho \frac{Du}{Dt} \\
\rho g_y - \frac{\partial p}{\partial y} + \mu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}) &= \rho \frac{Dv}{Dt} \\
\rho g_z - \frac{\partial p}{\partial z} + \mu(\frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2}) &= \rho \frac{Dw}{Dt}
\end{aligned} \tag{2.4}$$

where $\frac{D}{Dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z}$ is the material derivative. Lastly, the differential energy equation for a Newtonian, compressible, unsteady, viscous, and heat-conducting flow is formulated below [26].

$$\rho C_p \frac{Du}{Dt} + p(\nabla \cdot V) = \nabla \cdot (k \nabla T) + \Phi \tag{2.5}$$

where T is temperature, k is thermal conductivity and Φ is the viscous heat generation.

2.1.2 Aerodynamic Forces

When a solid object is placed inside a fluid, interaction forces acting on the object consist of shear stress and normal stress terms. The force which is in the same direction as upstream velocity is called drag force. The force which is perpendicular to the upstream velocity is called drag force [27]. Two methods are explained to calculate the resultant force here. One is from the integration of the stress field over the object. Another one is from the Reynolds transport theorem by applying the conservation of linear momentum on a control volume.

Method 1: Integration of stress field over the solid object

Total stress tensor is provided as follows:

$$\sigma_{ij} = -p\delta_{ij} + \tau_{ij} \tag{2.6}$$

where p is static pressure, τ is viscous stress tensor, and δ is kronecker delta. Total stress tensor terms are expressed separately below.

$$\tau_{ij} = \mu(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}) \tag{2.7}$$

where μ is dynamic viscosity and $\mu = 1/Re = 1/200$.

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & otherwise \end{cases} \tag{2.8}$$

Forces acting on the object can be obtained from the integration of the stress field over the surface area. Assume that upstream velocity is provided in the x-direction. Since drag force occurs in parallel to upstream velocity, it is calculated from the integration of the stress field in the x-direction as shown in Equation 2.9 where n_x is x-component of the normal vector and n_y is y-component of the normal vector. Similarly, since lift force occurs perpendicular to the upstream velocity, it is calculated from the integration of the stress field in the y-direction as shown in Equation 2.10.

$$F_D = \int (-pn_x + 2\,\mu \frac{\partial u}{\partial x}\,n_x + \mu\,(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})\,n_y)\,dA \tag{2.9}$$

$$F_L = \int (-pn_y + 2\,\mu \frac{\partial v}{\partial y}\,n_y + \mu\,(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})\,n_x)\,dA \tag{2.10}$$

Method 2: Control volume approach

Reynolds Transport Theorem (RTT) is applied when system analysis is desired to change by control volume analysis. For this purpose, RTT makes the transformation of mathematical formulation based on specific regions rather than individual masses. RTT application is changed according to control volume structure such as fixed, moving, or deformable boundaries. For a fixed control volume, RTT is written in the compact form as follows [26]:

$$\frac{D}{Dt}(\mathcal{B}_{sys}) = \frac{d}{dt}(\int_{CV} \mathcal{B}\rho\,d\mathcal{V}) + \int_{CS} \mathcal{B}\rho\,(V \cdot n)\,dA \tag{2.11}$$

where $\mathcal{B}$ can be any fluid property, β is an intensive value of $\mathcal{B}$ per unit mass, i.e., $d\mathcal{B}/dm$, ρ is density, V is velocity, n is outward unit normal vector, dA is differential area, CS is control surface, and CV is control volume. Plugging mass, momentum, or energy as a $\mathcal{B}$ property allows for the expression of physical laws such as conservation of mass, linear momentum, and energy respectively. When $\mathcal{B}$ represents linear momentum, i.e., $\mathcal{B} = mV$, conservation of linear momentum for a fixed control volume is indicated as below.

$$\sum F = \frac{d}{dt}(\int_{CV} V\rho\,d\mathcal{V}) + \int_{CS} V\rho(V \cdot n)\,dA \tag{2.12}$$

Equation 2.12 is a vector equation since all terms include V which represents the

velocity vector. $\sum F$ is the vector summation of forces appearing on the object. In order to calculate lift and drag forces acting on the object in the control volume, x and y components of Equation 2.12 should be written while considering the pressure and viscous forces as shown below.

$$\begin{aligned} F_D = \frac{d}{dt}(\int_{CV} u\rho \, d\mathcal{V}) + \int_{CS} u\rho(un_x + vn_y)dA \\ + \int_{CS}(pn_x)dA - \int_{CS}[(2\mu\frac{\partial u}{\partial x}n_x) + (\mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})n_y)]dA \end{aligned} \tag{2.13}$$

$$\begin{aligned} F_L = \frac{d}{dt}(\int_{CV} v\rho \, d\mathcal{V}) + \int_{CS} v\rho(un_x + vn_y)dA \\ + \int_{CS}(pn_y)dA - \int_{CS}[(2\mu\frac{\partial v}{\partial y}n_y) + (\mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})n_x)]dA \end{aligned} \tag{2.14}$$

2.1.3 Airfoil Characteristics and Aerodynamics

The lift and stall characteristics of an airfoil are affected by its geometry. The most essential geometric characteristics are leading-edge radius, which affects boundary-layer separation, the mean camber line, maximum thickness, and thickness distribution. Airfoil geometric terminology is illustrated in Figure 2.1. A chord line is a straight line that connects the leading and trailing edges of the airfoil, the corresponding distance is called the chord. Angle of attack is the angle formed between the chord line and free stream flow. The term camber, which is perpendicular to the chord line, refers to the maximum difference between the chord line and the camber line. The shape of the camber line is important since cambered airfoils generate lift even at zero angle of attack while symmetric airfoils can not. The thickness of the airfoil, which is the distance between the upper and lower surfaces, is another perpendicular distance from the chord line [1].

Lift force varies with the angle of attack between the chord line and the free stream velocity. Relation between the angle of attack and lift is presented in Figure 2.2 [2]. The figure shows that when the angle of attack varies within a small range, lift behaves linearly. When the angle of attack is within the linear region, flow shows smooth behavior with attached streamlines to the surface of the airfoil. An increase in the angle of attack causes a tendency for separation of the streamlines from the top surface of the airfoil. Thus, a recirculating flow occurs in the wake of the airfoil. This flow characteristic is also named reverse flow. Hence, some portion of the fluid flows in the opposite direction of the free stream. The viscous effect is responsible

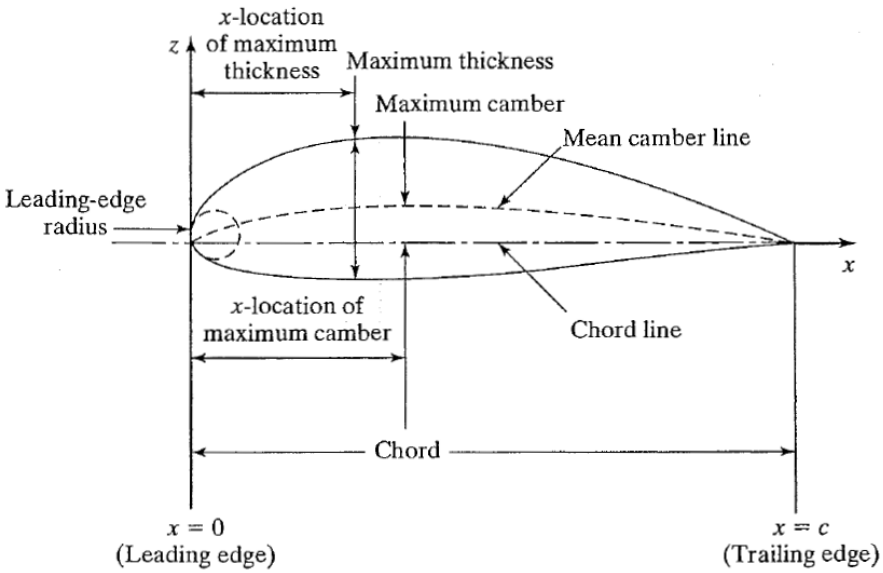


Figure 2.1 Airfoil geometry and terminology [1]

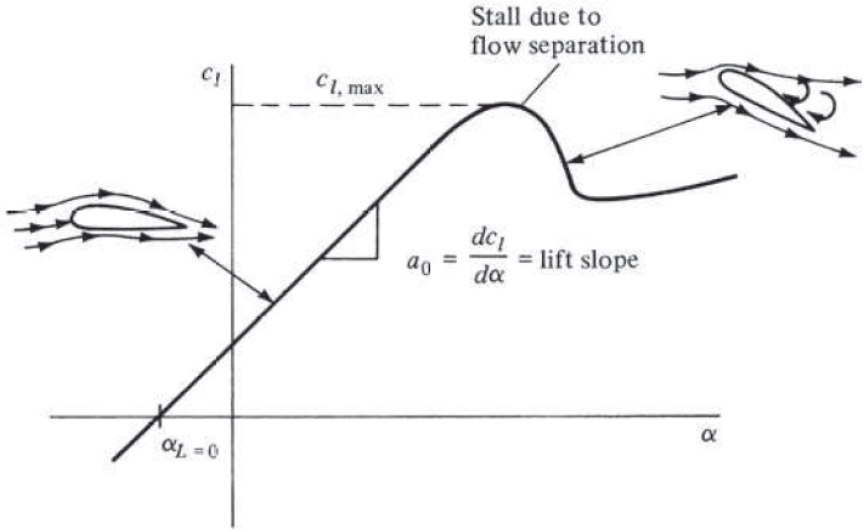


Figure 2.2 Change in the lift coefficient with respect to angle of attack [2]

for fluid flow behavior that starts with separation, which named as stall. When the flow enters the stall regime, a sudden drop occurs in the lift force on the airfoil. Maximum lift occurs just before stall happens. Since maximum lift is an important characteristic, especially for airplanes it determines the stall speed. In many airfoil aerodynamics studies, it is desired to maximize the lift coefficient of the airfoil. If the airfoil is symmetric, it is known that the angle of attack corresponding to zero lift is zero [2].

2.2 Physics Informed Neural Networks

Data-driven machine learning models can be perfectly fitted on training data, however model may not be able to extract physically meaningful predictions. Inevitably, it is necessary to incorporate physical rules and domain knowledge into the machine learning model. Embedding physics into the machine learning model can be considered as informative priors that are hard theoretical constraints for the training. Observational, empirical, or physical understanding of knowledge as a prior information are advantageous for improving training performance of the network model [24]. Generally, enough useful data is very limited for data-driven techniques as it is not practical to obtain experimental data from complex systems. In most of the cases, experimental data for partially known physics are available but with noise and random interruptions. Thus, it is possible to extract missing or even hidden quantities by recovering the solution of partial differential governing equations [24].

Physics-informed Neural Networks (PINNs) [14] are an example of SciML that complement integration of data and governing equations of the physics that are driven by generally partial differential equations. These machine learning methods can stay robust by having governing equations as hard constraints on even imperfect data, i.e., missing data, noisy data or having outliers [24]. Such data-efficient physics-informed neural network models offer an alternative approach to conventional numerical solvers.

Physics-informed neural network structure is illustrated in Figure 2.3. As seen from the figure, physics-informed neural networks can be divided into two parts such as physics-uninformed neural network and physics-informed parts. In the following, PINN details are provided part by part.

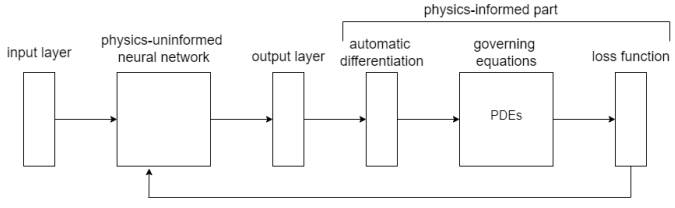


Figure 2.3 Physics-informed neural networks (PINNs) basic structure

Physics-uninformed Neural Network

The first part of the PINN algorithm is an artificial conventional neural network. Neural networks are formed as a result of the connections of neurons with each other. Neuron is mathematically formulated in Equation 2.15.

$$y = \sigma(W^T x + b) \tag{2.15}$$

where x denotes inputs, W is vector of weights, σ is activation function, y is the output of neuron [28].The depth of the network is found by the number of layers minus one and the width is calculated from the number of neurons in a hidden layer. The depth and width of the network affect the complexity and capability of the neural network. Commonly, networks with hidden layers of more than two are considered deep neural networks.

The feed-forward neural network, also called as a multi-layer perceptron (MLP), is the simplest architecture. In such configurations, neurons are in layers, and each neuron is connected to all neurons in their successive layer. The output of each layer can be denoted as follows [29]:

$$f_i(x) = \begin{Bmatrix} x, & i = 1 \\ \sigma(W_i f_{i-1}(x) + b_i), & i = 1, 2, ..k-1 \\ W_i f_{i-1}(x) + b_i, & i = k \end{Bmatrix} \tag{2.16}$$

where i is the layer number, f_i is is output of the i^{th} layer, $i = 0$ is corresponding to the input layer, σ is an activation function. Activation function takes place in neural networks mainly for non-linearity. Even though there are some examples of linear activation functions in the literature [30], a non-linearity is better for accurate approximation [31, 32, 30]. Nonlinear activation function authorize neural networks to recognize complex patterns. W_i is the weight matrix that makes a connection between the $(i-1)^{th}$ and i^{th} layer. Lastly, k represents the output layer. Thus, the network output is obtained by $f(x;\theta) = f_{k(x)}$. In order to find optimum parameters θ, a learning rule is needed. Generally, this is done by cost (loss) function minimization by an optimization algorithm.

The loss function is a measure to lead the algorithm to better performance. It is necessary to find the distance between the predicted and expected output. This measurement can be considered as a feedback signal to the algorithm to adjust it as a way of learning [33]. Loss functions can be either discrete valued for classification problems or continuous valued for regression problems. This section focuses

on solving regression-type problems. A general concern in regression problems is finding a proper function that maps the relation between input features and corresponding labels. Consider a feed-forward neural network $f_\theta : R^a \rightarrow R^b$ with network parameters θ describing the function. A common approach for the solution is the reconstruction of regression the problem as an optimization problem. In this sense, the loss function, also known as the objective function or cost function, is used for penalizing the deviations of the regression function from the data. Thus, the best network parameters are found by minimizing loss function for optimum network parameters θ^*. Network evaluation with a parameter set θ and corresponding loss calculation is named as forward pass. Some mostly applied loss functions are mean absolute error (MAE), mean absolute percentage error (MAPE), Huber loss, mean square error (MSE), and root mean square error (RMSE). In this thesis, MSE is implemented as [34]:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i, \theta) - y_i)^2 \tag{2.17}$$

where $f(x_i, \theta)$ is network prediction, y_i is reference value, N is number of samples.

The loss minimization process is called training or learning. This process takes place on the training data which is a sub-sample of the whole data space that follows, which are being discussed in the context of deep learning. All are local optimization-based algorithms that find local minima of the loss function in every iteration.

The gradient of the entire training data set is calculated by the gradient descent optimization algorithm. To accelerate the process, stochastic gradient descent (SGD) optimization algorithm that calculates gradients over randomly sampled mini-batches of data can be applied. This randomness produces stochastic behavior. Mini-batch size and learning rate are involved in hyper-parameters of the network. The size of the update step in each iteration is determined by the learning rate [35]. Adam Optimizer is another first order gradient-based optimization algorithm with less memory requirement. It is created by taking advantages of AdaGrad method that is good at sparse gradients and RMSProp that is good at on-line and non-stationary conditions [36]. In this study, Adam optimizer is implemented to the PINN algorithm.

The gradient of loss function with respect to network parameters, $\nabla_\theta \mathcal{L}(\theta)$, is key in the update step of the minimization process in the optimization algorithm. Back-propagation algorithm, which takes loss information and sends it back through the network, is implemented for gradient computation [37]. Applications of back-propagation are not limited by the gradient of the loss function but also for any information through a network including the Jacobian of a function. The chain rule is applied through the network to obtain derivatives. Back-propagation executes

chain rule in a highly efficient way with respect to any node in the computational graph [38].

Machine learning software libraries such as Tensorflow [39] and Pytorch [40] are offering a technique named automatic differentiation. Instead of implementing a back-propagation algorithm manually, automatic differentiation makes the same construction with less effort. It creates a framework to watch desired variables and a computational graph is produced accordingly. Graph leafs are associated with variables and nodes are linked to the operations. Since derivations at each operation are well-defined, the chain rule is carried out easily.

Up to this point, multi-layer perceptron type neural network structure is explained. There are other notable types of deep learning models in the literature such as recurrent neural network (RNN), deep belief network (DBN), long short-term memory (LSTM), deep Boltzmann machine (DBM), and restricted Boltzmann machine (RBM) [41]. In this thesis, multi-layer perceptron (MLP) is implemented as a physics-uninformed neural network part of the PINN algorithm.

Physics-informed part

Physical laws are taken into account in the uninformed neural network through the definition of a custom define loss function. Uninformed neural network provides output predictions at the end of the each iteration during training. Gradients of the outputs with respect to the inputs are calculated by automatic differentiation technique. Residuals of the PDE-based governing equations of the problem are calculated by plugging corresponding gradients and outputs into the equations. Norm of the equation residuals are added to the loss function. This penalization leads better output predictions from the network.

Physics-based deep learning method brings some benefits over conventional learning algorithms. Firstly, governing equations as physical laws are highlighted by the help of the custom loss function. In other words, physical laws are imposed in loss function and acting as constraints for the network training. Hence, network predictions are forced the consider integral form of the physical laws. In conventional neural networks, it is possible to have physically irrelevant predictions even, if the accuracy is high.

Neural networks are data-driven methods; therefore, a huge amount of data is needed to build well-performed neural networks. Another advantage of the physics-informed neural network is eliminating the need of huge data sets. It is possible to train a network for a regression-type problem without labels which corresponds to missing data at collocation points in this study. In the absence of available data, network

penalization for the outputs without labels is done by minimization of the governing equation residuals included in the loss function. In real life, it is known that collecting experimental data is challenging in fluid dynamics. Reasons behind this are because of several factors from measurement techniques to the complex geometry of the problem. Besides, applying numerical solutions can be unfeasible in terms of computational cost. Therefore, sparse data set is available for a problem generally. PINN is a relatively new method to approximate PDEs to tackle sparse data sets.

For better understanding, general formulation of PINN is explained in the following as proposed in [42]. Assume that $u(x,t)$ is solution of a system of nonlinear partial differential equations that formulated in the general form below:

$$\begin{aligned} &u_t + \mathcal{N}_x[u] = 0,\ x \in \Omega,\ t \in [0,T] \\ &u(x,0) = h(x),\ x \in \Omega \\ &u(x,t) = g(x,t),\ t \in [0,T],\ x \in \partial\Omega \end{aligned} \tag{2.18}$$

where $\mathcal{N}[\cdot]$ is nonlinear differential operator, subscripts show gradient with respect to either time t or spatial coordinate x, $h(x)$ is initial condition, and $g(x,t)$ is boundary condition. PINN aims to infer continuous solution $u(t,x)$ of PDE by a neural network $f_\theta(x,t)$. For this purpose, PDE residual is expressed as below to be used in the loss function.

$$r_\theta(x,t) = \frac{\partial}{\partial t} f_\theta(x,t) + \mathcal{N}_x[f_\theta(x,t)] \tag{2.19}$$

where $f_\theta(x,t)$ represents neural network output which is prediction of $u(t,x)$ and θ is network parameters. Corresponding loss function is defined by

$$\mathcal{L}(\theta) = \mathcal{L}_u(\theta) + \mathcal{L}_r + \mathcal{L}_{u_0}(\theta) + \mathcal{L}_{u_b}(\theta) \tag{2.20}$$

where the first term is data loss if observational data is available, second term is residual of the PDE, third term is initial condition, and fourth term is boundary condition. Then, network training is done by SGD method:

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta \mathcal{L}(\theta_n) \tag{2.21}$$

where n is iteration number, and η is learning rate.

Chapter 3

Laminar Flow Over a Cylinder

3.1 Problem Definition

Complex fluid behavior is observed even flow passed a simple geometry. Flow over a cylinder is seen as one of the simplest examples of fluid-object interaction. In this chapter, in order to perceive the feasibility of physics-informed neural networks on fluid dynamics, laminar flow over a cylinder in two dimensions is studied. For the PINN algorithm, sparse data set which includes only the temperature field and velocity information at the inlet of the domain is taken from a numeric solution. PINN algorithm is not only implemented for recovering missing temperature field data but also for inferring latent quantities such as velocity and pressure field by approximating PDE-based governing equations. Moreover, aerodynamic forces acting on the cylinder are extracted from the PINN algorithm. Regressed output predictions are used to calculate the forces by two different methods: from the integration of the stress field over the cylinder and from the control volume approach around a cylinder. While the first method is mostly gradient-based, the second method is integral-based. The purpose of implementing two different methods is to find the best fit in terms of produced error.

3.2 Numerical Assessment

COMSOL Multiphysics is used as a CFD tool and a 2D time-dependent model is created to simulate velocity, pressure, and stress fields around a cylinder. Besides, energy transport by the fluid is considered. For this purpose, the laminar flow interface of the CFD module and heat transfer in fluids interface of the heat transfer module is applied.

3.2.1 Geometry and Computational Domain

A cylinder with a diameter of 1 is positioned in a rectangular domain with its axis normal to the free stream flow, U_0. If the cylinder is absent, the velocity would be U_0 everywhere in the rectangular domain. The rectangular domain represents a wind tunnel. The length of the cylinder is considered too long when compared to its diameter. Hence, the same behaviors occur in every plane perpendicular to the normal in an infinite cylinder.

The computational domain which consists of a rectangular stationary domain and cylinder is shown in Figure 3.1. The cylinder wall is defined as a no-slip boundary. In other words, the relative fluid velocity with respect to the cylinder wall is zero, i.e., $u = 0$ at the boundary. The slip boundary condition is implemented on the top and bottom walls of the stationary rectangular domain. Hence, there is no viscous effect on the top and bottom walls, i.e., $u \cdot n = 0$ and thus no boundary layer development appears. It is a valid assumption if walls are used just for keeping the flow in the domain. Unit normal inflow velocity is defined at the inlet of the domain, and zero pressure is applied to the outlet. Initial conditions of velocity and pressure are set to zero.

Additionally, the thermal insulation boundary condition is applied to the top and bottom walls. Temperature is set to 1 in cylinder walls, and 0 is assumed at the inlet.

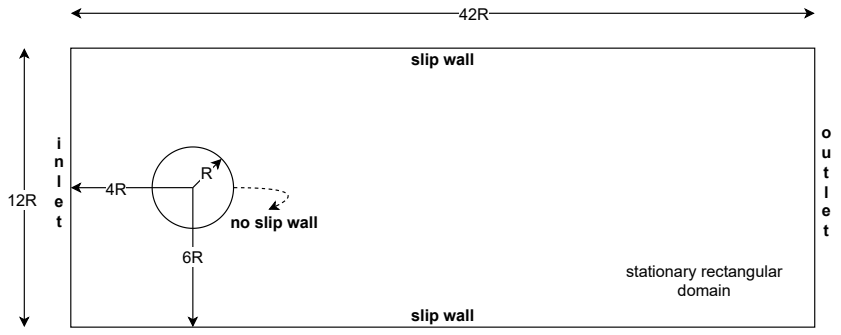


Figure 3.1 Representation of problem geometry and computational domain

3.2.2 Governing Equations

Governing equations of the 2D laminar flow problem in the non-dimensional form are represented below. Superfix * shows that corresponding variable is non-dimensional.

$$\nabla^* \cdot \mathbf{u}^* = 0 \tag{3.1}$$

$$\frac{\partial \mathbf{u}^*}{\partial t} + (\mathbf{u}^* \cdot \nabla^*)\mathbf{u}^* = -\nabla^* p^* + \frac{1}{Re}\nabla^{*2}\mathbf{u}^* \tag{3.2}$$

Non-dimensional equation 3.1 applies conservation of mass and known as continuity equation. Equation 3.2 shows Navier-Stokes equation in non-dimensional vector form that applies conservation of momentum where $\mathbf{u}^*$ is non-dimensional velocity vector, p is non-dimensioanl pressure, Re is Reynolds number.

The flow remains in the laminar flow regime if the Reynolds number is kept on some threshold value. Diameter-based Reynolds number is calculated as 200 from $Re = \rho U_0 D/\mu$. Following scale parameters are used for non-dimensionalization: cylinder diameter D as a length scale, free stream velocity U_0 as a velocity scale, ratio between cylinder diameter and free stream velocity D/U_0 as a time scale, and ρU_0^2 as a pressure scale.

Here, laminar flow and heat transfer in fluids interfaces are coupled to model both slow flow and energy transportation. The main benefit of the multiphysics environment is that interfaces use the same predefined properties like density. Convection-diffusion equation which is provided in equation 3.3 is applied to define temperature in fluid domain.

$$\frac{\partial}{\partial t}^* c^* + u^* \cdot \nabla^* c^* = \frac{1}{Pe}\nabla^{*2} c^* \tag{3.3}$$

where u^* is non-dimensional fluid velocity coming from laminar flow interface, Pe is Peclet number, c is non-dimensional temperature.

Peclet number is related to transport phenomena. In heat transfer, it shows the ratio between the convection of thermal energy to the fluid and the inside of the fluid. It can be expressed as $Pe = RePr$ where Reynolds number (Re) shows the fluid characteristics and Prandtl number (Pr) gives the ratio between momentum diffusivity and thermal diffusivity (α). Thermal diffusivity is expressed as $k/(\rho Cp)$ where k is thermal conductivity, and C_p is heat capacity. In this problem, transport phenomena is characterized by Peclet number of 200 and Pr number is equal to 1.

3.2.3 Mesh

The flow and pressure fields are obtained by solving the governing equations with the finite element method. Both velocity and pressure fields are discretized by the second-order triangular elements. However, the temperature is discretized by linear elements. Applied mesh is shown in Figure 3.2. It consists of 35896 domain elements and 736 boundary elements. It is solved for 136589 number of degrees of freedom within P2+P2 Comsol discretization settings. In order to resolve the thin boundary layer along the no-slip boundary of the cylinder, the boundary layer mesh is implemented in the normal direction along the cylinder wall. It is a layered quadrilateral mesh that has a dense element distribution. Additionally, a smooth transition in element size is applied from the boundary layer mesh to the interior mesh. Corner refinement is implemented for sharp corners to decrease element size. Since it is 2D problem, vertexes are considered sharp corners.

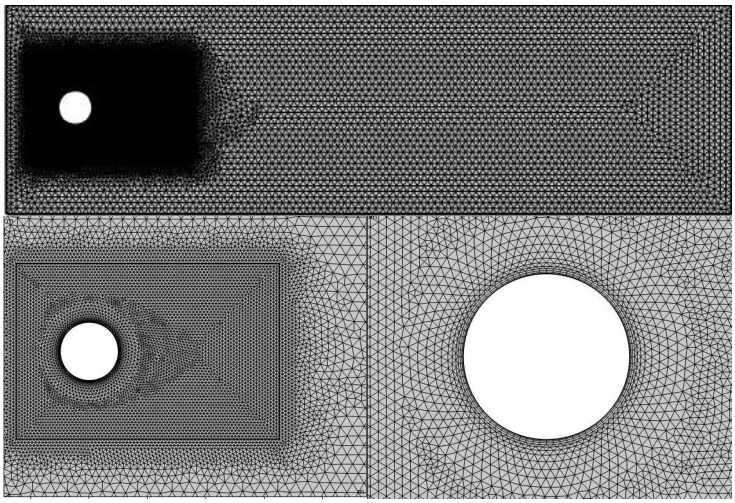


Figure 3.2 Mesh configuration

3.2.4 Solver Configuration

A nonlinear equation system is obtained from Navier Stokes equations. For this reason, a nonlinear solver which is based on iterative methods is needed for the solution. In each iteration, the formation of a linear system that is obtained from a nonlinear system is employed. Since the problem is time-dependent, an additional time marching method is implemented. The Backward Differentiation Formula (BDF)

solver is executed as a time-stepping method. BDF is also called as backward Euler method and is well-known for its stability. It is an implicit solver relying on backward differentiation. The order of accuracy is set to 2.

3.3 PINN Model

In order to test the PINN algorithm on recovering missing data and approximating PDE based-equations for inferring latent quantities, needed sparse data is taken from conducted numeric CFD solution for laminar flow around a cylinder problem as explained in Section 3.2. A rectangular domain with the size of [6R x 9R] as shown in Figure 3.3 with a black label is selected as a region of interest where training data for passive scalar and reference data for velocity and pressure fields are obtained. It is assumed that the training data set includes a passive scalar concentration field at the selected rectangular domain. In [16], it has been proposed that a sufficient amount of passive scalar concentration field gradient at the boundaries can eliminate the requirement of velocity and pressure boundary conditions to a great extent. For the sake of the current study, the left boundary of the region of interest is treated as the inlet boundary and velocity field information at the inlet is fed to the network. In other words, it is assumed that boundary conditions are partially known and include only the inlet boundary. Hence, the only inlet boundary condition is implemented rather than all boundary conditions. As a result, the training data set is consisting of a passive scalar concentration field on the interest domain and a velocity field only at the inlet.

For this problem, the passive scalar concentration field refers to the temperature field. As training data, the temperature field in the interest domain on the time interval of (70,80) with a time step of 0.05 is available in the scattered form of the time and spatial coordinates as illustrated in Figure 3.4. In another way, training temperature data can be explained as $(t, x, y, c)_{n=1}^{N}$ where n is n^{th} training data point from 1 to N, c is the temperature at that point. For instance, temperature field at $t = 30$ is provided in Figure 3.5 for the visualization. Similarly, velocity field data at the inlet is expressed as $(t, x, y, u, v)_{m=1}^{M}$ where m is m^{th} data point from 1 to M. Total data points of the velocity field at the inlet, M, is much smaller than the total data points of passive scalar concentration field, N.

Constructed PINN model architecture is presented in Figure 3.6. When the PINN structure is examined, it can be observed that there are two main parts. In the first part of the model, an uninformed neural network that has a feed-forward deep neural network structure is mapping a function from (t,x,y) to (c,u,v,p). Automatic differentiation is applied to outputs of the uninformed neural network to feed physics-

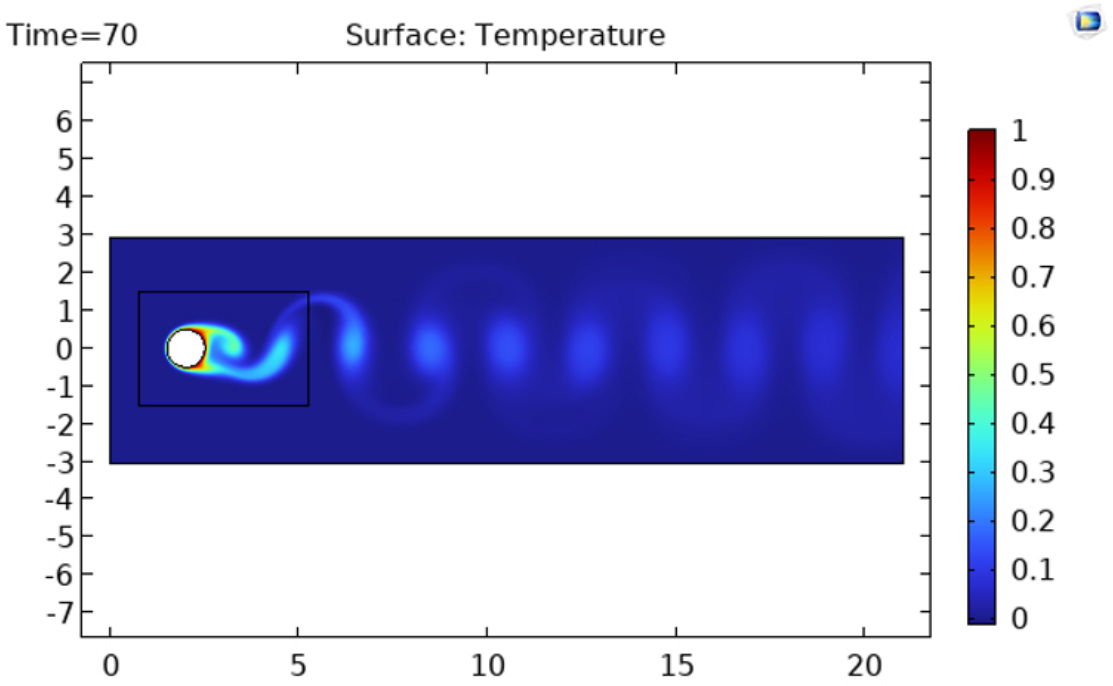


Figure 3.3 Numeric solution of passive scalar concentration field at t = 70. Smaller rectangular domain which has a black framework is selected as a region of interest for the PINN.

informed part for underlying physical laws by governing equations. Residuals of the governing equations in non-dimensional form, as formulated through Equation 3.4-3.7, are implemented to PINN for regression of passive scalar concentration, velocity field, and pressure field.

$$e_1 = \frac{\partial c}{\partial t} + u\frac{\partial c}{\partial x} + v\frac{\partial c}{\partial y} - \frac{1}{Pe}(\frac{\partial^2 c}{\partial x} + \frac{\partial^2 c}{\partial y}) \tag{3.4}$$

$$e_2 = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \frac{1}{Re}(\frac{\partial^2 u}{\partial x} + \frac{\partial^2 u}{\partial y}) \tag{3.5}$$

$$e_3 = \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \frac{1}{Re}(\frac{\partial^2 v}{\partial x} + \frac{\partial^2 v}{\partial y}) \tag{3.6}$$

$$e_4 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \tag{3.7}$$

During the training, besides minimizing the mean square of the training data, norms of the equation residuals are minimized. Thus, physical laws underlying the problem are used by the network as a constraint to penalize predictions of hidden quantities whose training data are not available.

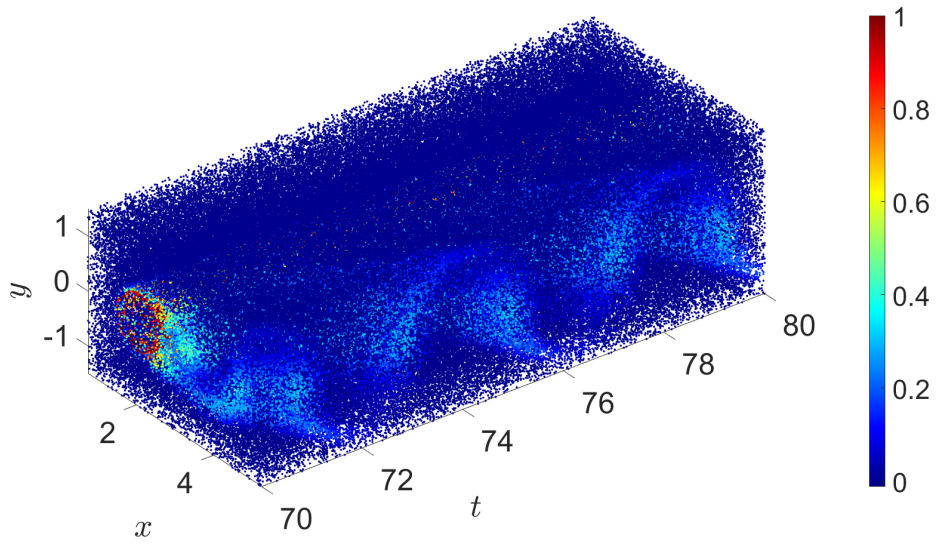


Figure 3.4 Temperature field at the interest domain on the time interval of (70,80) with time step of 0.05 is illustrated in the scattered form.

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{eqns} + \mathcal{L}_{BC} \tag{3.8}$$

In detail, the first loss term $\mathcal{L}_{data}$ denotes the mean square error between training temperature data and network prediction. N is the number of training data points.

$$\mathcal{L}_{data} = \frac{1}{N}\sum_{n=1}^{N} |c_{data}(t^n, x^n, y^n) - c_{pred}(t^n, x^n, y^n)|^2 \tag{3.9}$$

$\mathcal{L}_{BC}$ refers mean square of error between the velocity field at the inlet and model predictions. M is the number of points at the inlet.

$$\begin{aligned}\mathcal{L}_{BC} = \frac{1}{M}\sum_{m=1}^{M} &|u_{data}(t^m, x^m, y^m) - u_{pred}(t^m, x^m, y^m)|^2 \\ &+ |v_{data}(t^m, x^m, y^m) - v_{pred}(t^m, x^m, y^m)|^2\end{aligned} \tag{3.10}$$

The last term of the loss function, $\mathcal{L}_{eqns}$, calculates the norms of the equation residuals. In other words, $\mathcal{L}_{eqns}$ is the mean square error between equation residuals and zero since all the terms of equations are gathered on the left side of the equations, it is aimed to minimize equation residuals through zero). A finite set of residual points where equations are penalized can be at a different number and located differently from the training data set. The finite set of residuals is also known as collocation points. In this study N and J are accepted as equal.

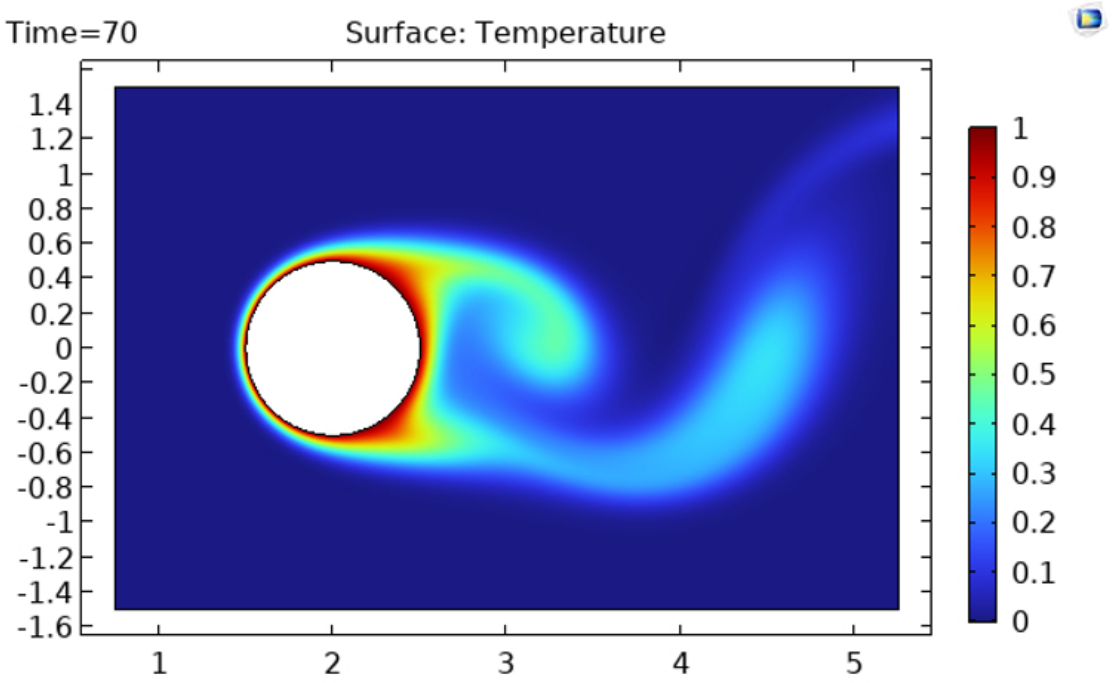


Figure 3.5 The domain where training data for the temperature and reference data for velocity and pressure are obtained. Here, temperature field at t = 70 is visualized.

$$\mathcal{L}_{eqns} = \sum_{i=1}^{4} \frac{1}{J} \sum_{j=1}^{J} |e_i(t^j, x^j, y^j)|^2 \tag{3.11}$$

In the following section, a parametric study is presented for finding the best-fit network parameters for the training. Besides, weight normalization is applied to accelerate the optimizer. Accordingly, the weight vector is parameterized, and thus decoupling is achieved between the vector's length and direction. Weight normalization is a method that is inspired by batch normalization. However, unlike batch normalization, a dependency between mini-batches is not observed [43]. The reason why batch normalization, which is widely used in deep learning, cannot be applied in PINN is that network has a physics-based interface. Otherwise, equations are batch-dependent [16].

3.4 PINN Parametric Study

In this section, before test the physics-informed neural network on the problem, hyper-parameter tuning is studied to reach the best-performed network parameters for the training. Tuned parameters are activation function, number of layers, number of nodes per layer, learning rate, and mini-batch size. A parametric study is applied

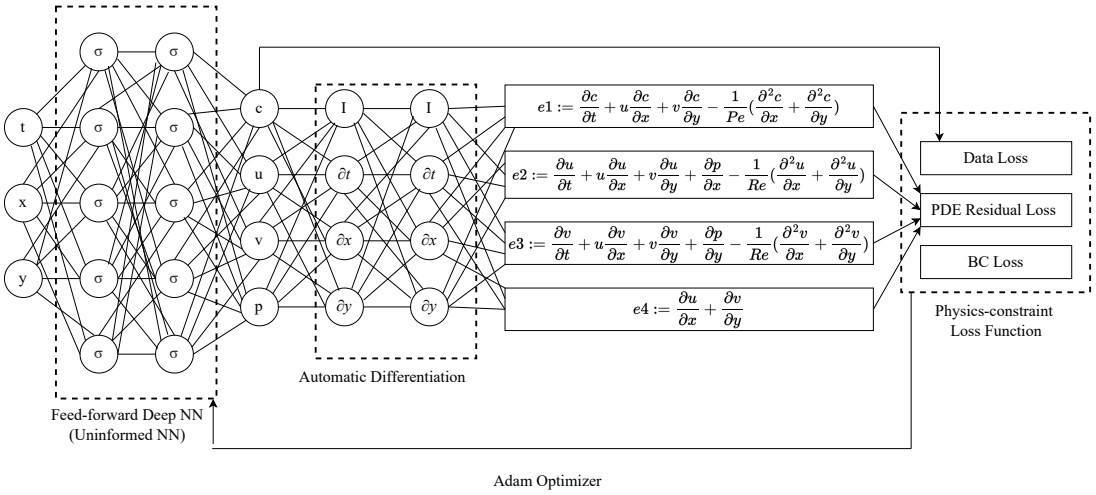


Figure 3.6 Physics informed neural network architecture

manually. While other parameters are kept constant, each network with a different value of the relevant parameter within a range is trained by $5 \cdot 10^4$ iterations.

3.4.1 Activation Function

The selection of activation function plays an important role in achieving accurate PINN training because the derivative of the loss function with respect to network parameters is used during backpropagation in the training process. Additionally, backpropagation takes place not only for the gradient of loss but also for partial differential equation approximation. Hence, backpropagation is dependent on the gradient of activation function as well. Therefore, the first and second derivatives of chosen activation function should not have vanished during the backpropagation. For example, one of the most popular activation functions ReLu (Equation 3.12 [31]) is not a suitable choice for a PINN model since its second derivative vanishes.

In order to find the best-fit activation function, some frequently used ones in the literature are implemented in the PINN training for $5 \cdot 10^4$ iterations. Mean square error training loss results are presented in Figure 3.7. The frequently used sigmoid function in feed-forward deep neural networks is performed worst in the PINN model. It has not a spectacular effect since outcomes of the sigmoid change slightly as approaches the input boundaries. That is why small gradients appear in those regions and learning is stopped by the gradient vanishing problem. Since derivatives are multiplied during the backpropagation and a very small gradient, i.e., near zero, causes a zero result for the whole process. Thus, network parameters, weights, and biases cannot be updated. On the other hand, *elu* (Equation 3.13) and *selu* (Equation 3.4.1) as variants of the *relu* activation function are performed poorly

as well. Even though *selu* did not suffer from the vanishing gradient problem, was not successful on the PINN training. In the literature, it is shown that one of the activation function requirements for a well-trained PINN is smoothness. Since *relu* and their variants show non-smooth behavior, convergence is not achieved [44].The *sin* and *tanh* are differentiable activation functions and as seen they perform better. The best-performed function is the *swish* (Equation 3.14 [45]) activation function which is another differentiable one and implemented for full training in this study.

$$ReLu(x) = max(0, x) \tag{3.12}$$

$$elu(x) = max(0, x) + min(0, \alpha(e^x - 1)) \tag{3.13}$$

where α is a hyperparameter that controls values for negative inputs [46].

$$selu(x) = \lambda \begin{cases} x, & if x > 0 \\ \alpha e^x - \alpha, & if x \leq 0 \end{cases}$$

where $\lambda > 1$ is defined for a slope for positive inputs [47].

$$swish(x) = \frac{x}{1 + e^{-x}} \tag{3.14}$$

3.4.2 Number of layers and number of nodes

A well-trained PINN can be achieved with proper selection of nuumber of layers and node numbers per layer. The number of hidden layers indicates network depth and number of nodes in each layer shows width. As network becomes deeper and wider, its mapping capability for complex functions increases but it may end up with over-fitting. On the other hand shallow and narrow networks may suffer from under-fitting. Since physics-informed neural networks struggle with sparse data sets by using governing equation residuals calculated on collocation points in the training, it prevents over-fitting problem. In order to find best performed network architecture, different networks from depth of 6 to 12 and number of nodes per layer from 100 to 250 are implemented. For each cases, training is limited by $5 \cdot 10^4$ iterations and other hyper-parameters are kept same. Figure 3.8 shows that as number of layer increasing, learning capacity is increasing in case of any number of nodes per layer as well. Networks with same number of layers but having higher number of nodes show better learning performance. The effect of those parameters

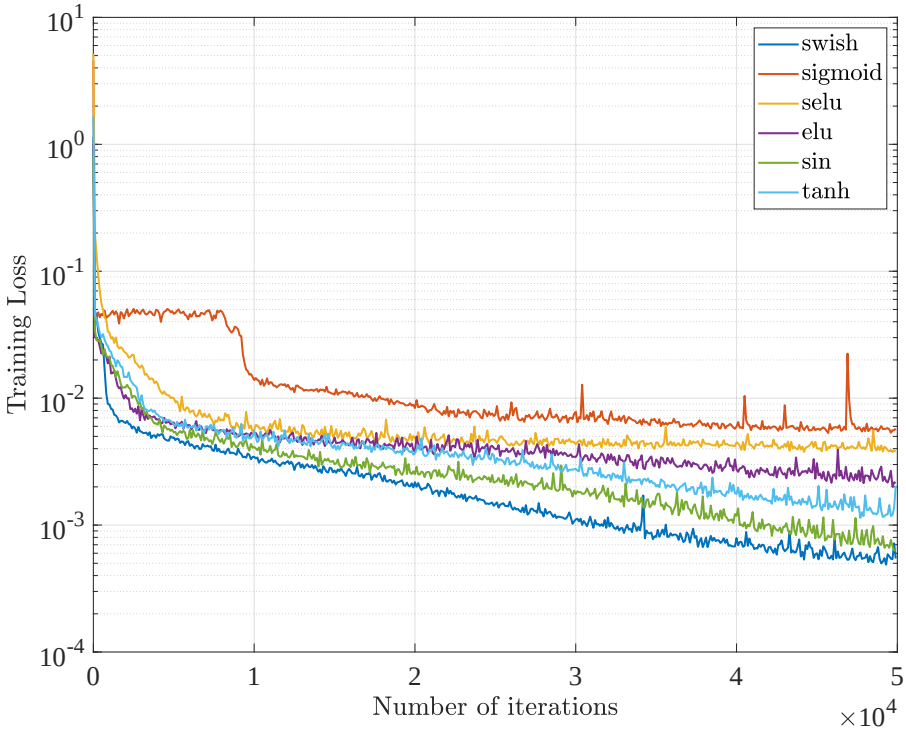


Figure 3.7 MSE Loss for different activation functions

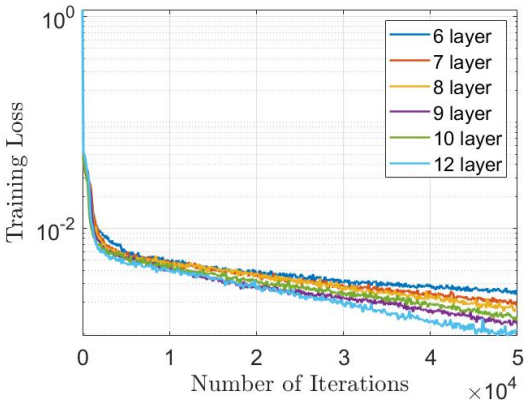
on the model prediction shows consistence behavior with the training as seen from Table 3.1. As a result of relative L2 error (Equation 3.15) comparison, 12 layer and 250 nodes per layer are chosen as best fit parameters for laminar flow over a cylinder problem.

Table 3.1 L2 error norm between model predictions and reference outputs for PINNs in different depth and width

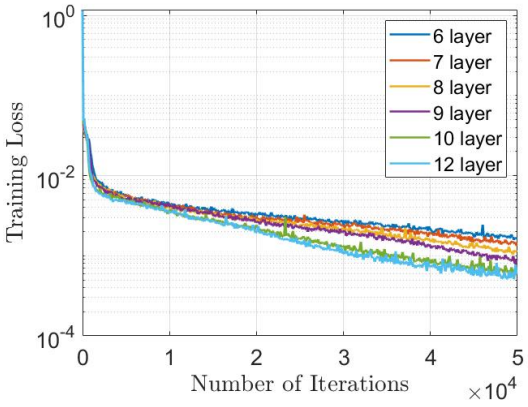
Nodes per layer	6 layer	7 layer	8 layer	9 layer	10 layer	12 layer
100	0.5734	0.4464	0.3812	0.286	0.3593	0.2192
150	0.3593	0.3451	0.2884	0.2419	0.165	0.1419
200	0.3203	0.2711	0.2335	0.1808	0.1676	0.2192
250	0.3834	0.2557	0.1771	0.1574	0.1341	0.1063

3.4.3 Learning Rate

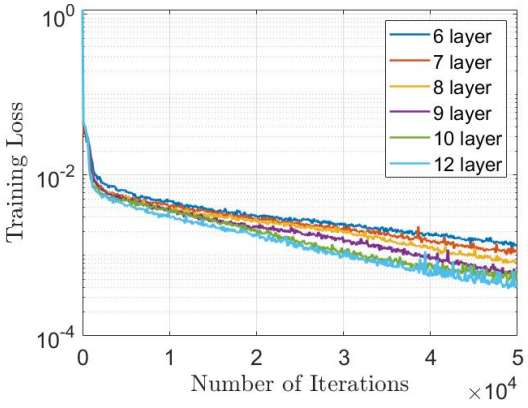
For the PINN training, the Adam optimizer [48] which is a variant of the stochastic gradient descent algorithm is applied. Rather than a learning rate, all optimizer parameters are kept on default parameters. Learning rate provides an amount of update in the opposite direction to the mini-batch gradients of network parameters. The result may not be converged in the case of a high learning rate. Keeping the learning rate relatively smaller may allow to find more optimum network weights.



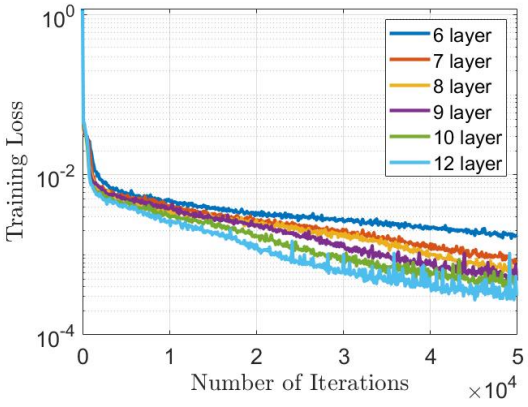
(a) 100 node per layer



(b) 150 node per layer



(c) 200 node per layer



(d) 250 node per layer

Figure 3.8 MSE training loss comparison for number of nodes per hidden layer optimization

However, a smaller learning rate creates computational cost since learning steps become smaller but the necessary number of iterations to reach a minimum of the cost function becomes higher. For 4 different learning rate values ranging from 10^{-2} to 10^{-5}, PINN is performed with $5 \cdot 10^4$ iteration limited training. Figure 3.9 presents loss function behaviours for each case. Training happens slowly at small learning rates as expected. The minimization of the loss function is greater for a learning rate of 10^{-2}. However, it is known that learning happens fast in the first stages of the training and then the rate of change in learning becomes smaller. For this reason, completing training with such a relatively high learning rate can cause a convergence problem when loss falls some threshold. Additionally, 10^{-2} shows faster learning process but it has an oscillatory behavior which points out variation in the network weights are higher than they should be. The learning rate of 10^{-3} shows a sharp learning curve. As a result of the parametric study, the learning rate is accepted as 10^{-2} in the first $5 \cdot 10^4$ iteration and it is decided to update as 10^{-3} in the remaining part of the training for laminar flow over a cylinder problem.

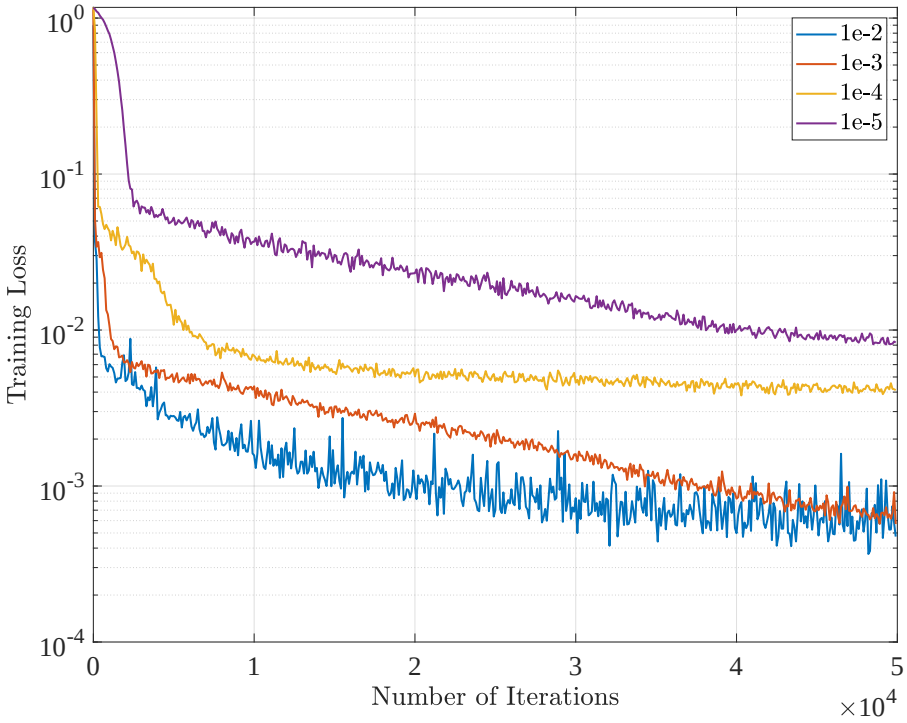


Figure 3.9 MSE Loss for different learning rates

3.4.4 Batch Size

The gradient of the loss function is necessary for network parameter updates as discussed in detail in section 2.2. It can be computationally costly if this process takes place over an entire data set at the end of each iteration. For this reason, randomly sampled small subsets of data called mini-batch are used in machine learning algorithms frequently. In the case of an adequately small learning rate, training with a mini-batch technique and an entire data set give equal results. However, the mini-batch technique achieves good accuracy even with a bigger learning rate in smaller time intervals [49]. One of the limiting factors of the mini-batch technique is size. A larger mini-batch size requires more memory, Additionally, when training hardware is selected as GPU, it is known that mini-batch size as a power of 2 provides better run time [35]. For this work, a code is written in the Python with TensorFlow package for GPU, therefore, powers of 2 from 512 to 8192 are used for mini-batch size hyperparameter tuning. For this purpose, training is limited by $5 \cdot 10^4$ iterations and the learning rate is fixed as 10^{-3}. As seen from Figure 3.10, the learning process speeds up as the mini-batch size increases. Additionally, the loss function shows relatively less fluctuations in higher mini-batch sizes. Besides the training performance, the model prediction accuracy for all cases is compared as well. Table 3.2 shows L2 error between model outputs and reference values. Accordingly, the

error is highest when the mini-batch size is the smallest, i.e., 512. Up to 2048, the error is decreasing and it reaches the lowest error at 2048. Relatively higher batch sizes such as 4096 and 8192 lead to higher errors again. As a result of the training and prediction capabilities of networks in different mini-batch sizes, 2048 is decided to be applied for full training of the problem.

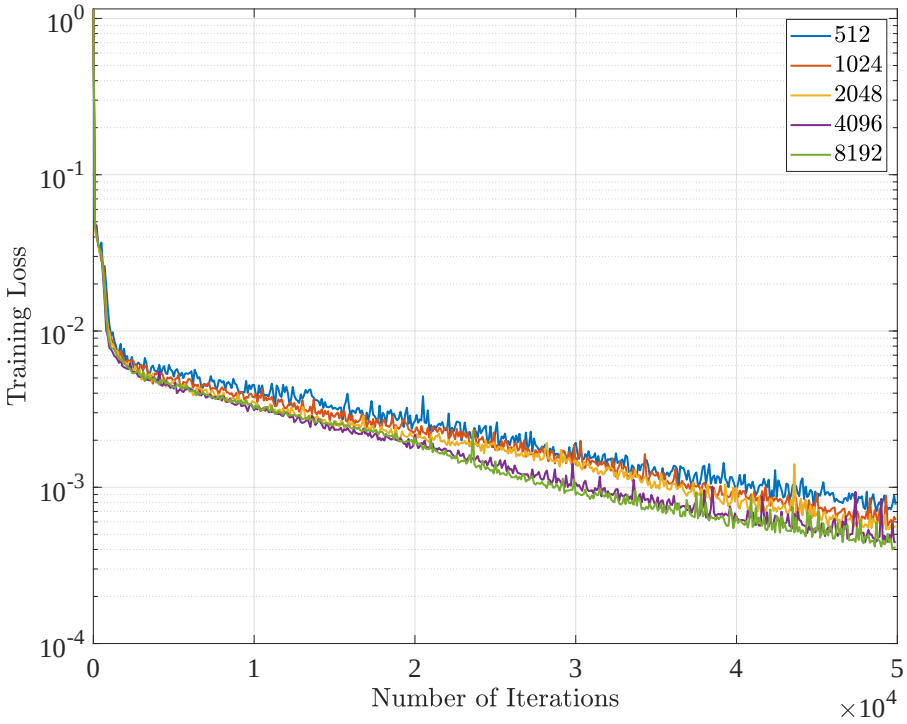


Figure 3.10 MSE loss for different mini-batch sizes

Table 3.2 L2 error norm between model predictions and reference outputs for PINNs in different mini-batch sizes

Batch Size	L2 norm
512	0.2027
1024	0.1469
2048	0.1458
4096	0.1553
8192	0.1638

3.5 Results

Selected hyper-parameters according to parametric study and number of inputs are summarized in Table 3.3. This section represents PINN results after the training with 10^6 iterations. The network is trained and tested on the high performance computer (HPC) with an Intel Xeon Scalable Gold 6148 processor and a single

Nvidia Tesla V100 (16GB, NVLink) GPU. Generally, in deep learning, an epoch that refers to one pass through the entire data set is used as a training metric. PINN differs from conventional deep learning techniques since a physics-informed neural network deals with different data sets of different sizes (i.e., available training data points and collocation points for equation residuals) at the same time. That is why iteration is used as a training metric here rather than epoch.

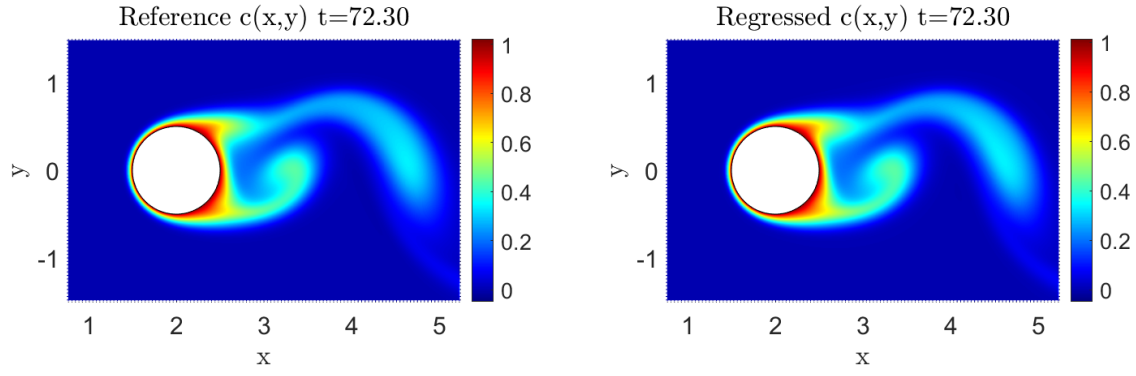
Table 3.3 PINN Summary

Activation Function	Swish
Batch size	2048
Learning rate	10^{-2} if loss $> 6 \cdot 10^{-3}$ otherwise 10^{-3}
Optimizer	Adam
Training time (in iterations)	1 million
Number of Layers	12
Node number per layer	250
Number of data points	11215
Number of collocation points	11215
Number of total time frame	201
Number of time frame (train-test)	(161-40)

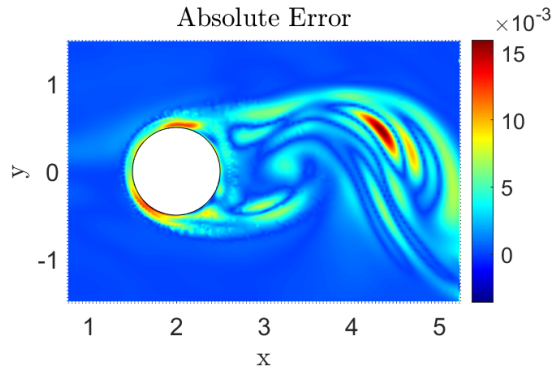
3.5.1 PINN Results

As shown in from Figure 3.11, the temperature field as a passive scalar concentration is regressed successfully, and absolute error between the reference values and model predictions are on the order of 10^{-3}. As seen from the absolute error, higher error occurs where eddy is the strongest. Even any other information is not provided other than the velocity at the inlet as a boundary condition, regression of the velocity and pressure fields shows good accuracy thanks to embedded governing equations into PINN. Since training data for temperature field is available, training time can be increased to reach further improvements in temperature field predictions.

Velocity field components are separately investigated in the same time frame with the represented temperature field at Figure 3.11 over the same data points. Figure 3.12-3.13 illustrate PINN regression on x and y components of the velocity field respectively. It is known that flow is characterized by the Reynolds number of 200 as explained in the 3.2. In this flow regime, as seen from reference velocity field components, Figure 3.12b- 3.13b, there is no symmetry between the upstream and downstream because eddies are observed at the wake of the cylinder. It has been studied that the diameter-based Reynolds number should be at least 47 for a cylinder to observe eddy shedding at the wake region [50]. The flow pattern shows Karman Vortex street behavior in this problem. Accordingly, eddies are shed from both sides of the cylinder boundary continuously and periodically at the wake



(a) (b)

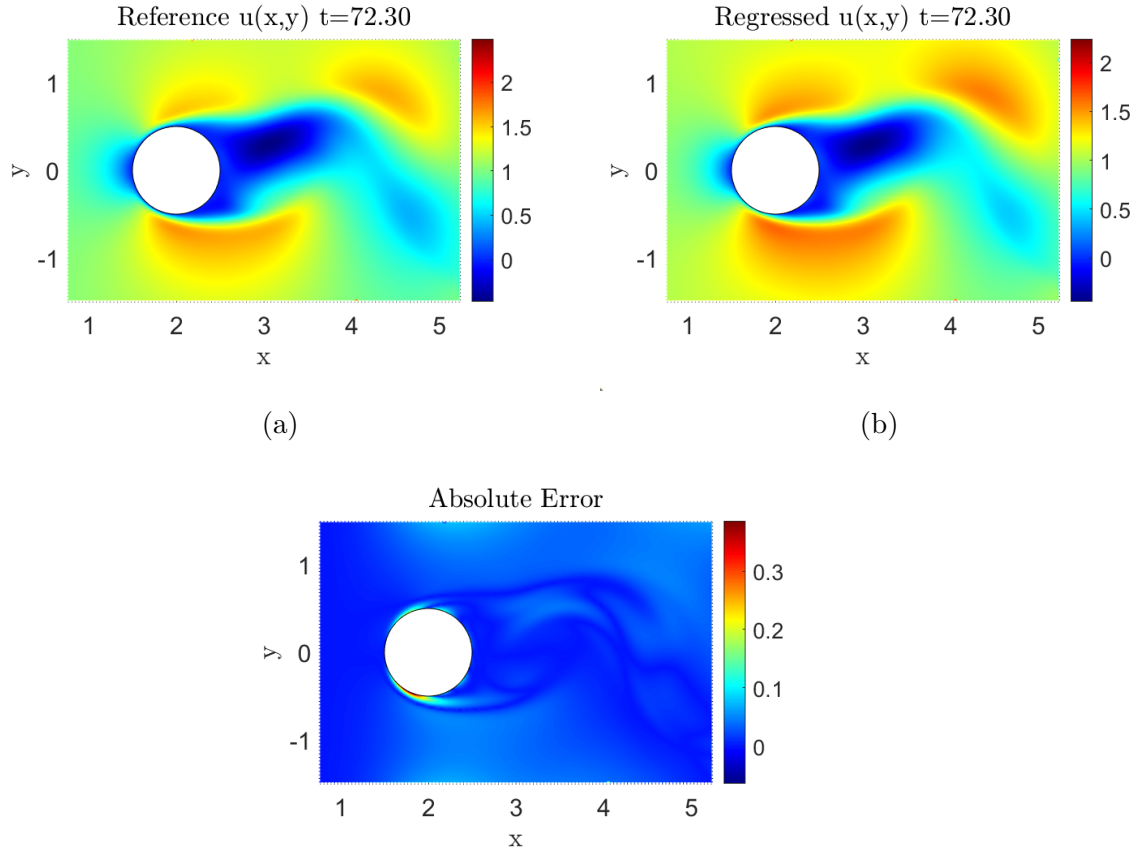


(c)

Figure 3.11 Passive scalar concentration field at t=72.3 a)reference field b)regressed field by PINN c)absolute error

region. Thus, vortices are formed in rows. As moving downstream, the energy of the vortices is reduced by the viscosity. Consequently, as presented in Figures 3.12 and 3.13, PINN algorithm is accurately captured mentioned Karman vortex sheet pattern. When absolute errors of the x and y components of the velocity field are examined, absolute error is shown relatively higher at the cylinder wall boundary. In the numerical study, cylinder boundary condition is defined as no-slip boundary but for the PINN algorithm it is assumed that this information is not available in order to show PINN capability in less information. For further improvements in the velocity field prediction, cylinder-boundary condition can be imposed on loss function as another loss term. However, when the overall domain is considered, errors are small.

Even one single vortex shedding causes asymmetric flow around a cylinder. Thus, the pressure distribution is changed and results in aerodynamic forces which occur periodically on the object. Figure 3.14 shows good agreement between the PINN pressure prediction and reference pressure field. Note that, at least velocity at the inlet is introduced to the PINN algorithm for the sake of a unique solution.



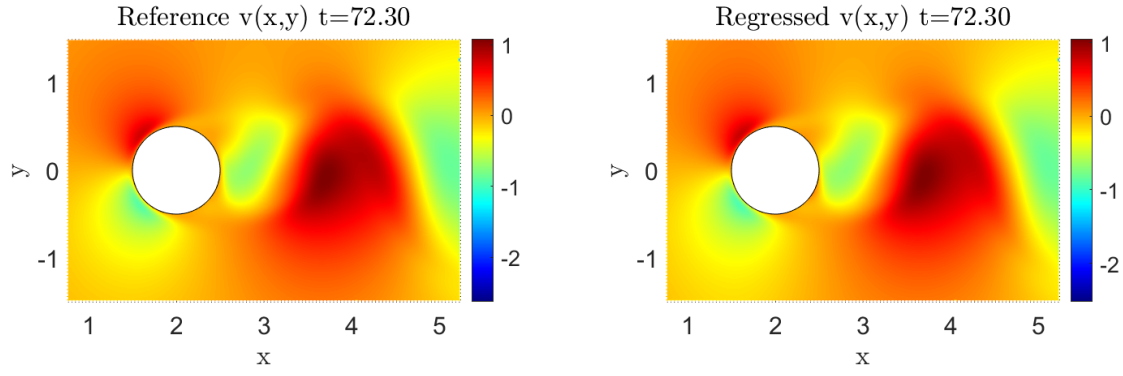
(a) (b)

(c)

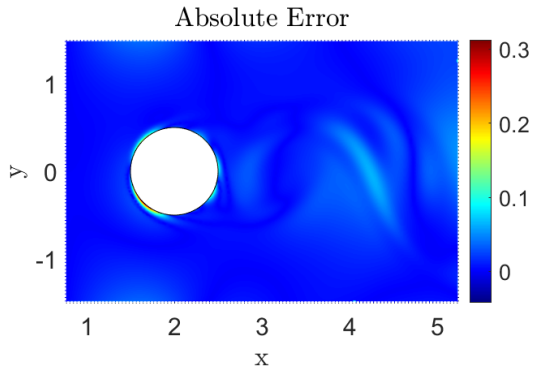
Figure 3.12 x component of the velocity field at t=72.3 a)reference field b)regressed field by PINN c)absolute error

However, it is assumed that any information from pressure field is not available to for the network training. Pressure is just extracted from the approximate solution of Navier Stokes equations for incompressible flow by PINN. If the presented absolute error of the pressure field in Figure 3.14 is compared with the absolute errors of the velocity field, it can be seen that the error for pressure over the domain is relatively higher than velocity field errors. The reason is that only the gradient of the pressure is included in incompressible Navier-Stokes equations rather than a direct pressure term. However, pressure regression result still can be considered in the high accuracy range when one considers no observational data being provided.

For illustrative purposes of PINN algorithm performance, regression of velocity and pressure fields on single time (t = 72.30) is provided in Figures 3.11-3.14. At the whole time window of [70,80], relative L2 errors for all output fields on the interest domain are provided in 3.15. Relative L2 error is used as an error metric as the reference paper of this study suggested [16] since it is agnostic to any shift or scaling on regressed and reference function. Relative L2 error is expressed in Equation 3.15.



(a) (b)



(c)

Figure 3.13 y component of the velocity field at t=72.3 a)reference field b)regressed field by PINN c)absolute error

$$Relative\ L2\ error = \frac{\frac{1}{N}\sum_{i=1}^{N}[f(x_i) - g(x_i)]^2}{\frac{1}{N}\sum_{i=1}^{N}[g(x_i) - \frac{1}{N}\sum_{i=1}^{N} g(x_i)]^2} \tag{3.15}$$

where x_i is points from 1 to N in a scattered form through the domain, f is regressed function and g is reference function g. Figure 3.15 shows prediction error is high at the beginning of the time windows for each field. This high error can be explained by a lack of training data when $t < 70$.

3.5.2 Force Calculation

Velocity and pressure fields are hidden quantities that are extracted from PINN by training on passive scalar concentration field and underlying physical laws as a constraint for regression. Regressed fields are used to calculate the lift and drag force exerted on the cylinder. Two different methods are applied to calculate aerodynamic forces. The first method is relying on stress field integration over the cylinder which is the same applied method with the reference paper of this study [16]. As an alternative method, the control volume approach is implemented to exert forces

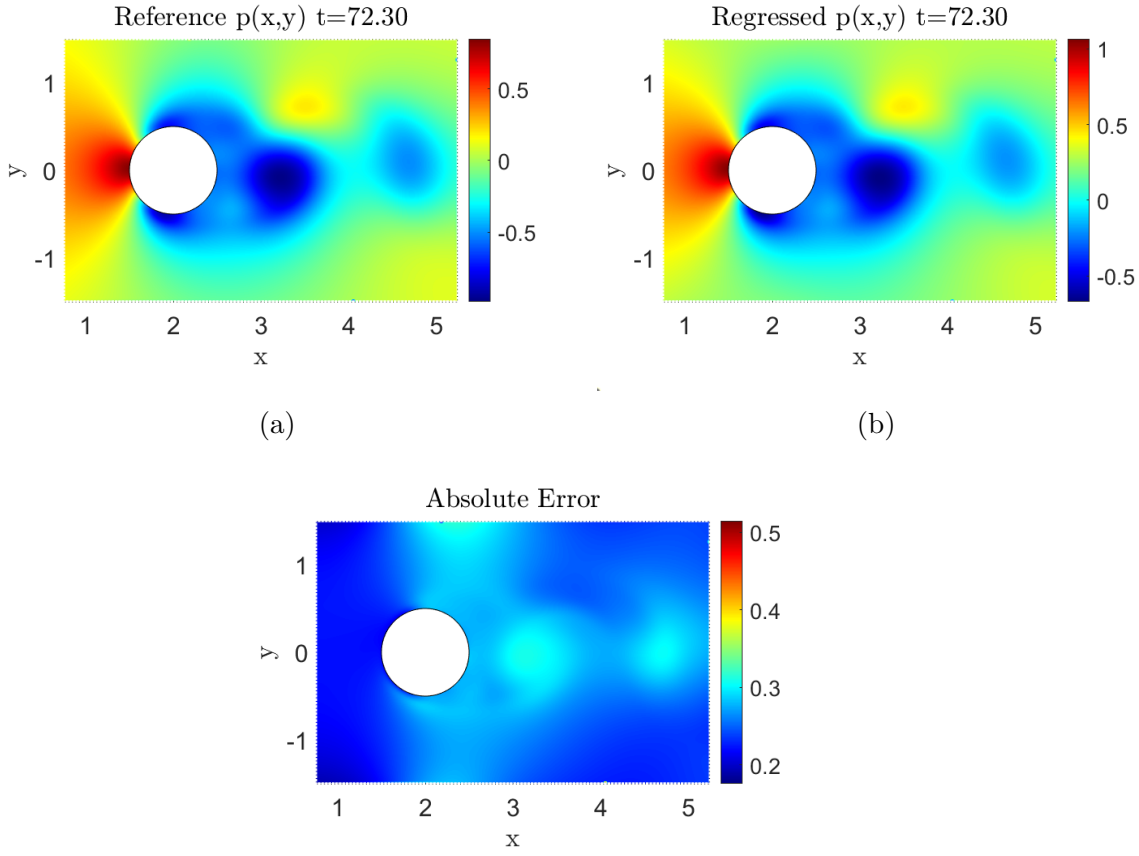


Figure 3.14 pressure field at t=72.3 a)reference field b)regressed field by PINN c)absolute error

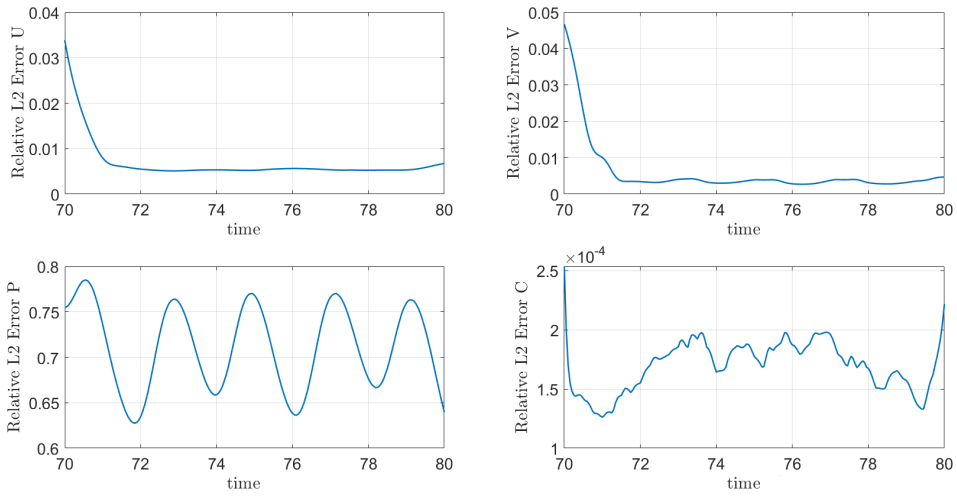


Figure 3.15 Relative L2 error of outputs on the domain over whole time window

from the conservation of momentum. Detailed analysis of obtaining aerodynamic forces from stress and control volume approaches are provided in Section 2.1.2.

Since drag force occurs in parallel to upstream velocity, it is calculated from the integration of the stress field in the x-direction as shown in Equation 3.16. Similarly, since lift force occurs perpendicular to the upstream velocity, it is calculated from the integration of the stress field in the y-direction as shown in Equation 3.17.

$$F_D = \int (-pn_x + 2\,\mu \frac{\partial u}{\partial x}\,n_x + \mu\,(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})\,n_y)\,dA \tag{3.16}$$

$$F_L = \int (-pn_y + 2\,\mu \frac{\partial v}{\partial y}\,n_y + \mu\,(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})\,n_x)\,dA \tag{3.17}$$

As an alternative approach for aerodynamic force calculation, conservation of linear momentum for a fixed control volume is indicated below.

$$\sum F = \frac{d}{dt}(\int_{CV} V\rho\,d\mathcal{V}) + \int_{CS} V\rho(V \cdot n)\,dA \tag{3.18}$$

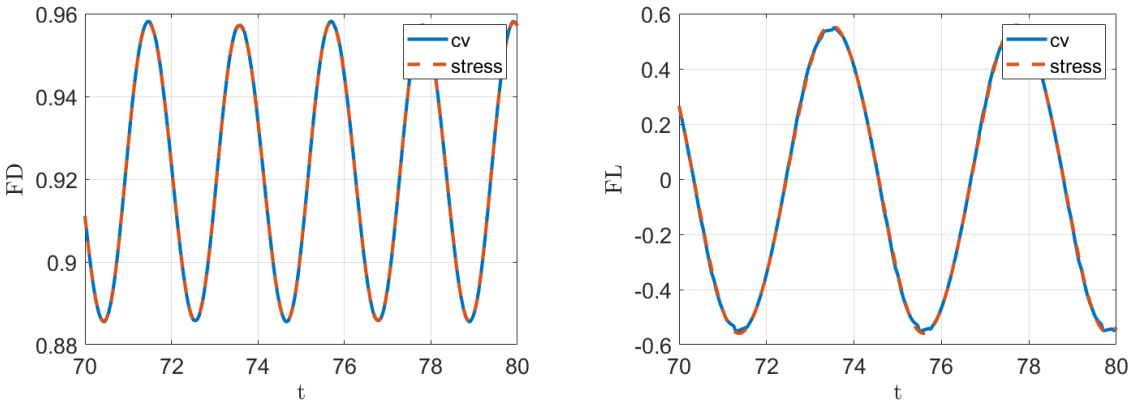
Equation 3.18 is a vector equation since all terms include V which represents the fluid velocity. $\sum F$ is the vector sum of forces acting on the cylinder. Control volume of defined as boundary of the rectangular domain. In order to calculate lift and drag forces acting on the cylinder in the control volume, x and y components of Equation 3.18 should be written while considering the pressure and viscous forces as shown below.

$$\begin{aligned} F_D = {} & \frac{d}{dt}(\int_{CV} u\,\rho\,d\mathcal{V}) + \int_{CS} u\,\rho\,(un_x + vn_y)\,dA \\ & + \int_{CS}(pn_x)\,dA - \int_{CS}[(\mu\,2\frac{\partial u}{\partial x}n_x) + (\mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})n_y)]dA \end{aligned} \tag{3.19}$$

$$\begin{aligned} F_L = {} & \frac{d}{dt}(\int_{CV} v\,\rho\,d\mathcal{V}) + \int_{CS} v\,\rho\,(un_x + vn_y)\,dA \\ & + \int_{CS}(pn_y)\,dA - \int_{CS}[(\mu\,2\frac{\partial v}{\partial y}n_y) + (\mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})n_x)]dA \end{aligned} \tag{3.20}$$

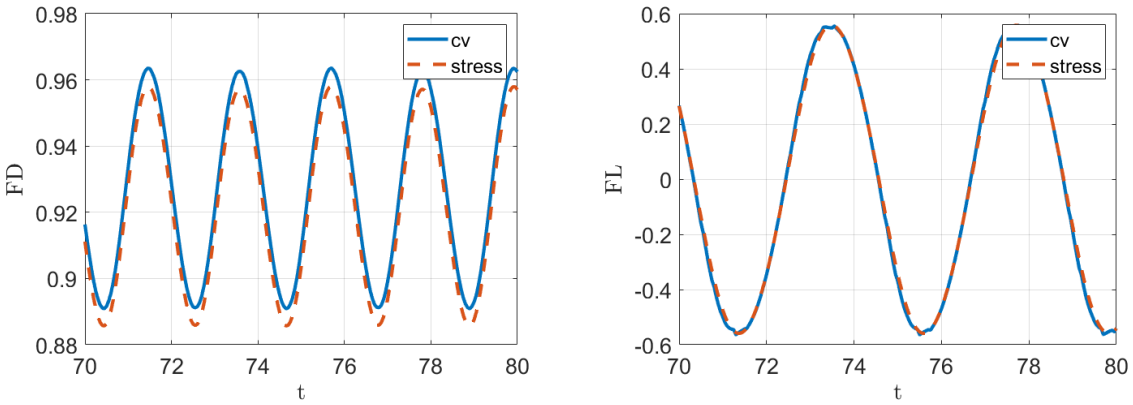
Both methods were applied to the numeric study in COMSOL for comparison. As seen from Figure 3.16, they are in good agreement for lift and drag force calculations. In equations 3.19 and 3.20, viscous forces are taken into consideration for the calculation of aerodynamic forces via the control volume approach. In order to see the effect of viscous terms, they are neglected, and the validation study is repeated as presented in Figure 3.17. Especially in the lift force, the viscous effect

is so small therefore it can be neglected. Although the error seems relatively high at peak points for the drag force, there is an average of 0.58% error. Therefore, viscous terms can be neglected for computational efficiency to avoid calculations of unnecessary gradients. Besides, viscous terms are neglected when aerodynamic forces are extracted from PINN predictions via a control volume approach.



(a) (b)

Figure 3.16 Aerodynamic force calculation by SI and CV methods on reference data. CV approach includes viscous term at the boundary for a)drag force b)lift force



(a) (b)

Figure 3.17 Aerodynamic force calculation by SI and CV methods on reference data. CV approach does not include viscous term at the boundary for a)drag force b)lift force

Using the velocity and pressure field predictions given by the PINN algorithm, lift and drag forces acting on the cylinder are calculated in two ways through the mentioned stress and control volume methods. Figure 3.18 shows drag force extraction from PINN predictions. As seen from the figure, both methods perform similarly. Even the oscillations of the drag force is captured by the model, there is some offset in terms of amplitude between the reference value and predictions by each method.

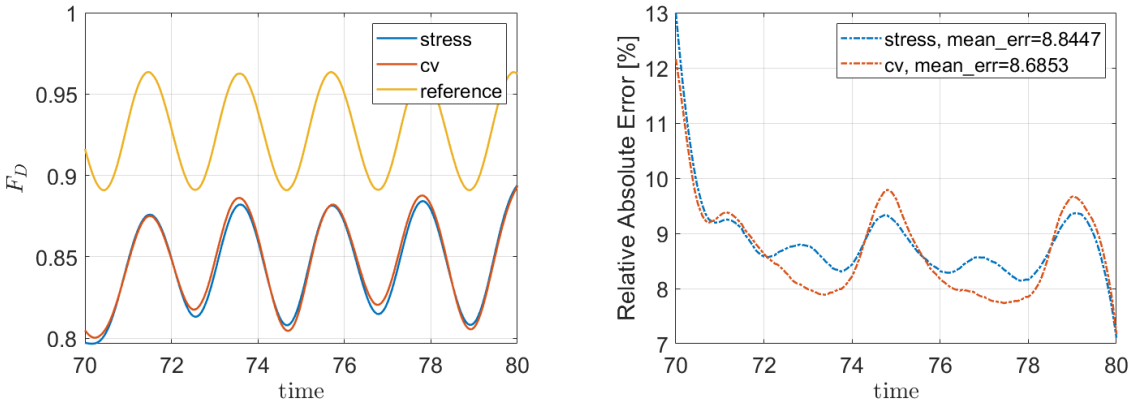
The mean relative absolute error (i.e., Equation 3.21) of drag force prediction is 8.84% for the stress integration method and 8.68% for the control volume method. For both methods, at the very beginning of the time window (t=70) the relative absolute error is maximum, about 13% and 12% for stress and cv methods respectively. The higher error there can be explained by the lack of training data at t<70.

$$\text{Mean Relative Absolute Error} = mean_i|\frac{f(x_i) - g(x_i)}{f(x_i)}| \cdot 100 \tag{3.21}$$

$$\text{Mean Absolute Error} = mean_i|f(x_i) - g(x_i)| \tag{3.22}$$

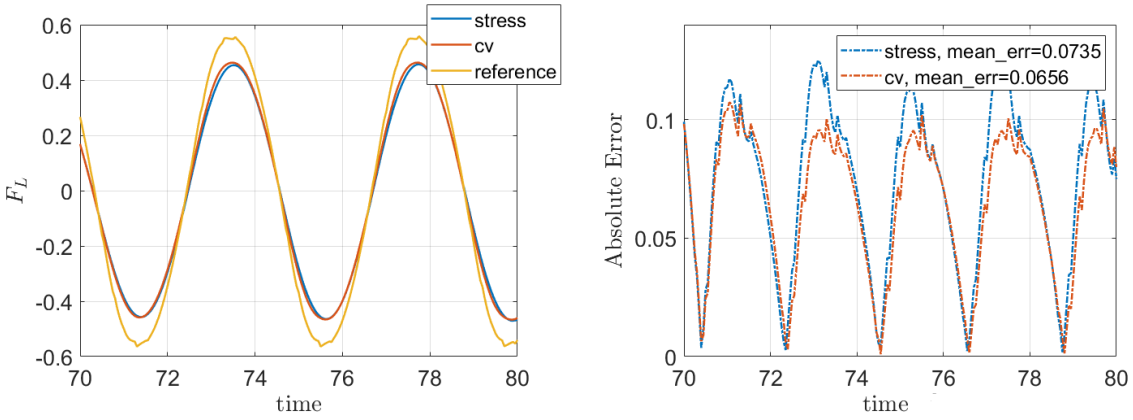
where $f(x_i)$ is the reference data and $g(x_i)$ is the network prediction.

Figure 3.19 show the result for lift force's from PINN predictions. When compared with the drag force's, PINN predictions lead to more accurate results in lift force calculations. Here, since both the control volume and the stress methods are in very good agreement with the reference, the error plot is given directly in absolute error instead of the relative absolute error metric. Since, especially in the regions where F_L is zero or very close to zero, the relative absolute error results in a much larger than 100% although the actual error is very low. Considering the mean absolute error (i.e., Equation 3.22), this value is about 0.07 for the stress method, while it is 0.065 for the control volume method. Besides, the maximum absolute error is 0.12 for the stress method and 0.1 for the control volume method. Here, it can be said that the two methods show very close performances and are good at catching oscillatory behavior with accurate amplitudes.



(a) (b)

Figure 3.18 Drag force calculation a) by control volume approach and stress field approach b) relative absolute error of applied methods



(a) (b)

Figure 3.19 Lift force calculation a) by control volume approach and stress field approach b) absolute error of applied methods

Chapter 4

Turbulent Flow Investigation via PINN

4.1 Problem Definition

Complex fluid behavior and chaotic pressure changes are observed in case of turbulent flow over an object. Turbulence is accepted as one of the most important unsolved problems in science. It is known that airfoils show high lift generation when compared with other solid objects because of their special aerodynamic design. In high angle of attacks, stall problem causes a sudden drop in the lift force. The stall is another complex phenomenon. In this study, flow over a stationary airfoil with a high angle of attack in two dimension is studied in order to investigate stall in turbulent flow. Firstly, 2D-time dependent CFD studies with Spalart-Allmaras and k-ω turbulence models are conducted separately by COMSOL. Details of the numeric study is presented in Section 4.2. From numeric solutions, two separate sparse data sets are created for the PINN algorithm. Sparse data sets include the y-component of the velocity field and eddy viscosity information over the domain. PINN algorithm is not only implemented for recovering missing data of y-component velocity field and eddy viscosity but also for inferring latent quantities such as x-component of the velocity and pressure fields by approximating PDE-based governing equations. It is known that eddy viscosity is a variable that does not exist in real life. It represents transported and dissipated energy by turbulence. Therefore, it is expected to have different eddy viscosity information from different turbulence models for the same problem. The purpose of studying the two different data sets for the same problem is to observe the PINN performance with different data for the regression of the same problem with the same governing equations. Moreover, aerodynamic forces acting on the airfoil are extracted from the PINN algorithm. Regressed output predictions

are used to calculate the forces by two different methods: from the integration of the stress field over the airfoil and from the control volume approach around an airfoil. While the first method is mostly gradient-based, the second method is integral-based. The purpose of implementing two different methods is to find the best fit in terms of produced error. Details of the PINN model for turbulence problem is provided in section 4.3 and results are shown in section 4.5.1.

4.2 Numerical Assessment

CFD module of the COMSOL Multiphysics is used to create 2D time-dependent models to simulate turbulent flow around a stationary airfoil. For this purpose, two different turbulent flow interfaces of the CFD module are applied: Spalart-Allmaras and k-ω turbulent flow models. Hence, two CFD studies are conducted for the same geometry and computational domain.

4.2.1 Geometry and Computational Domain

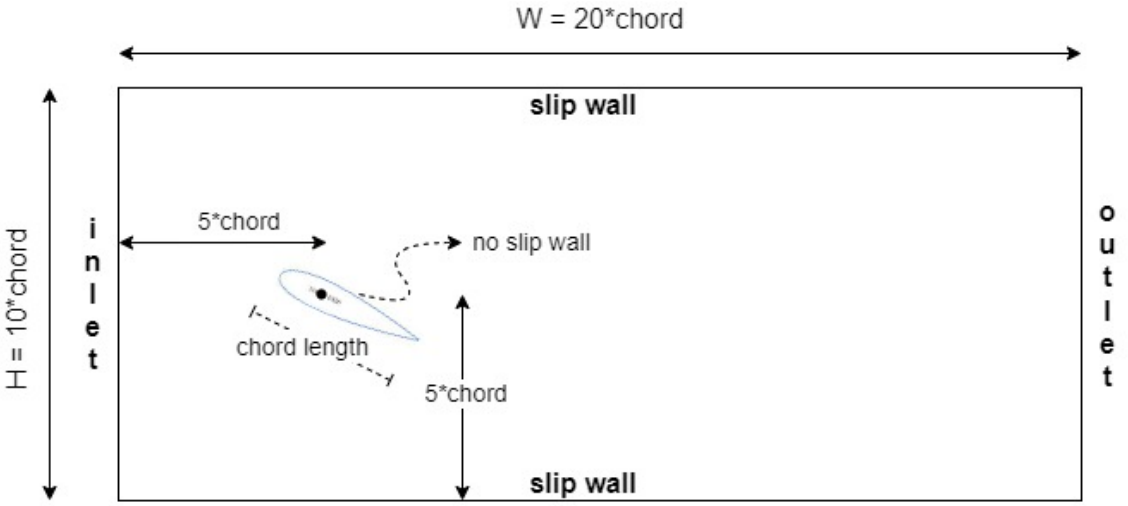


Figure 4.1 Representation of geometry and computational domain

NACA0020 airfoil with a chord length of 1 is positioned in a rectangular domain with a -25^o angle of attack with respect to free stream flow, U_0. A computational domain that consists of a rectangular stationary domain and airfoil is shown in Figure 4.1. The airfoil wall is defined as a no-slip boundary. In other words, the relative fluid velocity with respect to the airfoil wall is zero, i.e., $u = 0$. The slip boundary condition is implemented on the top and bottom walls of the stationary rectangular domain. Hence, there is no viscous effect on the top and bottom walls, i.e., $u \cdot n = 0$ and thus no boundary layer development appears. It is a valid assumption if walls are used just for keeping the flow in the domain. Unit normal inflow velocity is

defined at the inlet of the wind tunnel, and zero pressure is applied to the outlet. Initial conditions of velocity and pressure are set to zero.

4.2.2 Mesh Configuration

The velocity and pressure fields are obtained by solving the fluid flow governing equations with the finite element method. Both velocity and pressure fields are discretized by the first-order linear elements. Triangular mesh is applied over the domain as shown in Figure 4.2. It consists of 73488 domain elements and 990 boundary elements. It is solved for 300923 number of degrees of freedom. In order to resolve the thin boundary layer along the no-slip boundary of the airfoil, the boundary layer mesh is implemented in the normal direction along the airfoil wall. It is a layered quadrilateral mesh that has a dense element distribution. The thickness of the first layer is 1/20 of the local domain element height. Number of layers is 6 and stretching factor is 1.2 which increase the thickness between the successive layers by 20%. Additionally, a smooth transition in element size is applied from the boundary layer mesh to the interior mesh. Corner refinement is implemented for sharp corners to decrease element size. Since it is 2D problem, vertexes are considered sharp corners.

4.2.3 Turbulent Model

Two CFD studies are conducted with two different turbulence models as Spalart-Allmaras and k-ω. The reason behind conducting two different CFD studies is that train the PINN algorithm on two data sets coming from different turbulent models and compare the performance whether it is affected or not. Detailed information about turbulent models is provided below.

Spalart-Allmaras Turbulence Model

Spalart-Allmaras (S-A) is a Reynolds-Averaged-Navier-Stokes (RANS) type turbulence model. In other words, RANS equation is solved for the conservation of momentum, and the continuity equation is implemented for the conservation of mass. It is known that the Spalart-Allmaras model is frequently used for aerodynamic applications such as flow around airfoil and turbine blade simulations. In general, it is accepted as a robust model.

Spalart-Allmaras model is known as a one-equation turbulence model[51]:

$$\frac{\partial \tilde{\nu}}{\partial t} + u \cdot \nabla \tilde{\nu} = c_{b1}\tilde{S}\tilde{\nu} - c_{w1} f_w (\frac{\tilde{\nu}}{l_w})^2 + \frac{1}{\sigma}\nabla \cdot ((\nu + \tilde{\nu})\nabla \tilde{\nu}) + \frac{c_{b2}}{\sigma}\nabla \tilde{\nu} \cdot \nabla \tilde{\nu} \qquad (4.1)$$

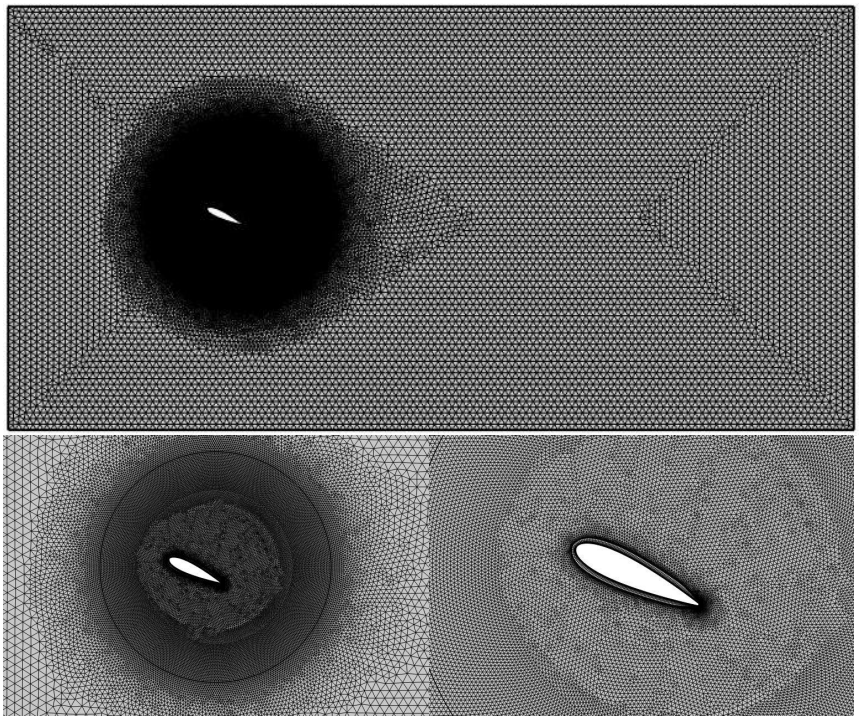


Figure 4.2 Mesh configuration

where $\tilde{\nu}$ is S-A working variable that fulfill the transport equation, ν is kinematic viscosity.

Eddy viscosity, ν_t is calculated by $\nu_t = \tilde{\nu} f_{v1}$, $f_{v1} = \frac{\mathcal{X}^3}{\mathcal{X}^3 + c_{v1}^3}$, $\mathcal{X} = \frac{\tilde{\nu}}{\nu}$

Modified vorticity, $\tilde{S}$, is calculated as follows

$$\tilde{S} = S + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v^2}, \quad f_{v2} = 1 \frac{\mathcal{X}}{1 + \mathcal{X} f_{v1}} \tag{4.2}$$

where S is vorticity magnitude, and d is closest wall distance.

f_w is a function defined by

$$f_w = G[\frac{1 + c_{w3}^6}{G^6 + c_{w3}^6}]^{1/6}, \quad G = r + c_{w2}(r^6 - r), \quad r = min(\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, r_{lim}) \tag{4.3}$$

Constant terms are specified as $c_{b1} = 0.1355$, $\sigma = 2/3$, $c_{b2} = 0.622$, $\kappa = 0.41$, $c_{w1} = c_{b1}/\kappa^2 + (1 + c_{b2}/\sigma)$, $c_{w2} = 0.3$, $c_{w3} = 2$, $c_{v1} = 7.1$, $r_{lim} = 10$.

$k-\omega$ **Turbulence Model**

$k-\omega$ is turbulence model also utilizes RANS equations for conservation of momentum and continuity equation for conservation of mass. COMSOL Multiphysics CFD provides the revised version of $k-\omega$ model by Wilcox [52]. k which is turbulent kinetic energy and ω which is dissipation per unit of turbulent kinetic energy are solved by the model from the following equations.

$$\begin{aligned}\rho\frac{\partial k}{\partial t}+\rho u\cdot\nabla k &= P_k-\rho\beta^* k\omega+\nabla\cdot((\mu+\sigma^*\mu_T)\nabla k)\\ \rho\frac{\partial w}{\partial t}+\rho u\cdot\nabla\omega &= \alpha_0\frac{\omega}{k}P_k-\rho\beta_{00}\omega^2+\nabla\cdot((\mu+\sigma\mu_T)\nabla\omega)\end{aligned} \tag{4.4}$$

Kinematic eddy viscosity is defined as

$$\nu_T=\frac{k}{\tilde{\omega}} \text{ where } \tilde{\omega}=max(\omega,\ C_{lim}\sqrt{\frac{2S_{ij}S_{ij}}{\beta^*}}),\ \ C_{lim}=7/8 \tag{4.5}$$

Closure coefficients and auxiliary relations between them are specified below.

$$\alpha_0=\frac{13}{25},\ \ \beta_{00}=\beta_0 f_\beta,\ \ \beta^*=\frac{9}{100},\ \ \sigma=\frac{1}{2},\ \ \sigma^*=\frac{3}{5},\sigma_{do}=\frac{1}{8} \tag{4.6}$$

$$\sigma_d=\begin{cases}0, & \frac{\partial k}{\partial x_j}\frac{\partial w}{\partial x_j}\le 0\\ \sigma_{do}, & \frac{\partial k}{\partial x_j}\frac{\partial w}{\partial x_j}>0\end{cases}$$

$$\beta_0=0.0708,\ \ f_\beta=\frac{1+85\mathcal{X}_w}{1+100\mathcal{X}_w},\ \ \mathcal{X}_w=|\frac{\Omega^0_{ij}\Omega^0_{jk}S_{ki}}{(\beta\omega)^3}| \tag{4.7}$$

where Ω^0_{ij} is mean rotation rate tensor and S_{ij} is mean strain rate tensor:

$$\Omega_{ij}=\frac{1}{2}(\frac{\partial\bar{u}_i}{\partial x_j}-\frac{\partial\bar{u}_j}{\partial x_i}),\ \ S_{ij}=\frac{1}{2}(\frac{\partial\bar{u}_i}{\partial x_j}+\frac{\partial\bar{u}_i}{\partial x_j}+\frac{\partial\bar{u}_j}{\partial x_i}) \tag{4.8}$$

P_k is the production term and formulated as

$$P_k=\mu_T(\nabla u:(\nabla u+(\nabla u)^T)-\frac{2}{3}(\nabla\cdot u)^2)-\frac{2}{3}\rho k\nabla\cdot u \tag{4.9}$$

where dynamic eddy viscosity is

$$\mu_T=\rho\frac{k}{\omega} \tag{4.10}$$

4.3 PINN Model

In order to see the capability of the PINN algorithm on recovering missing data and approximating PDE based-equations for inferring latent quantities, needed sparse data is taken from conducted numeric CFD solution for turbulent flow around an airfoil problem as explained in 4.2. The same architecture of PINN models is created for different turbulence models.

For the PINN models, sparse data sets are obtained from numeric solutions. A circular domain with the radius of $2chord$ length as shown in Figure 4.3 is selected as a region of interest where training and reference data are obtained. It is assumed that the training data set includes the y component of the velocity field and eddy viscosity at the selected circular domain. x-component of velocity and pressure fields at the circular boundary is treated as a boundary condition to the network. As a result, the training data set is consisting of a y-component of the velocity field, eddy viscosity, and as a boundary condition x-component of velocity and pressure information at the circular boundary for each algorithm corresponding to Spalart-Allmaras and k-ω turbulence models.

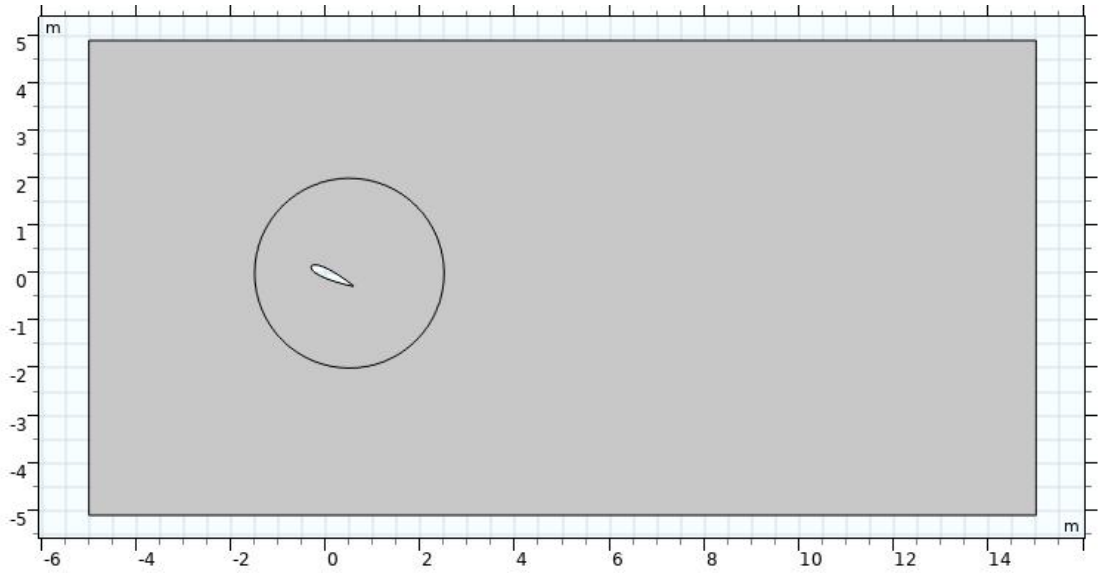


Figure 4.3 Selected circular domain as a region of interest for PINN algorithm. Also, it represents the control volume for force calculations

As training data, the y-component of the velocity field and eddy viscosity in the interest domain on the time interval of (75.96, 79.96) with a time step of 0.04 is available in the scattered form of the time and spatial coordinates as illustrated in Figure 4.4 and 4.5. In another way, training data can be explained as $(t, x, y, v, \mu_T)_{n=1}^{N}$ where n is n^{th} training data point from 1 to N, v is the y-component of the velocity field, μ_T is eddy viscosity at that point. For instance, v and μ_T at $t = 76$ are provided in

Figure 4.6 and 4.7 for the visualization. Similarly, the x-component of the velocity field and pressure field data at the circular boundary is expressed as $(t, x, y, u, p)_{m=1}^{M}$ where m is m^{th} data point from 1 to M. Total points of circular boundary data, M, is much smaller than the total data points of training data, N.

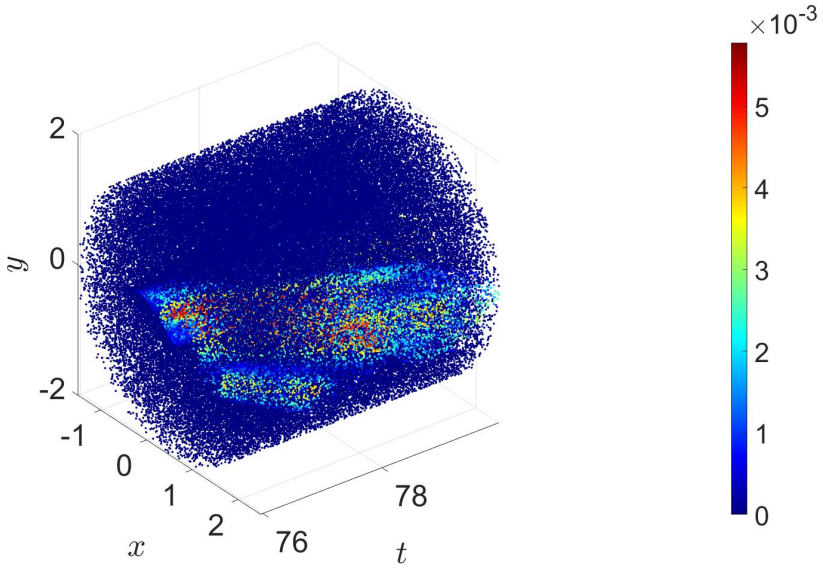


Figure 4.4 Eddy viscosity at the interest domain on the time interval of (75.96,79.96) with time step of 0.04 is illustrated in the scattered form (from the Spalart-Allmaras turbulence model).

Constructed PINN model architecture is presented in Figure 4.8. When the PINN structure is examined, it can be observed that there are two main parts. In the first part of the model, an uninformed neural network which has a feed-forward deep neural network structure is mapping a function from (t, x, y) to (μ_T, u, v, p). Automatic differentiation is applied to outputs of the uninformed neural network to feed physics-informed part for underlying physical laws by governing equations. Residuals of the governing equations in non-dimensional form, as formulated through Equation 4.11-4.13, are implemented to PINN for regression of passive scalar concentration, velocity field, and pressure field.

$$e_1 = \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - (\frac{1}{Re})(1 + \frac{\mu_T}{\mu})(\frac{\partial^2 u}{\partial x} + \frac{\partial^2 u}{\partial y}) \tag{4.11}$$

$$e_2 = \frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - (\frac{1}{Re})(1 + \frac{\mu_T}{\mu})(\frac{\partial^2 v}{\partial x} + \frac{\partial^2 v}{\partial y}) \tag{4.12}$$

$$e_3 = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \tag{4.13}$$

During the training, besides minimizing the mean square of the training data, norms

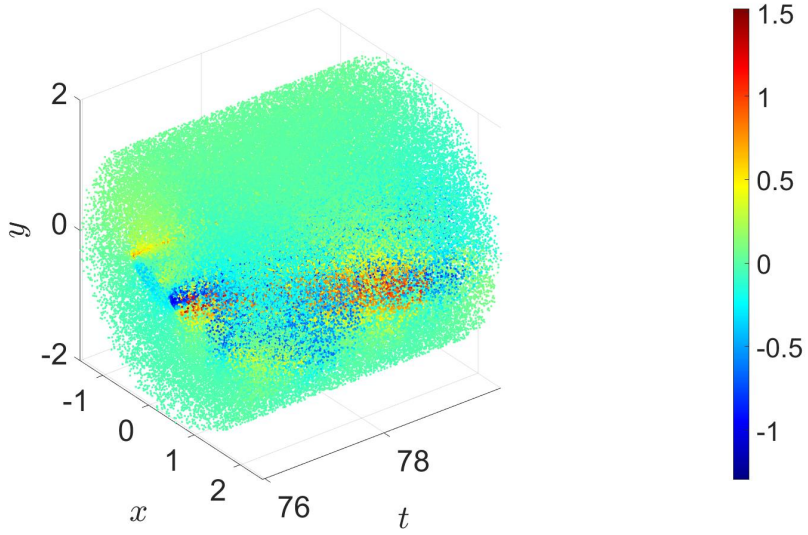


Figure 4.5 y-component of the velocity field at the interest domain on the time interval of (75.96, 79.96) with time step of 0.04 is illustrated in the scattered form (from the Spalart-Allmaras turbulence model)

of the equation residuals are minimized. Thus, physical laws underlying the problem are used by the network as a constraint to penalize predictions of hidden quantities whose training data are not available.

$$\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{eqns} + \mathcal{L}_{BC} \tag{4.14}$$

In detail, the first loss term $\mathcal{L}_{data}$ denotes the mean square error between training data and network predictions. N is the number of training data points. In Figure 4.8, red arrows show outputs which have training data.

$$\begin{aligned} \mathcal{L}_{data} = \frac{1}{N}\sum_{n=1}^{N} |\mu_{Tdata}(t^n, x^n, y^n) - \mu_{Tpred}(t^n, x^n, y^n)|^2 \\ \frac{1}{N}\sum_{n=1}^{N} |v_{data}(t^n, x^n, y^n) - v_{pred}(t^n, x^n, y^n)|^2 \end{aligned} \tag{4.15}$$

$\mathcal{L}_{BC}$ refers summation of mean square of errors of x-component velocity and pressure predictions at the circular boundary. M is the number of points at the circular boundary.

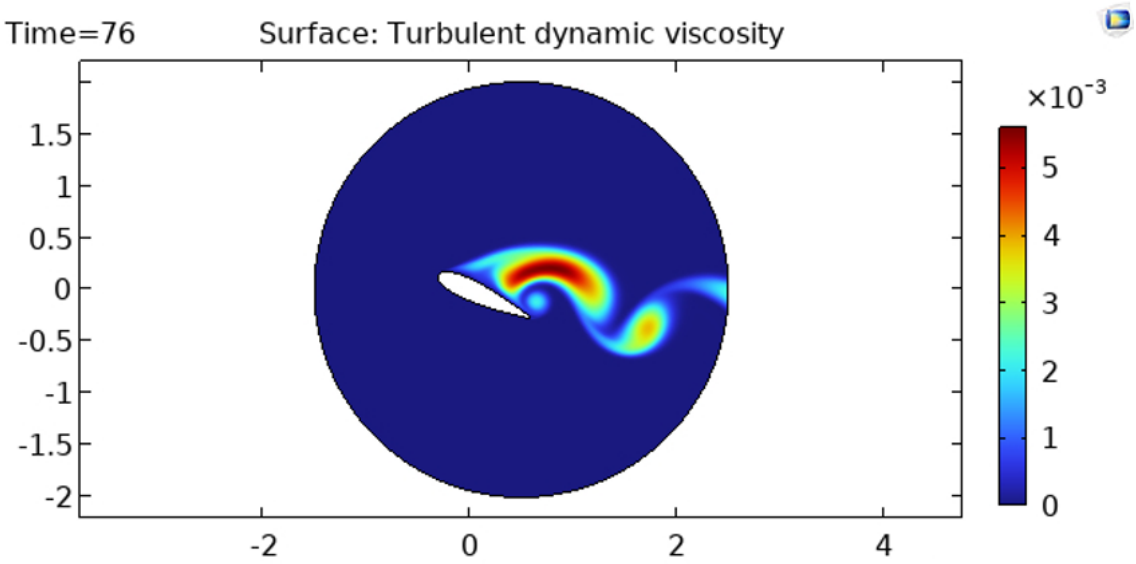


Figure 4.6 Turbulent dynamic viscosity at t = 70 on training domain is visualized (from Spalart-Allmaras turbulence model).

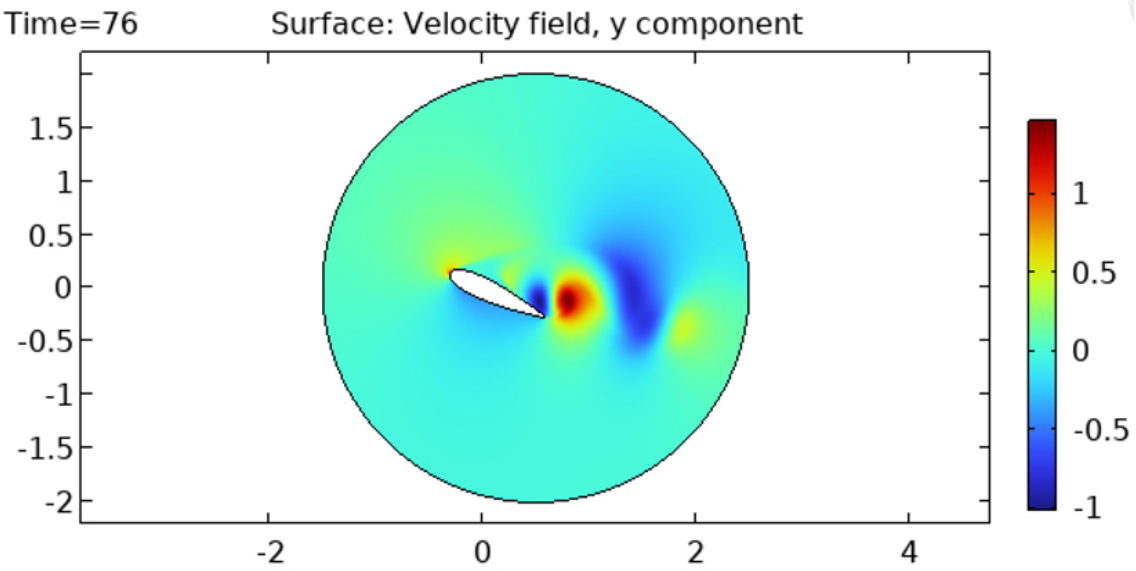


Figure 4.7 y-component of the velocity field at t = 70 on training domain is visualized (from Spalart-Allmaras Turbulence model).

$$\mathcal{L}_{BC} = \frac{1}{M}\sum_{m=1}^{M} |u_{data}(t^m, x^m, y^m) - u_{pred}(t^m, x^m, y^m)|^2 + |p_{data}(t^m, x^m, y^m) - p_{pred}(t^m, x^m, y^m)|^2 \tag{4.16}$$

The last term of the loss function, $\mathcal{L}_{eqns}$, calculates the norm of the equation residuals. In other words, $\mathcal{L}_{eqns}$ is the mean square error between equation residuals and zero since all the terms of equations are gathered on the left hand side of the

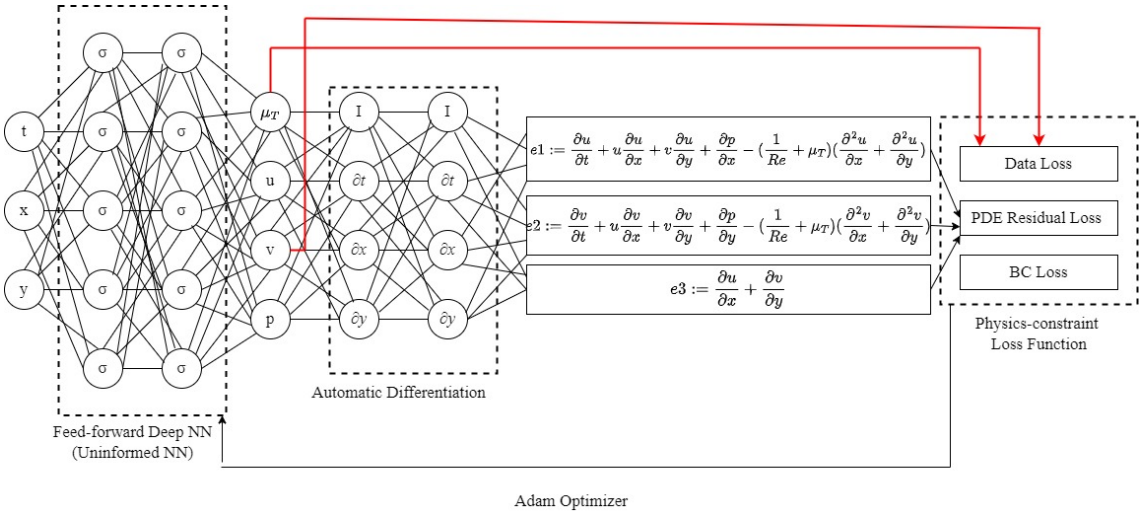


Figure 4.8 Physics informed neural network architecture

equations, it is aimed to minimize equation residuals through zero). The finite set of residual points where equations are penalized can be at different numbers and located differently from the training data set. The finite set of residuals is also known as collocation points. In this study N and J are accepted as equal.

$$\mathcal{L}_{eqns} = \sum_{i=1}^{3} \frac{1}{J} \sum_{j=1}^{J} |e_i(t^j, x^j, y^j)|^2 \tag{4.17}$$

In the following section, a parametric study is presented for finding the best-fit network parameters for the training. Besides, weight normalization is applied in order to accelerate the optimizer. Accordingly, the weight vector is parameterized, and thus decoupling is achieved between the vector's length and direction. Weight normalization is a method that is inspired by batch normalization. The reason why batch normalization, which is widely used in deep learning, cannot be applied in PINN is that network has a physics-based interface.

4.4 PINN Parametric Study

In this section, before applying the physics-informed neural network to the problem, hyper-parameter tuning is studied to reach the best-performed network parameters for the training. Tuned parameters are activation function, number of layers, number of nodes per layer, learning rate, and mini-batch size. A parametric study is applied manually. While other parameters are kept constant, each network with a different value of the relevant parameter within a range is trained by $5 \cdot 10^4$ iterations. The parametric study is performed on the data coming from the Spalart-Allmaras turbulence model and selected parameters are applied for both PINN algorithms.

4.4.1 Activation Function

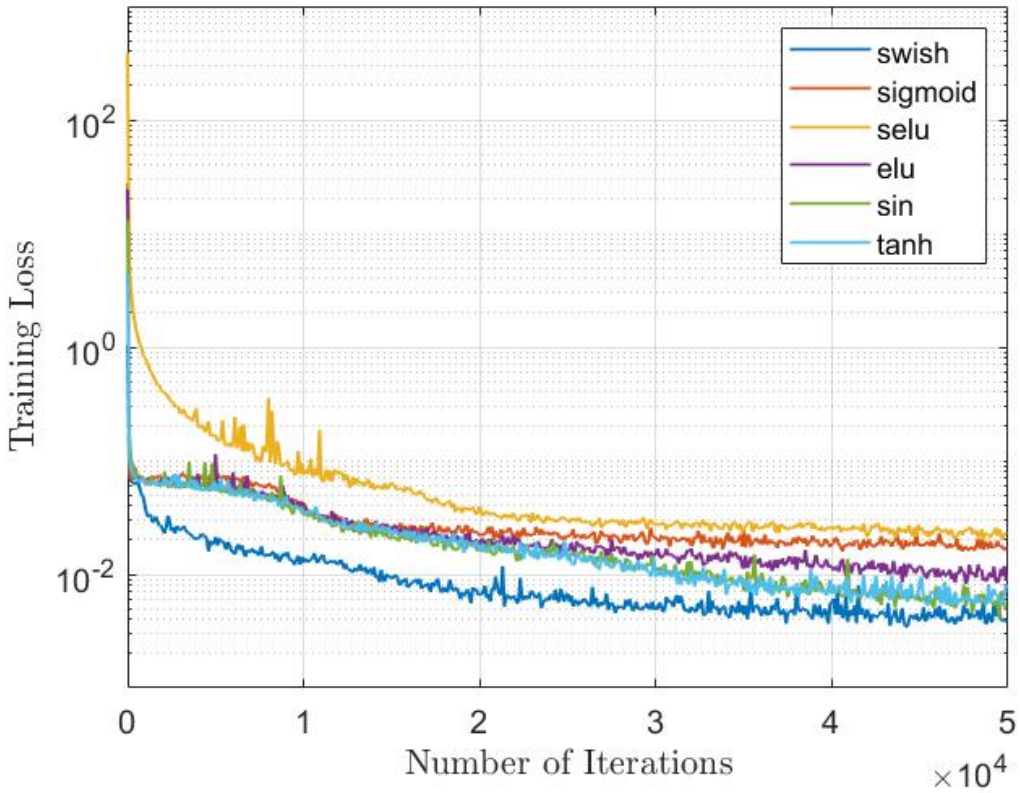


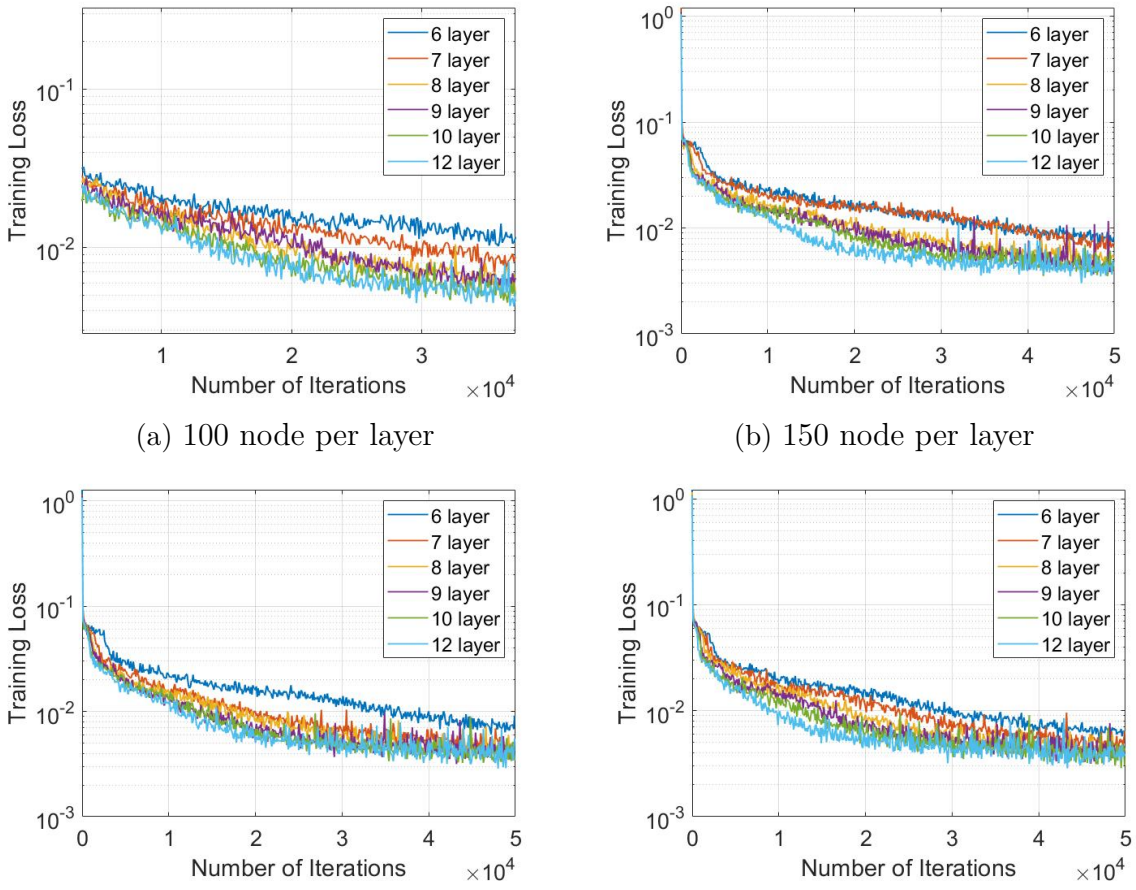
Figure 4.9 MSE loss for different activation functions

In order to find the best fit activation function, *swish*, *sigmoid*, *selu*, *elu*, *sin*, and *tanh* are implemented in the PINN training for $5 \cdot 10^4$ iterations. Mean square error training loss results are presented in Figure 4.9. The frequently used *sigmoid* function in feed-forward deep neural networks is performed worst in the PINN model. It has not a spectacular effect since outcomes of the *sigmoid* change slightly as approaches the input boundaries. That is why small gradients appear in those regions and learning is stopped by the vanishing gradient problem. Since derivatives are multiplied during the backpropagation and a very small gradient, i.e., near zero, causes a zero result for the whole process. Thus, network parameter cannot be updated. On the other hand, *elu* and *selu* as variants of the *relu* activation function are performed poorly as well. Even though *selu* did not suffer from the vanishing gradient problem, it has not ended up with successful PINN training. In the literature, it is shown that one of the activation function requirements for a well-trained PINN is smoothness. Since *relu* (Equation 3.12) and their variants show non-smooth behavior, convergence is not observed [44]. The *sin* and *tanh* are differentiable activation functions and as seen they perform better. The best-performed function is the swish activation function which is another differentiable

one and implemented for full training in this study.

4.4.2 Number of layers and number of nodes

In order to find the best-performed network architecture, different networks from the depth of 6 to 12 and the number of nodes per layer from 100 to 250 are implemented. For each case, training is limited by 5.10^4 iterations and other hyper-parameters are kept the same. Figure 4.10 shows that as the number of layers increases, learning capacity is increasing in the case of any number of nodes per layer as well. Networks with the same number of layers but having a higher number of nodes on them show better learning performance. The effect of those parameters on the model prediction shows consistent behavior with the training as seen from Table 4.1. As a result of L2 error comparison, 12 layers and 250 nodes per layer are chosen as best fit parameters for the investigation of turbulent flow over a stationary airfoil.



(a) 100 node per layer

(b) 150 node per layer

(c) 200 node per layer

(d) 250 node per layer

Figure 4.10 MSE training loss comparison for number of nodes per hidden layer optimization

Table 4.1 L2 error norm between model predictions and reference outputs for PINNs in different depth and width

Nodes per layer	6 layer	7 layer	8 layer	9 layer	10 layer	12 layer
100	0.0708	0.0494	0.0375	0.0363	0.0332	0.034
150	0.0552	0.0478	0.0367	0.0349	0.0327	0.0292
200	0.0487	0.0356	0.0327	0.0298	0.0289	0.0299
250	0.0708	0.0494	0.0375	0.0363	0.0332	0.034

4.4.3 Learning Rate

For the PINN training, the Adam optimizer which is a variant of the stochastic gradient descent algorithm is applied. Rather than a learning rate, all optimizer parameters are kept on default parameters. For 4 different learning rate values ranging from 10^{-2} to 10^{-5}, PINN is performed with $5 \cdot 10^4$ iteration limited training. Figure 4.11 presents loss function behaviours for each case. Training process is slow at small learning rates as expected. At the beginning of training, minimization of the loss function occurs most quickly with a learning rate of 10^{-2}. However, it is known that learning happens fast in the first stages of the training and then the rate of change in learning becomes smaller. Completing training with a high learning rate can cause a convergence problem. Additionally, 10^{-2} shows a faster learning process but it has oscillatory behavior which points out that changes in the network weights are high than they should be. The minimization of the loss function is greater for a learning rate of 10^{-3}. The learning rate of 10^{-3} shows a sharp learning curve. As a result of the parametric study, the learning rate is accepted as 10^{-3} for turbulent flow over a stationary airfoil problem.

4.4.4 Batch Size

As seen from Figure 4.12, learning process speeds up as mini-batch size increases. Additionally, the loss function shows relatively less fluctuation in higher mini-batch sizes. Besides the training performance, the model prediction accuracy for all cases is compared as well. Table 4.2 shows L2 error between model outputs and reference values. Accordingly, the error is highest when the mini-batch size is the smallest, i.e., 512. Up to 8192, the error is decreasing and it reaches the lowest error at 8192. As a result of the training and prediction capabilities of networks in different mini-batch sizes, 8192 is decided to be applied for full training of the problem.

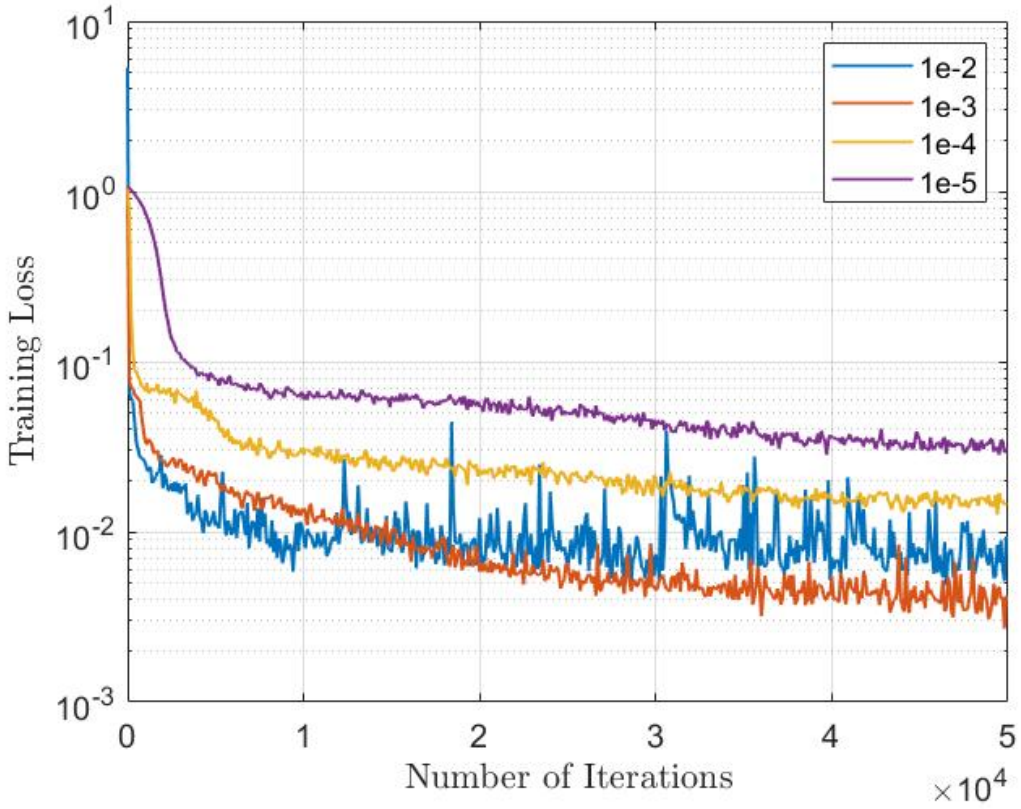


Figure 4.11 MSE loss for different learning rates

Table 4.2 L2 error norm between model predictions and reference outputs for PINNs in different mini-batch sizes

Batch Size	512	1024	2048	4096	8192
L2 norm	0.0364	0.0317	0.0305	0.028	0.0268

4.5 Results

Selected hyper-parameters according to parametric study and number of input samples are summarized in Table 4.3. This section represents PINN results for both Spalart-Allmaras and k-w turbulence models after the training with 10^6 iterations. The network is trained and tested on the HPC with an Intel Xeon Scalable Gold 6148 processor and a single Nvidia Tesla V100 (16GB, NVLink) GPU. Generally, in deep learning, an epoch refers to one pass through the entire data set and is used as a training metric. PINN differs from conventional deep learning techniques since a physics-informed neural network deals with different data sets of different sizes (i.e., available training data points and collocation points for equation residuals) at the same time. That is why iteration is used as a training metric here rather than epoch.

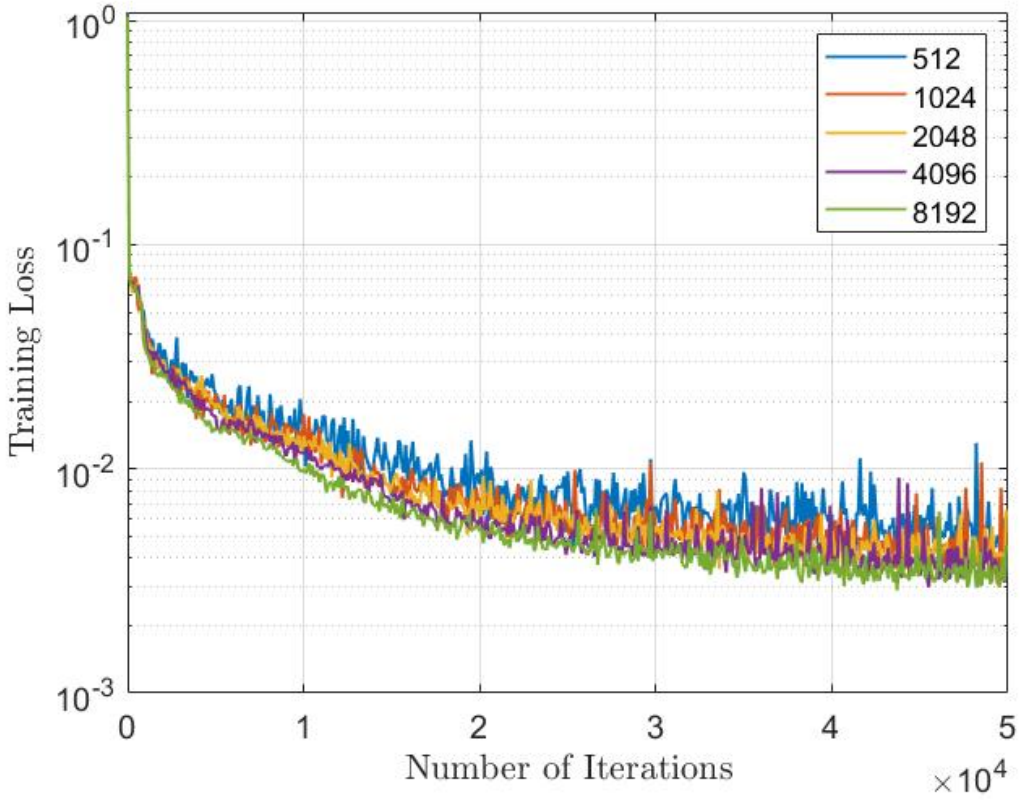


Figure 4.12 MSE loss for different mini-batch sizes

4.5.1 PINN Results

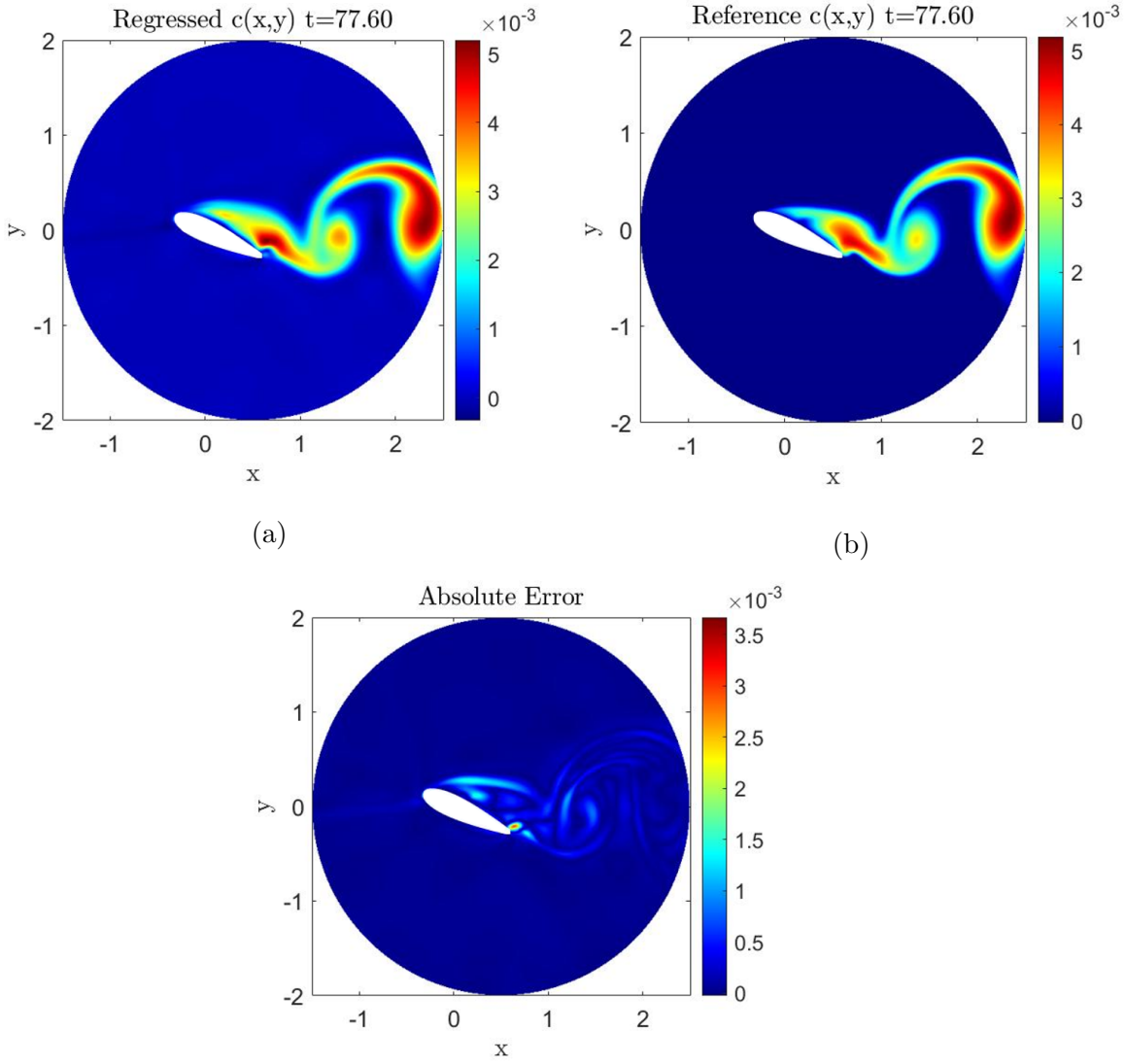
PINN Results of Spalart-Allmaras Turbulence Model

As seen from Figure 4.13, eddy viscosity (dynamic turbulent viscosity) is regressed successfully, and absolute error between the reference values and model predictions are on the order of 10^{-3}. From the definition of eddy viscosity, it represents the transport and dissipated energy of turbulence modeling therefore it is expected to have higher values at vortexes. PINN algorithm accurately catches this dissipation. Higher error occurs at backflow region because of the complex fluid motion, solving Navier-Stokes equations is becoming challenging by the PINN algorithm. On the other hand, since eddy viscosity data is available and it feeds to the network during the training, it is straightforward to have accurate results for eddy viscosity. Similarly, 4.14 shows that the y-component of the velocity field is regressed successfully as well.

For a better understanding of the capability of the algorithm, the x-component of the velocity and pressure fields are separately investigated in the same time frame with the represented temperature field in Figure 4.13 over the same data points. Figure 4.15-4.16 illustrates PINN regression on x component of the velocity and

Table 4.3 PINN Summary

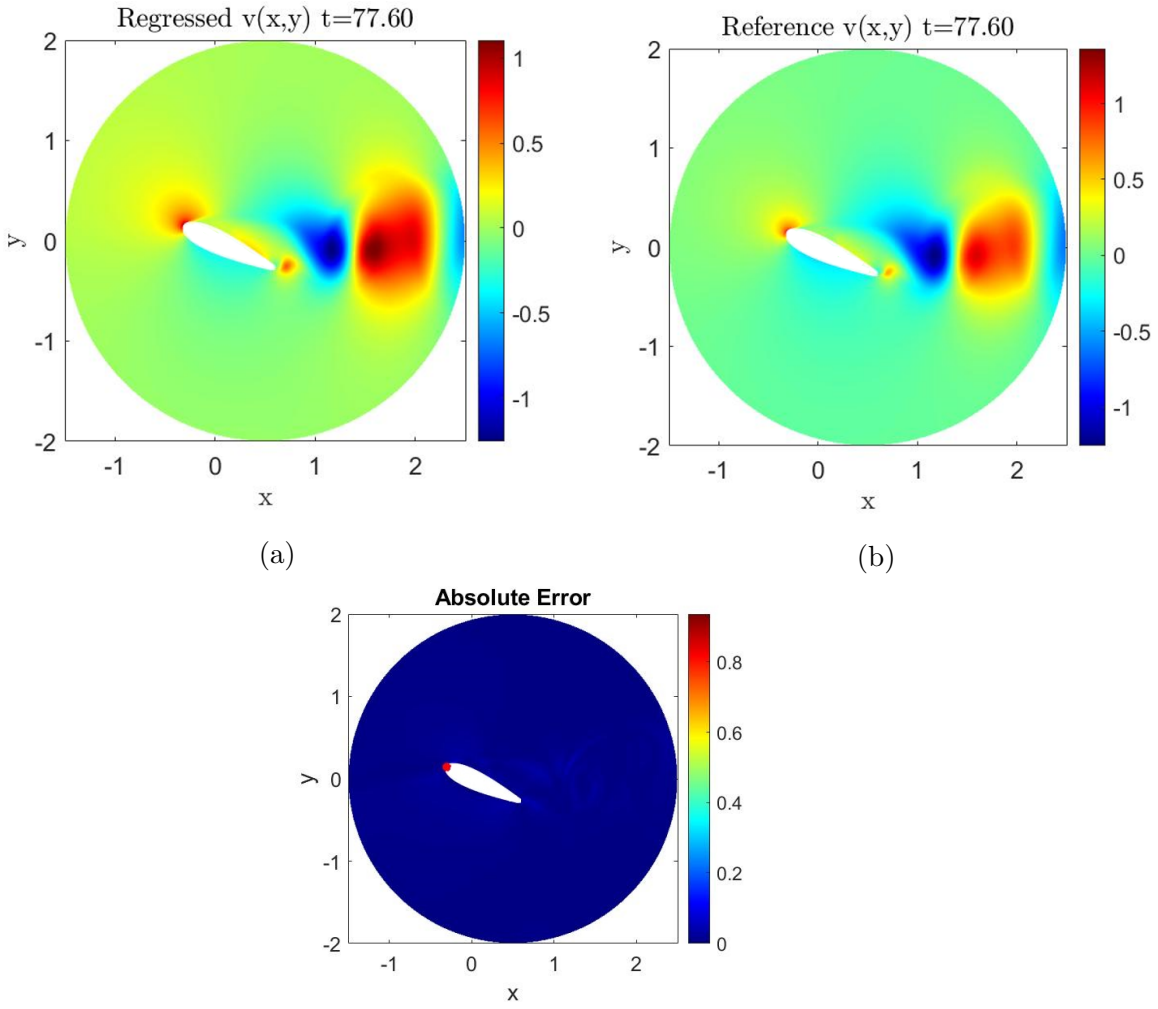
Activation Function	Swish
Batch size	8192
Learning rate	1e-3
Optimizer	Adam
Training time (in iterations)	1 million
Number of Layers	12
Node number per layer	250
Number of data points	23422
Number of collocation points	23422
Number of total time frame	101
Number of time frame (train-test)	81-20



(a) (b)

(c)

Figure 4.13 Dynamic turbulent viscosity at t=77.6 a)reference field b)regressed field by PINN c)absolute error



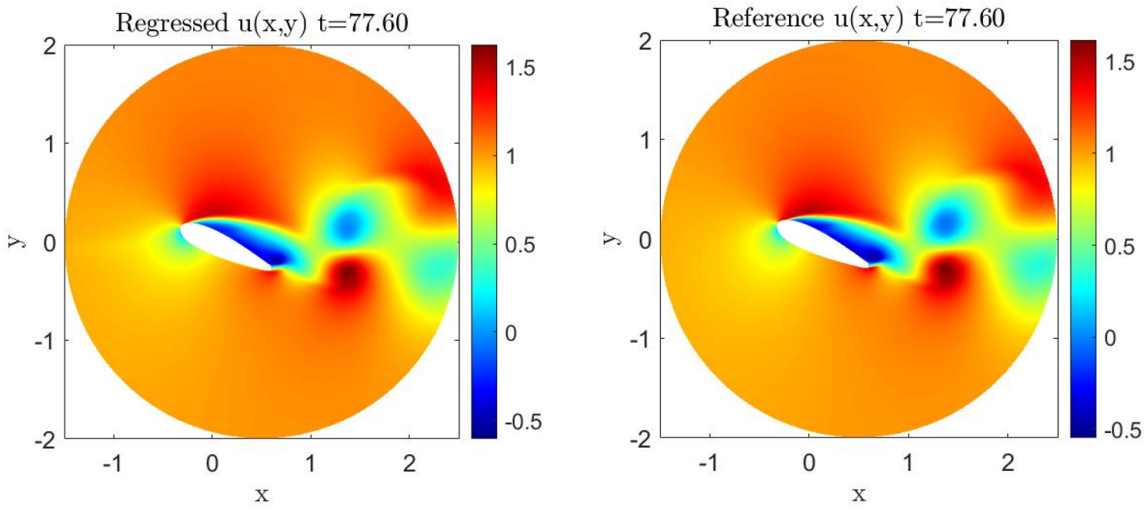
(a) (b)

(c)

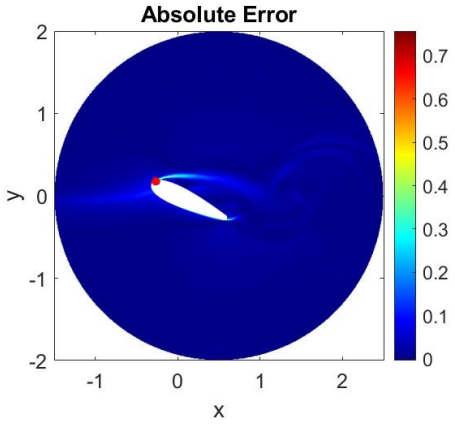
Figure 4.14 y-component of the velocity field at t=77.6 a)reference field b)regressed field by PINN c)absolute error. Maximum error is labelled by red dot.

pressure fields respectively. It is known that flow is characterized by the Reynolds number of 50000 as explained in the numerical assessment Section 4.2. In this flow regime, as seen from reference velocity field components 4.15b and 4.14b,algorithm is capable of catching fluid separation from top of the airfoil. Besides, eddies of different sizes at the wake of the airfoil are regressed accurately in terms of both the velocity magnitude and size. At the inside of the eddies, flow is circulating so some flow is in the opposite direction to the free stream. PINN algorithm performs successfully at the reverse flow as well.

Asymmetric flow occurs when flow separation starts from the top of the airfoil. Thus, the pressure distribution is changed and results in aerodynamic forces which occur on the airfoil. Figure 4.16 shows good agreement between the PINN pressure prediction and reference pressure field. Note that, only pressure at the circular boundary is introduced to the PINN algorithm for the sake of a unique solution.



(a) (b)

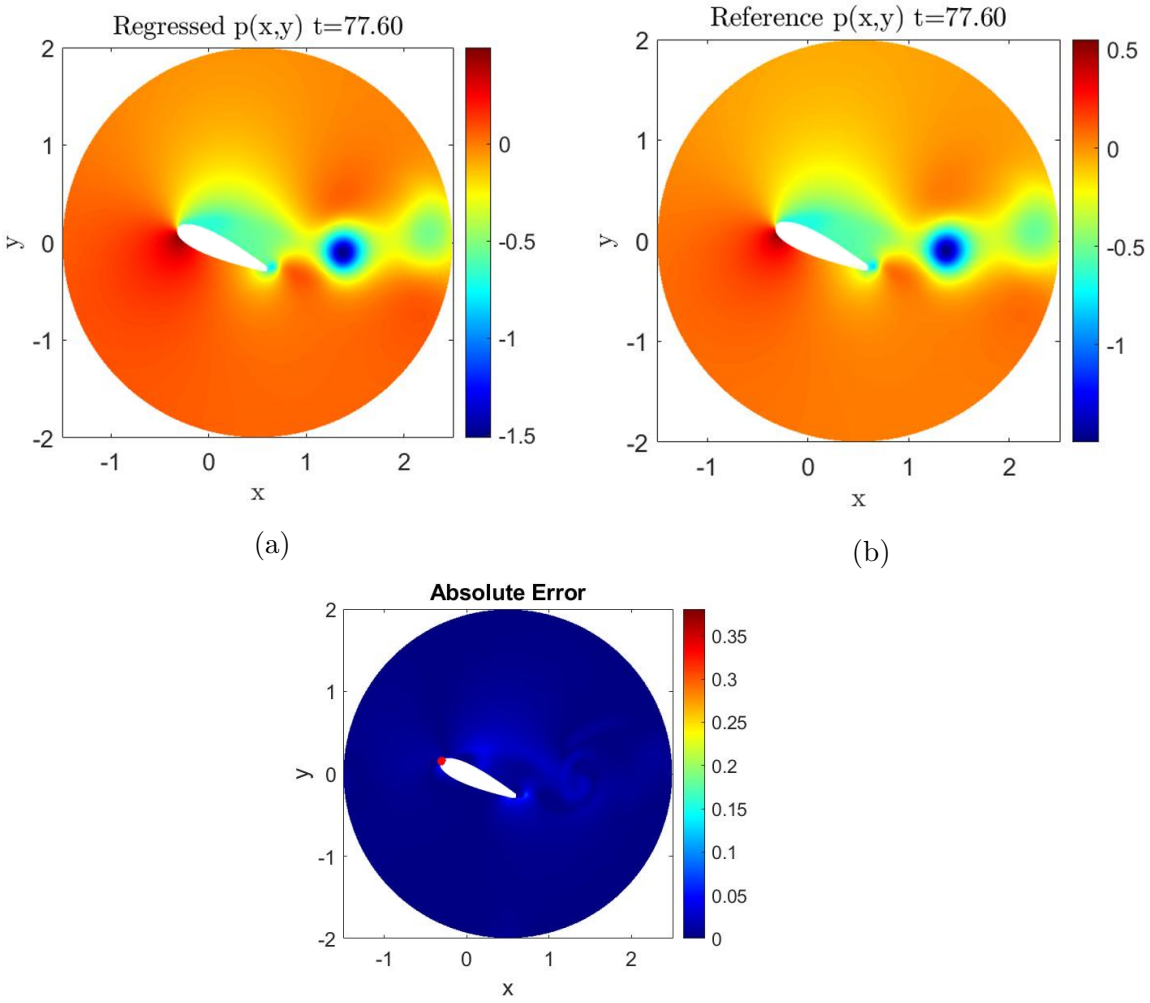


(c)

Figure 4.15 x-component of the velocity field at t=77.6 a)reference field b)regressed field by PINN c)absolute error. Maximum error is labelled by red dot.

Therefore, it is assumed that any information from pressure is not available to feed the network rather than the boundary condition. Pressure is just extracted from the approximate solution of Navier Stokes equations for incompressible flow by PINN. If the presented absolute error of the pressure field in Figure 4.16 is compared with absolute errors of the velocity field, it can be seen that the error for pressure over the domain is similar to velocity field errors. Even though only the gradient of the pressure is included in incompressible Navier-Stokes equations rather than a direct pressure term, pressure regression performance is magnificently accurate.

For illustrative purposes of PINN algorithm performance, regression of velocity and pressure fields on single time (t = 77.60) is provided above figures. At whole time window of [75.96, 79.96], relative L2 errors for all output field on the interest domain are provided in 4.17. As shown from the figure, algorithm prediction capability for each output shows similar characteristics over the time windows. In other words,



(a) (b) (c)

Figure 4.16 Pressure field at t=77.6 a)reference field b)regressed field by PINN c)absolute error. Maximum error is labelled by red dot.

prediction of each output is less accurate at the beginning and end of the time intervals. It show a kind of history effect there. Lack of data at just before the start and just after the end time of the time windows affect predictive capability imperfectly. Additionally, it is observed that higher error occurs at t $>$ 79 since approximating Navier-Stokes equations at that time is challenging by the PINN algorithm due to the complex fluid motion.

PINN Results of $k-\omega$ Turbulence Model

Figure 4.18 shows relative L2 error of PINN predictions for $k-\omega$ turbulence model. When results are compared with Spalart-Allmaras case which is presented in Figure 4.17, it can be seen that error is reduced for all the outputs. Especially error order is reduced by ten-fold for u, p and c. When error characteristics is examined, it is observed that high error at the beginning and end of the time window as similar with Spalar-Allmaras case. However, in addition to tip points, error peak is

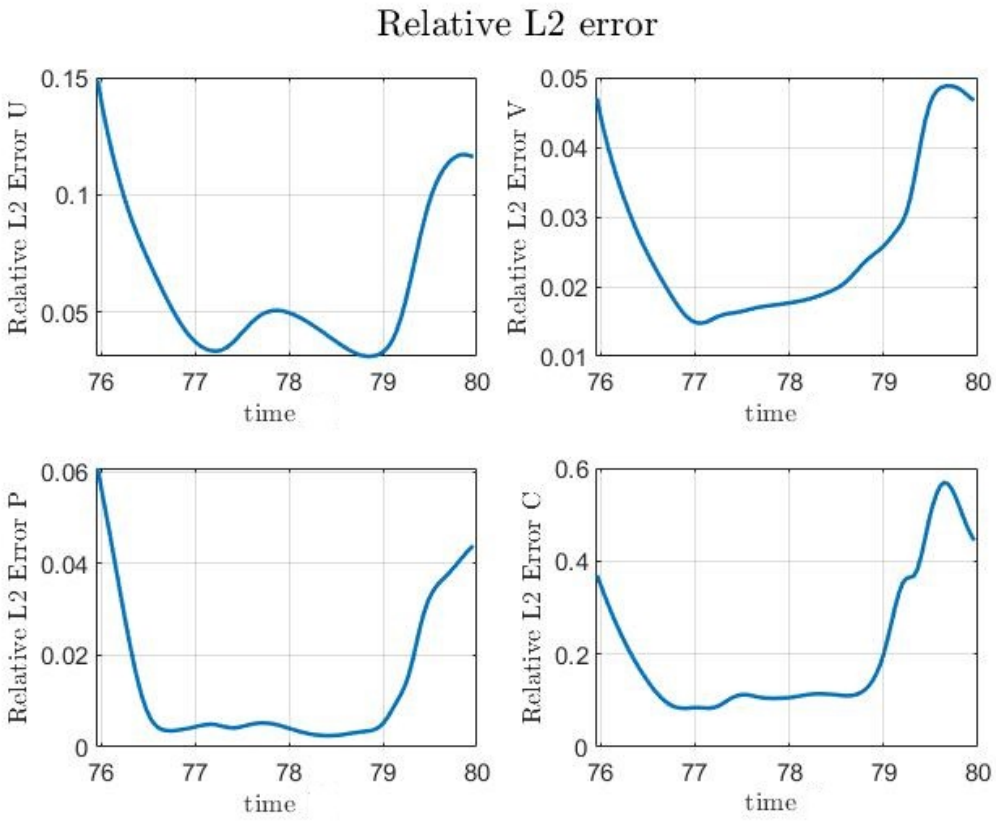


Figure 4.17 Relative L2 error of PINN algorithm for Spalart-Allmaras turbulence model outputs

occured around t = 78. When fluid motion is examined at t = 78, separation from tip of the airfoil is observed. PINN algorithm is suffered by prediction of outputs at t = 78 due to the separation. As a result, although same pinn algorithm is used regressed outputs for both turbulence models, PINN performance is different. Accuracy is higher for the $k-\omega$ case. The reason behind is that during the training eddy viscosity is fed to the network. Eddy viscosity is a variable shows differences according to turbulence models. It shows that different type of training data in terms of distribution and magnitude affects the learning capability and thus the prediction accuracy.

4.5.2 Force Calculations

The x component of velocity and pressure fields are hidden quantities that are extracted from PINN by training on eddy viscosity and y component of the velocity field and underlying physical laws as a constraint for regression. Regressed fields are used to calculate lift and drag forces on the airfoil. Aerodynamic forces are obtained from both stress field integration method and control volume approach. Detailed description of the derivation of these forces is provided in Section 2.1.2.

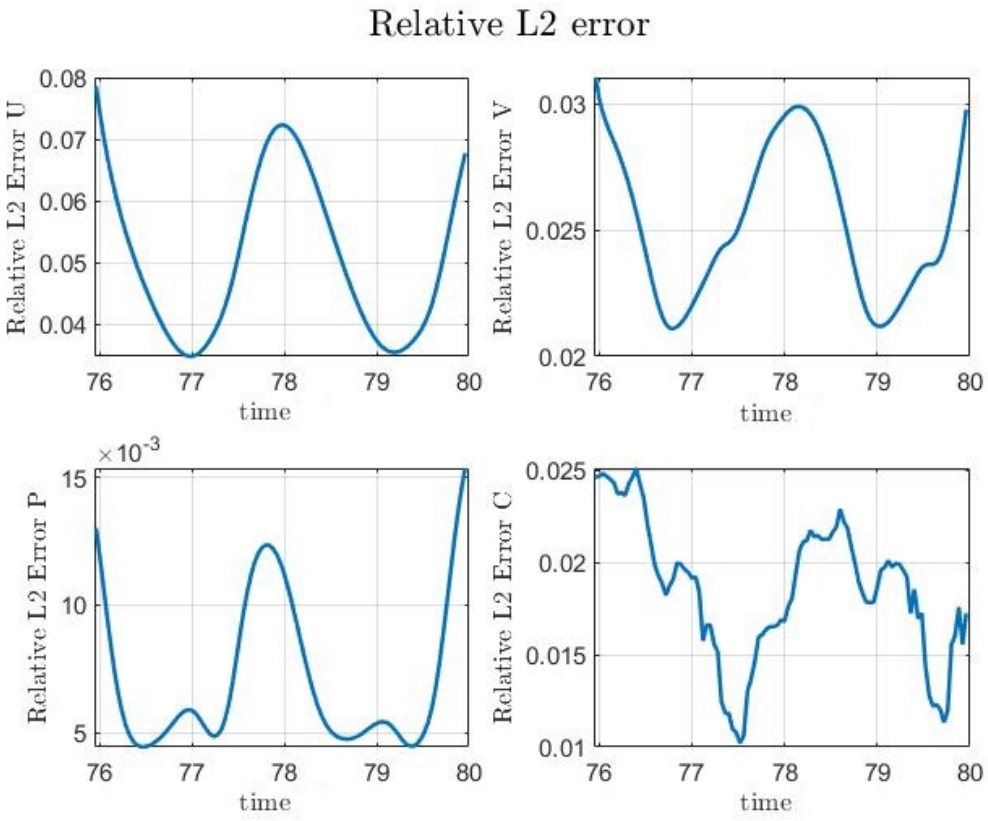


Figure 4.18 Relative L2 error of PINN algorithm for $k-\omega$ turbulence model ouputs

Since drag force is in parallel to the upstream velocity, it is calculated from the integration of stress field in the x direction as shown in Equation 4.18. Similarly, since the lift force is perpendicular to the upstream velocity, it is calculated from the integration of stress field in the y direction as shown in Equation 4.19.

$$F_D = \int (-pn_x + 2\,\mu \frac{\partial u}{\partial x}\, n_x + \mu\, (\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y})\, n_y)\, dA \tag{4.18}$$

$$F_L = \int (-pn_y + 2\,\mu \frac{\partial v}{\partial y}\, n_y + \mu\, (\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})\, n_x)\, dA \tag{4.19}$$

As an alternative approach for aerodynamic force calculation, conservation of linear momentum for a fixed control volume is indicated as below.

$$\sum F = \frac{d}{dt}(\int_{CV} V\rho\, d\mathcal{V}) + \int_{CS} V\rho(V \cdot n)\, dA \tag{4.20}$$

Equation 4.20 is a vector equation since all terms include the velocity vector V. $\sum F$ is the vector sum of forces acting on the control volume. In order to calculate the lift and drag forces acting on the airfoil in the control volume, x and y components

of Equation 4.20 should be written while considering pressure and viscous forces as shown below.

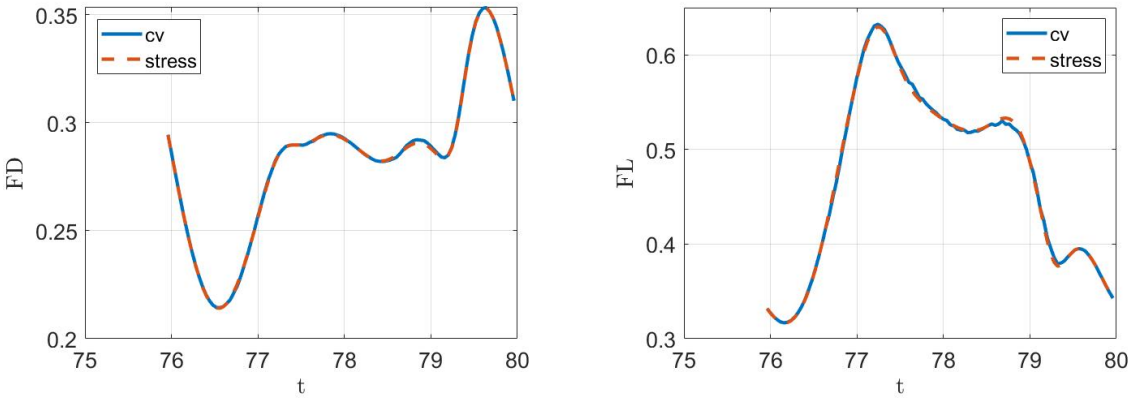
$$F_D = \frac{d}{dt}(\int_{CV} u\ \rho\ d\mathcal{V}) + \int_{CS} u\ \rho\ (un_x + vn_y)\ dA + \int_{CS}(pn_x)\ dA - \int_{CS}[(\mu\ 2\frac{\partial u}{\partial x}n_x) + (\mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})n_y)]dA \tag{4.21}$$

$$F_L = \frac{d}{dt}(\int_{CV} v\ \rho\ d\mathcal{V}) + \int_{CS} v\ \rho\ (un_x + vn_y)\ dA + \int_{CS}(pn_y)\ dA - \int_{CS}[(\mu\ 2\frac{\partial v}{\partial y}n_y) + (\mu(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x})n_x)]dA \tag{4.22}$$

Force Calculation from PINN model for Spalart-Allmaras Turbulence Model

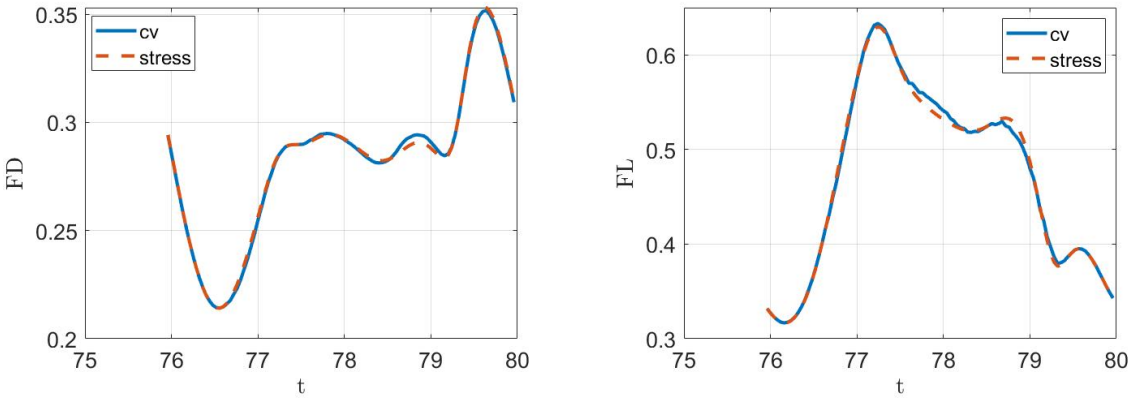
In order to compare the methods, both methods were applied on the numeric solution in COMSOL. As seen from Figure 4.19, both methods are in a good agreement with each other for lift and drag force calculations. It is important to perform this validation in order to show that the methods are implemented correctly. Viscous forces are taken into consideration for aerodynamic force calculation via control volume approach as seen from from last terms of Equations 4.21 - 4.22. In order to see the effect of viscous terms at the boundary, they are discharged from equations and validation study is repeated as presented in Figure 4.20. In both cases, the viscous effect is so small therefore it can be neglected for computational efficiency to avoid calculations of unnecessary gradients. Therefore, viscous terms are neglected when aerodynamic forces are extracted from PINN predictions via control volume approach.

Using the velocity and pressure field predictions obtained from the PINN approach, lift and drag forces acting on the airfoil are calculated in two ways through the stress integration and the control volume methods. Figure 4.21 shows drag force calculation from the PINN approach. As shown in the figure, stress integration method perfors much better than control volume approach. Beside the offset in the amplitude, oscillatory behaviour of the drag force is partially caught by the control volume approach. When the performances of each method is compared in terms of mean relative absolute error, it is 4.97% for the stress integration method and 13.04% for the CV method. For both methods, at the very beginning (t=75.96) and at the end (t=79.96) of the time window the relative absolute error is maximum, about 26% and 36% for SI and CV methods respectively. The higher error there can be explained by the lack of training data at t<75.96 and t>79.96.



(a) (b)

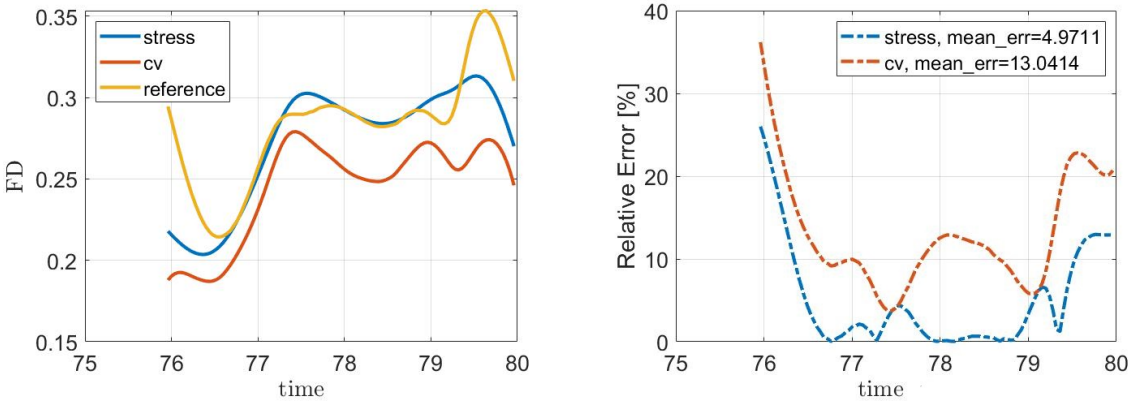
Figure 4.19 Aerodynamic force calculation by SI and CV methods on reference data of a full cycle. CV approach includes viscous term at the boundary for a)drag force b)lift force



(a) (b)

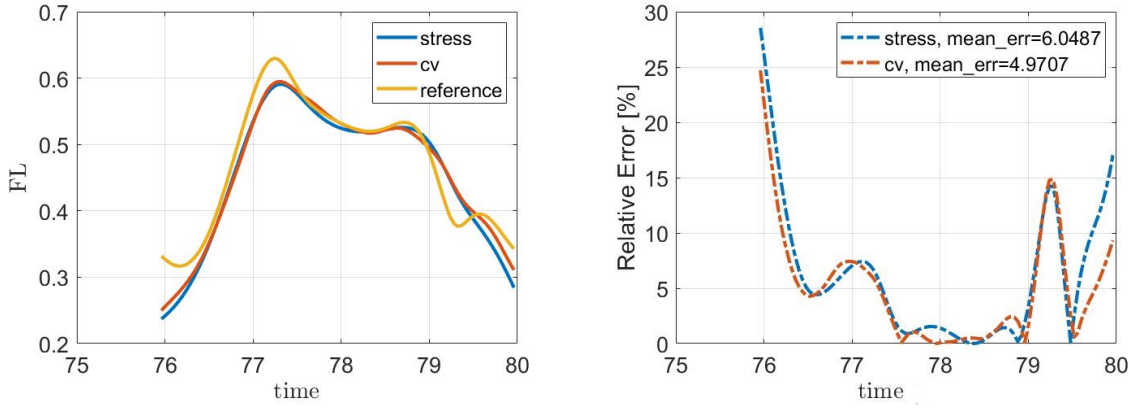
Figure 4.20 Aerodynamic force calculation by SI and CV methods on reference data of a full cycle. CV approach does not include viscous term at the boundary for a)drag force b)lift force

Figure 4.22 shows the lift force results from the PINN approach. There is no significant difference between the accuracy of SI and CV methods. When the performance of these methods are compared in terms of mean relative absolute error, we have 6.04% for the SI method and 4.97% for the CV method. In terms of mean relative absolute error, CV performance is slightly better. As seen from the figure, initial and final time frames are the highest error regions and CV method performs better. Here, it can be said that the two methods show very close performances and good at catching oscillatory behaviour in reasonable amplitudes.



(a) (b)

Figure 4.21 Drag force extraction from PINN model for Spalart-Allmaras turbulence model a) method comparison b) corresponding relative absolute errors



(a) (b)

Figure 4.22 Lift force extraction from PINN model for Spalart-Allmaras turbulence model a) method comparison b) corresponding relative absolute errors

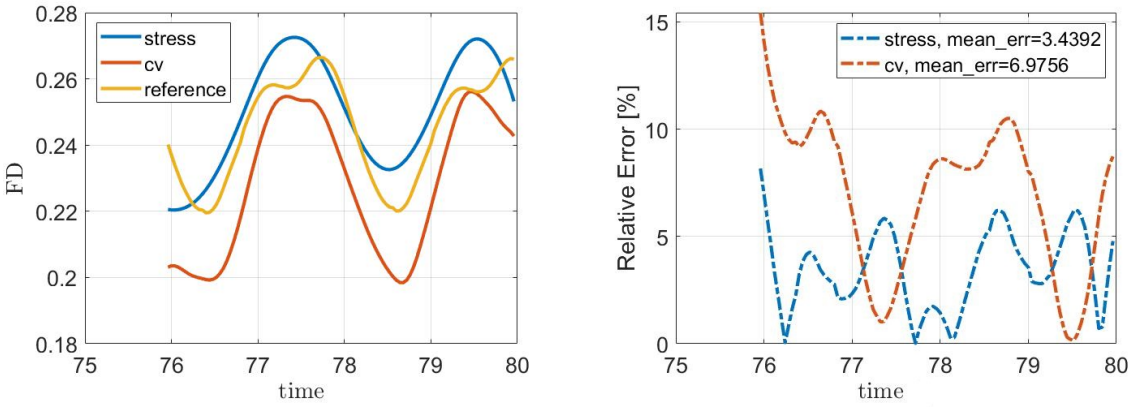
Force Calculation from PINN model for $k-\omega$ Turbulence Model

Similarly to the Spalart-Allmaras case, force calculation from the PINN model is done through both integration of the stress field over the airfoil and a control volume around the airfoil. While applying the control volume approach, the viscous term is neglected as well.

Figure 4.23 shows drag force prediction from regressed outputs. As shown in the figure, the stress approach performs better than the control volume approach. The mean relative absolute error for the SI method is 3.43% and 6.97% for the CV method. SI accuracy is almost twice better. Moreover, Figure 4.24 presents the lift force predictions. Lift force predictions with each method are better than the drag force. Even though both methods perform similarly, stress approach accuracy is slightly higher than the control volume. As seen from the relative absolute error

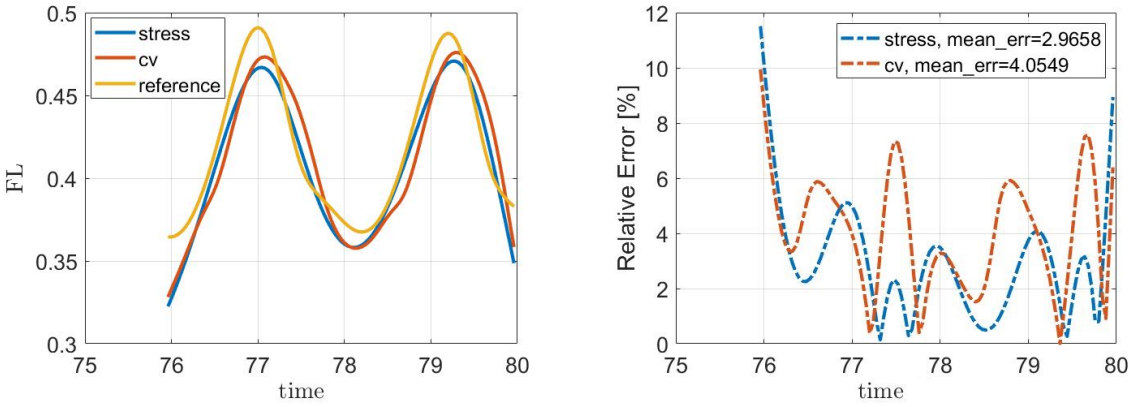
graph, the mean relative absolute error for the SI method is 2.96% and 4.05% for the CV method.

On the other hand, it is possible to compare the aerodynamic force predictions of PINN models for each turbulence model. Overall, force prediction accuracy is higher with the PINN model for $k-\omega$ turbulence model outputs. Since outputs are predicted more accurately in $k-\omega$ case, it is expected to have higher accuracy at force calculation as well.



(a) (b)

Figure 4.23 Drag force extraction from PINN model for $k-\omega$ turbulence model a) method comparison b) corresponding relative absolute errors



(a) (b)

Figure 4.24 Lift force extraction from PINN model for $k-\omega$ turbulence model a) method comparison b) corresponding relative absolute errors

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Obtaining flow data is one of the most difficult problems in fluid dynamics. Often large and detailed datasets are needed to analyze the flow and calculate forces accurately. For many years, conventional numerical and experimental methods have been used to better understand fluid dynamics. Numerically, the partial differential equation-based Navier Stokes equation as a governing equation of fluid dynamics is solved under some assumptions. Specifically, solving the Navier Stokes equations for turbulent flow is challenging due to lack of mathematical theory, computational limitations, and high number of degrees of freedom. On the other hand, obtaining data experimentally is also difficult due to experimental limitations, i.e., uncertainty of experimental equipment. Generally, the data obtained from experiments are insufficient, sparse, and noisy. Reconstruction of sparse data or missing data either from experiments or numerical studies becomes a crucial problem. In this thesis, in order to tackle this problem, physics-informed neural network is applied as an alternative approach to the conventional methods [16]. For this purpose, residuals of the governing equations as physical constraints are implemented to the loss function of a neural network for the training.

The applicability of the PINN for solving Navier-Stokes equations for laminar flow over a cylinder is demonstrated. Firstly, a 2D time-dependent CFD study for the problem is conducted by COMSOL software. The sparse data set obtained from a numerical solution for the PINN approach includes temperature field at the region of interest and velocity data at the inlet as a boundary condition. In order to reconstruct the velocity field and pressure fields on the domain, residuals of the transport equation, Navier-Stokes equation, and continuity equation are formed as

an additional loss term in the loss function. As a result, velocity field and pressure fields are obtained with high accuracy via the PINN.

Lift and drag forces acting on the cylinder are calculated from PINN predictions. Two different methods are applied to calculate aerodynamic forces. The first method relies on the integration of the stress field over the cylinder. Whilst the second one is the control volume approach applied for the conservation of momentum. Shortly, SI and CV respectively. While SI is a differentiation-based method, CV is an integral-based method. Necessary derivatives are calculated by automatic differentiation of Tensor Flow. Performances of these methods are similar. When drag force prediction is examined, it can be said that oscillation of the force is detected but there is an offset in the amplitude. On the other hand, SI and CV methods show similar accuracy for lift force prediction. Lift force prediction is better than the drag force.

Subsequently, the PINN approach is applied for turbulent flow over a stationary NACA0018. A time-dependent CFD study is conducted on 2-Dimensional domain. Two different turbulent models such as Spalart-Allmaras and $k-\omega$ are used. The sparse data set for the PINN obtained from numerical study includes eddy viscosity which is used to model turbulence and the y-component of the velocity field at the circular domain as a region of interest and boundary condition at the circular boundary. Navier Stokes and continuity equations are solved by PINN and as a result, x-component of the velocity and pressure fields are obtained. Results show that PINN for $k-\omega$ turbulent flow is better at the reconstruction of the data. In other words, the model prediction of PINN for $k-\omega$ model is more accurate than PINN for the Spalart-Allmaras model. One of the biggest impacts is eddy viscosity data. It can be stated that eddy viscosity data of $k-\omega$ model has more useful information than the Spalart-Almaras model.

In similar with the cylinder problem, aerodynamic forces acting on the airfoil are calculated by both SI and CV methods. Results of lift force calculation of PINN for Spalart-Allmaras model reveal that SI method's accuracy is almost three times higher than CV method's. For drag force calculation, the CV method is better than the SI method. On the contrary, the result of the drag force calculation of PINN for $k-\omega$ model shows that the SI method's error is two times more than the error of CV method. For the lift force prediction, the error for the SI method is slightly less than CV method. Overall, the aerodynamic force calculation of the PINN approach for $k-\omega$ is more accurate than the Spalart-Allmaras model as summarized in Table 5.1. In regard to force calculation strategies, it can be said that the SI method is preferable considering less error production.

Table 5.1 Force calculation summary that provides the relative mean absolute error of stress integration and control volume methods on drag and lift force calculations.

	Stress Integration		Control Volume	
	FL	FD	FL	FD
Spalart-Allmaras	6.04%	4.97%	4.97%	13.04%
k-ω	2.96%	3.43%	4.05%	6.97%

5.2 Future Work

Following studies can be conducted in the future:

- Artifical noise can be added to the training data which is obtained numerical study. Thus, PINN model capability can be observed on noisy data.
- PINN approach can be applied on 3D laminar flow to understand how sustainable PINN convergence.
- In order to improve the PINN training, weighted loss algorithm which arranges weights of loss terms during the training can be implemented.
- Problem with a rotating domain rather than a stationary domain can be selected. For instance, rotating airfoil can be conducted for dynamic stall prediction by the PINN.
- Effect of grid spacing on the sensitivity of PINN can be studied.

BIBLIOGRAPHY

[1] J. Bertin and M. Smith, "Aerodynamics for engineers—second edition," *Aeronautical Journal*, vol. 259, pp. 19–99.

[2] J. Anderson, *EBOOK: Fundamentals of Aerodynamics (SI units)*. McGraw Hill, 2011.

[3] Y. Pinchover and J. Rubinstein, *Numerical methods*, p. 309–336. Cambridge University Press, 2005.

[4] C. Beck, M. Hutzenthaler, A. Jentzen, and B. Kuckuck, "An overview on deep learning-based approximation methods for partial differential equations," 2020.

[5] C. Yang, X. Yang, and X. Xiao, "Data-driven projection method in fluid simulation," *Computer Animation and Virtual Worlds*, vol. 27, no. 3-4, pp. 415–424, 2016.

[6] W. Edeling, P. Cinnella, and R. Dwight, "Predictive rans simulations via bayesian model-scenario averaging," *Journal of Computational Physics*, vol. 275, pp. 65–91, 2014.

[7] W. Edeling, P. Cinnella, R. Dwight, and H. Bijl, "Bayesian estimates of parameter variability in the k-e turbulence model," *Journal of Computational Physics*, vol. 258, no. February, pp. 73–94, 2014. Available online 23-10-2013.

[8] B. D. Tracey, K. Duraisamy, and J. J. Alonso, *A Machine Learning Strategy to Assist Turbulence Model Development*.

[9] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik, "Prediction of aerodynamic flow fields using convolutional neural networks," *Computational Mechanics*, vol. 64, pp. 525–545, jun 2019.

[10] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," 2016.

[11] A. Pal, "Deep learning emulation of subgrid-scale processes in turbulent shear flows," *Geophysical Research Letters*, vol. 47, no. 12, p. e2020GL087005, 2020.

[12] Y. Zhu and N. Zabaras, "Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification," *J. Comput. Phys.*, vol. 366, p. 415–447, aug 2018.

[13] Z. Zhang, X.-d. Song, S.-r. Ye, Y.-w. Wang, C.-g. Huang, Y.-r. An, and Y.-s. Chen, "Application of deep learning method to reynolds stress models of channel flow based on reduced-order modeling of dns data," *Journal of Hydrodynamics*, vol. 31, no. 1, p. 58–65, 2018.

[14] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," 2017.

[15] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations," 2017.

[16] M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden fluid mechanics: A navier-stokes informed deep learning framework for assimilating flow visualization data," 2018.

[17] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998.

[18] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "NSFnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations," *Journal of Computational Physics*, vol. 426, p. 109951, feb 2021.

[19] H. Xu and Y. Zhang, Weiand Wang, "Explore missing flow dynamics by physics-informed deep learning: The parameterized governing systems," *Physics of Fluids*, vol. 33, no. 9, p. 095116, 2021.

[20] S. Wang, Y. Teng, and P. Perdikaris, "Understanding and mitigating gradient pathologies in physics-informed neural networks," 2020.

[21] T. P. Miyanawala and R. K. Jaiman, "An efficient deep learning technique for the navier-stokes equations: Application to unsteady wake flow dynamics," 2017.

[22] Y. Zhang, W. J. Sung, and D. N. Mavris, *Application of Convolutional Neural Network to Predict Airfoil Lift Coefficient.*

[23] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, K. Willcox, and S. Lee, "Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence,"

[24] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, p. 422–440, 2021.

[25] J. D. Anderson Jr and J. D. Anderson, *A history of aerodynamics: and its impact on flying machines.* No. 8, Cambridge university press, 1998.

[26] F. White, *Fluid Mechanics.* McGraw-Hill series in mechanical engineering, McGraw Hill, 2011.

[27] P. M. Gerhart, A. L. Gerhart, and J. I. Hochstein, *Munson, Young and Okiishi's fundamentals of fluid mechanics.* John Wiley & Sons, 2016.

[28] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, "An introductory review of deep learning for prediction models with big data," *Frontiers in Artificial Intelligence*, vol. 3, 2020.

[29] A. R. Webb and K. D. Copsey, *Statistical pattern recognition.* Wiley, 2011.

[30] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *towards data science*, vol. 6, no. 12, pp. 310–316, 2017.

[31] C. Banerjee, T. Mukherjee, and E. Pasiliao, "An empirical study on generalizations of the relu activation function," in *Proceedings of the 2019 ACM Southeast Conference*, ACM SE '19, (New York, NY, USA), p. 164–167, Association for Computing Machinery, 2019.

[32] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436–444, 2015.

[33] F. Chollet, *Deep learning with python*. Manning, 2017.

[34] P. J. Bickel and K. A. Doksum, *Mathematical statistics: basic ideas and selected topics, volumes I-II package*. Chapman and Hall/CRC, 2015.

[35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[37] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533–536, 1986.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[39] "https://www.tensorflow.org/," 2015.

[40] "https://pytorch.org/," 2016.

[41] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[42] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[43] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, (Red Hook, NY, USA), p. 901–909, Curran Associates Inc., 2016.

[44] S. Mishra and R. Molinaro, "Estimates on the generalization error of physics informed neural networks (pinns) for approximating pdes," 2020.

[45] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017.

[46] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," 2015.

[47] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," 2017.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[49] D. Wilson and T. R. Martinez, "The general inefficiency of batch training for gradient descent learning," *Neural Networks*, vol. 16, no. 10, pp. 1429–1451, 2003.

[50] C. P. Jackson, "A finite-element study of the onset of vortex shedding in flow past variously shaped bodies," *Journal of Fluid Mechanics*, vol. 182, p. 23–45, 1987.

[51] S. R. Allmaras and F. T. Johnson, "Modifications and clarifications for the implementation of the spalart-allmaras turbulence model," in *Seventh international conference on computational fluid dynamics (ICCFD7)*, vol. 1902, ICCFD7-1902 Big Island, Hawaii, 2012.

[52] D. C. Wilcox *et al.*, *Turbulence modeling for CFD*, vol. 2. DCW industries La Canada, CA, 1998.