ASSESSMENT OF PHYSICS INFORMED NEURAL NETWORK FOR FLOW OVER BACKWARD FACING STEP

A PROJECT REPORT

Submitted by

MOHSIN KHAN [Reg No: RA1911019010074]

ECHAMPATI SUDHISHA [Reg No: RA1911019010004]

Under the guidance of

Dr. S. SENTHILKUMAR, PhD

(Associate Professor, Department of Aerospace Engineering)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

AEROSPACE ENGINEERING

of

FACULTY OF ENGINEERING & TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled **"ASSESSMENT OF PHYSICS INFORMED NEURAL NETWORK FOR FLOW OVER BACKWARD FACING STEP"** is the bonafide work of **"MOHSIN KHAN and ECHAMPATI SUDHISHA",** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr. S. Senthilkumar
GUIDE
Associate Professor
Department of Aerospace Engineering

SIGNATURE
Dr. L.R. Ganapathy Subramanian
HEAD OF THE DEPARTMENT
Professor & Head
Department of Aerospace Engineering

Signature of the Internal Examiner

Signature of the External Examiner

ABSTRACT

With the advent of modern computers, Computational Fluid Dynamics (CFD) calculations have become more affordable and less time-consuming. Nonetheless, as per Moore's Law, it will still take decades for modern computers to effectively perform time-consuming, complex CFD methods like Direct Numerical Simulation. Deep learning provides an affordable alternative to the problem. Physics Informed Neural Networks (PINN) have recently emerged as a strong candidate for solving partial differential equations among all the algorithms. This research explores the scope of PINN in approximating the solution of Partial Differential Equations by applying it to several problems: one-dimensional heat conduction,one dimensional convection-diffusion, one-dimensional unsteady heat conduction,one-dimensional nozzle flow, two-dimensional rectangular duct, and the Backward Facing Step Problem, which is a major benchmarking problem in CFD. Our research demonstrates that PINN predicts solutions with a high degree of accuracy.

ACKNOWLEDGEMENT

We express our humble gratitude to Dr. T.R. Paarivendhar, Chancellor, SRM Institute of Science and Technology.

We are extremely grateful to Prof. C. Muthamizhchelvan, Vice chancellor, SRM Institute of Science and Technology, for his constant support.

We wish to express gratitude towards Dr. L.R.Ganapathy Subramaniam, Head of Department of Aerospace Engineering, SRM Institute of Science and Technology, for his invaluable guidance and support throughout the project.

We are incredibly grateful to Dr. S. Senthilkumar, Associate Professor in the department of Aerospace Engineering at the SRM Institute of Science and Technology, for giving us the chance to work on our project with his priceless guidance.

We express our gratitude to Dr. Senthil Krishnababu, Associate Lecturer at University of Lincoln, without whom this project would not have been possible.

We thoroughly appreciate the assistance provided to us during our research by the faculty, staff, and students of the Aerospace Engineering Department of the SRM Institute of Science and Technology.

Finally, we would want to take this time to thank our parents for all their help with this endeavour.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

Abbreviation

AI	Artificial Intelligence
CFD	Computational Fluid Dynamics
CNN	Convolutional Neural Network
DNS	Direct Numerical Simulation
ML	Machine Learning
NN	Neural Networks
PDE	Partial Differential Equations
$PINN$	Physics Informed Neural Networks
$RANS$	Reynolds Averaged Navier-Stokes

Symbols and Variables

α	Thermal Diffusivity
μ	Dynamic Viscosity
ρ	Density
A	Cross Sectional Area
g	Acceleration due to gravity
h	Convective Heat Transfer
k	Thermal Conductivity
P	Perimeter
p	Pressure at a particular point
T	Temperature
u	Velocity in x direction
v	Velocity in y direction
x	Location in x axis
y	Location in y axis

Chapter 1

INTRODUCTION

Recent years have seen increasing use of Artificial Intelligence and Machine Learning. Neural Networks are finding applications in Aerospace Sciences. One such application is the use of Neural Networks to solve Computational Fluid Dynamics problems.

Computational Fluid Dynamics (CFD) is a field that deals with the numerical simulation of fluid flow and heat transfer phenomena. It has several applications in engineering, including the modelling and construction of aircraft, rockets, and satellites in the field of aerospace sciences. A method that is used in CFD to solve the Navier-Stokes equations, which describe the behavior of fluids, is Direct numerical simulation (DNS). DNS is the direct solution of the Navier-Stokes equations without any approximations or simplifications. As a result, it requires solving equations for each fluid component in the system, which is costly computationally and time-consuming. As a result, DNS is mainly restricted to academia, where small-scale systems are modelled.

DNS is unsustainable due to its time limits because commercial uses of CFD call for the simulation of considerably larger systems. As a result, scientists have started investigating alternate techniques that can deliver accurate results while being computationally effective.

Machine learning (ML) and artificial intelligence (AI) have shown promise as techniques for solving CFD problems. In recent years, CFD problems have been solved using neural networks, a machine learning algorithm that is inspired by the structure and operation of the human brain.

Large datasets of CFD simulations can be used to train neural networks to identify the underlying patterns and correlations. Once trained, they can be used to make predictions about how fluid flows would behave in novel situations without having to solve tricky

equations. As a result, they are more computationally effective and quicker than DNS and other conventional CFD approaches.

Technology and computer resources have advanced, enabling the creation of more potent computing systems that can process the sizable datasets needed for neural network training. Nevertheless, despite the advancements made in computing technology, DNS is not a practical solution for the majority of commercial uses of CFD. This is where the use of Neural Networks becomes crucial. Neural Networks provide a viable alternative for commercial applications of CFD by enabling accurate predictions while being computationally efficient.

1.1 Project Description

In this project, we aim to use Physics Informed Neural Networks (PINN) to solve the Navier-Stokes equations for the Backward Facing Step problem. The Backward Facing Step problem is a benchmarking problem that has been solved multiple times and is widely used in Fluid Dynamics research. Because of the problem's straightforward geometry, we can focus on the physics of it and study how well PINN can approximate the solution to the Navier-Stokes equations without being constrained by complicated geometry. Despite its simple geometry, the Backward Facing Step problem is a complex problem due to the flow separation region and the presence of recirculating flow patterns.

To begin with, the Navier-Stokes equations are divided into smaller components, then PINN is used to solve each component in turn. The goal of PINN is to approximate the solution of partial differential equations using neural networks by integrating the physics of the problem into the neural network architecture.

First, the Neural Network is trained to learn the relationship between the input variables, such as velocity and pressure, and the output variables, for a given set of boundary conditions. Then, the trained network is used to predict the values of the output variables for new boundary conditions, based on the physics of the problem.

1.2 Purpose

The purpose of this research is to contribute to the development of more efficient and accurate methods for solving fluid dynamics problems, as it has significant applications in fields such as aerospace engineering, weather, and oceanography. An alternative to the computationally expensive CFD will make analysis much faster and affordable.

1.3 Limitations

The scope of this project is limited to low Reynolds number flows. The decision to limit the scope of the project to low Reynolds number flows was made as the turbulence caused by high Reynolds number flows is highly complex and challenging to approximate. We plan to extend the project to higher Reynolds number flows in the future, once we have established the effectiveness of the model for low Reynolds number flows. By extending the project to higher Reynolds numbers, we can explore the performance of the PINNs for more complex and challenging fluid dynamics problems.

1.4 Fundamentals of Fluid Mechanics

1.4.1 Navier Stokes Equations

The Navier-Stokes equations are a set of coupled partial differential equations that govern fluid flow. They consist of momentum balance, conservation of mass and energy equation, which relates the pressure, density, and temperature of the flow. The x, y, and z spatial coordinates and the time t are the four independent variables in these equations. There are six dependent variables: the pressure p, density ρ, and temperature T, and three components of the velocity vector. The left-hand side of the momentum equation contains the convection term while the right-hand side contains the diffusion term.

$$\rho(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z}) = -\frac{\partial p}{\partial x} + \mu(\frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y} + \frac{\partial^2 u}{\partial^2 z}) \tag{1.1}$$

$$\rho(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z}) = -\frac{\partial p}{\partial y} + \mu(\frac{\partial^2 v}{\partial^2 x} + \frac{\partial^2 v}{\partial^2 y} + \frac{\partial^2 v}{\partial^2 z}) \tag{1.2}$$

$$\rho(\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + v\frac{\partial w}{\partial y} + w\frac{\partial w}{\partial z}) = -\frac{\partial p}{\partial z} + \mu(\frac{\partial^2 w}{\partial^2 x} + \frac{\partial^2 w}{\partial^2 y} + \frac{\partial^2 w}{\partial^2 z}) \tag{1.3}$$

$$\frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} + \frac{\partial u}{\partial z} = 0 \tag{1.4}$$

ρ denotes the density, u denotes velocity in the x-direction, while v denotes velocity in the y-direction, w denotes velocity in the z-direction, p denotes the pressure, and μ denotes the dynamic viscosity.

Equation 1.1 to 1.3 are momentum equations whereas equation 1.4 is the continuity equation

One of the crucial applications of the Navier-Stokes equations is in the study of fluid dynamics. The equations are used to model the behavior of fluids in aircraft engines and wind turbines and pipelines. They are also used in weather forecasting and climate modeling to study the dynamics of the atmosphere and oceans.

Because of the Navier-Stokes' nonlinearity, it is impossible to predict the fluid's behaviors by adding up each force's effects. The equations, on the other hand, explain how the many forces and interactions inside the fluid interact with one another to form complicated patterns of flow, like turbulence or vortices.

The Navier-Stokes equations are continuous, that is, they capture the behaviour of the fluid at every location in space and time. This makes it possible to compute the fluid's velocity, pressure, and other parameters at any given point in time and space.

The equations are derived from applying Newton's second law. The equations are extensions of the Euler Equations. While the Navier-Stokes equations are used to describe viscous flow, the Euler equations are used to describe inviscid flows. The Euler can not model boundary layers or turbulence.

Analytical solutions to the Navier-Stokes equations are challenging. The solutions to the equations have been approximated using Finite Difference, Finite Element, and Finite

Volume methods. The advent of high-speed computers has made this possible.

1.5 Introduction to Machine Learning

The field of machine learning enables computers to learn and develop without being explicitly programmed. It is a subset of artificial intelligence (AI) that is concerned with building systems that can learn from data, identify patterns, and make decisions based on that data. Machine learning algorithms are used in a variety of applications, from voice recognition and image classification to fraud detection and recommendation systems.

At the heart of machine learning is the concept of training data. Large data sets are used to train machine learning algorithms to find patterns and relationships in the data. Once trained, these algorithms can be used to make predictions or classifications on new data that they have not seen before. The more data a machine learning algorithm has access to, the more accurate its predictions are likely to be.

Machine Learning algorithms are vastly divided into following :

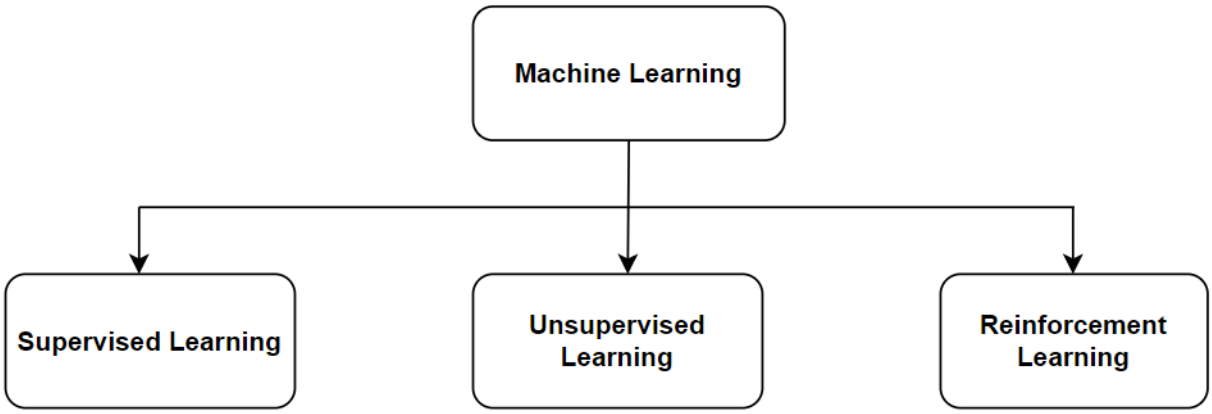


Figure 1.1: Classification of Machine Learning algorithm

1.5.1 Supervised Learning

Supervised Learning algorithm involves training the algorithm on labelled dataset i.e. dataset which already contains the required parameter.

Supervised learning is further divided into two parts which is given in figure

Regression algorithm as the name suggests, predicts numbers like predicting house pric-

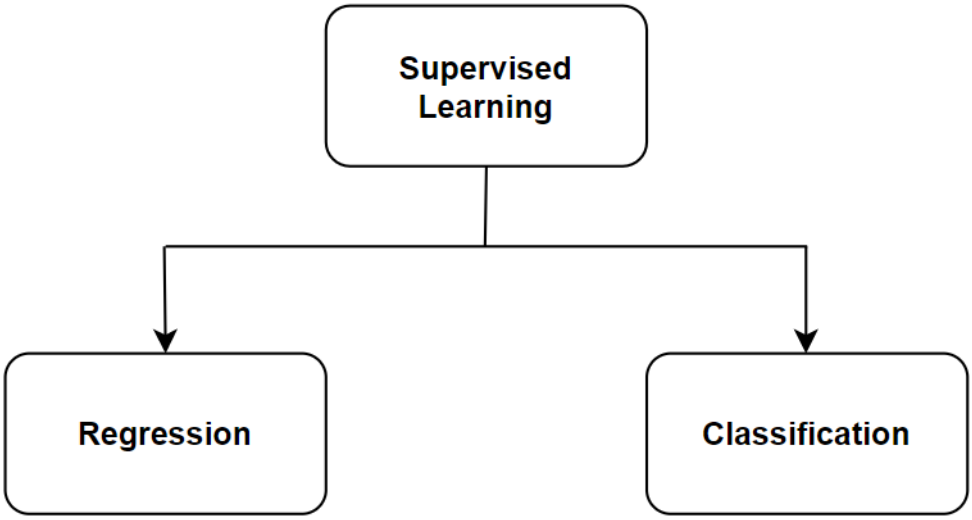


Figure 1.2: Classification of Supervised Learning algorithm

ing, or predicting Coefficient of Lift and Coefficient of Drag of an airfoil. Some examples of Regression algorithms are given below:

- Linear Regression
- Non-Linear Regression
- Regression Trees
- Polynomial Regression
- Bayesian Linear Regression

On the other hand classification algorithm are used for classification problems like predicting whether the tumor in brain is benign or malignant. Examples of Classification algorithm are given below.

- Random Forest
- Decision Trees
- Logistic Regression
- Support Vector Machines

1.5.2 Unsupervised Learning

Unsupervised Learning involves training a data on unlabeled data set, i.e the algorithm should figure out pattern and relation in the data on its own.

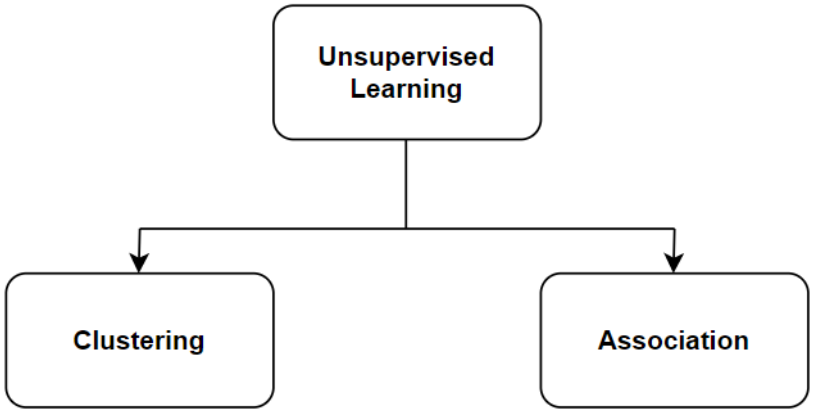


Figure 1.3: Classification of Unsupervised Learning algorithm

In clustering the algorithm groups similar data together into a cluster and the relationship between variables is found.

Types of unsupervised learning algorithms are:

- K-means clustering
- hierarchical clustering
- Anomaly detection
- KNN (k-nearest neighbors)

1.5.3 Reinforcement Learning

In reinforced learning, the algorithm learns through trail and error, where its rewarded when it makes correct decision and penalised when it makes wrong decision.

1.6 Neural Networks

Neural Networks is a sub branch of machine learning, which are modelled after working of the human brain. They are composed of a number of nodes which are interconnected and work together to generate the required output. Neural Networks now have applica-

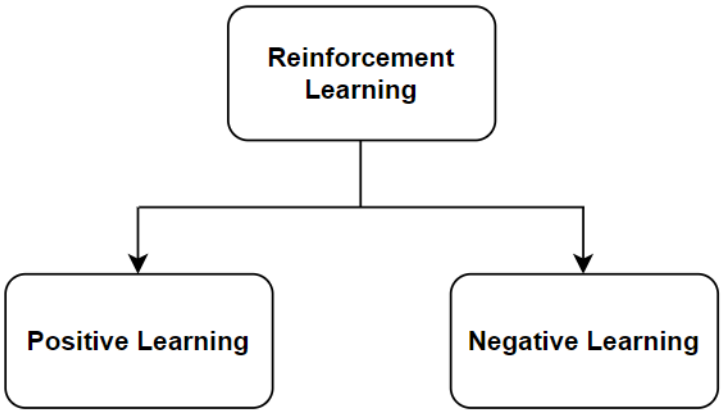


Figure 1.4: Types of Reinforcement Learning algorithm

tion in almost every field. They are most prominently used in image recognition, self driving cars, text generation, language processing etc.

At the core of neural networks is something called perceptron or neuron, which is a mathematical function that takes in the input and produces output based on the weights assigned to each input. The weights are adjusted during the training process and then the output is sent to the next layer. The process is repeated till we reach the required output.

There are various types of neural networks, such as Feedforward Neural Network, Convolutional Neural Networks, Recurrent Neural Networks, Physics Informed Neural Networks. Feedforward Neural Network are the simplest kind where the data flows in one direction. Convolutional neural networks used filters and are commonly used in image related tasks. Recurrent Neural Networks work best with time series data, such as audio. Physics informed networks are used where there are differential equation.

1.6.1 Back Propagation

Backward Propagation or back propagation is an algorithm used to train Neural Networks. The algorithm involves computing the gradient of a loss function with respect to the weights and biases of the neural network, and using this gradient to update the weights and biases to minimize the loss function.

Backpropagation has following steps:

1. Forward pass: The inputs to the neural network are fed through the network, layer by layer, to compute the outputs

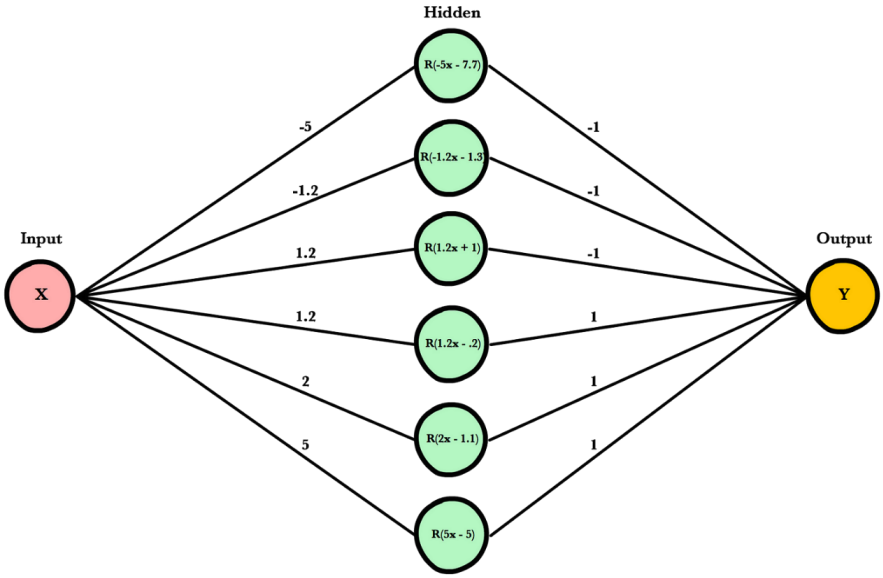


Figure 1.5: Neural Network architecture with weights

2. Loss calculation: The predicted outputs are compared to the actual outputs (targets) to calculate a loss function that measures the accuracy of the neural network.

3. Backward pass: This step is based on chain rule. The neural network computes the gradient of the loss function by breaking it down into smaller parts, in the backward direction starting from the output layer.

4. Updates Weights: The neural network updates the weights and biases.

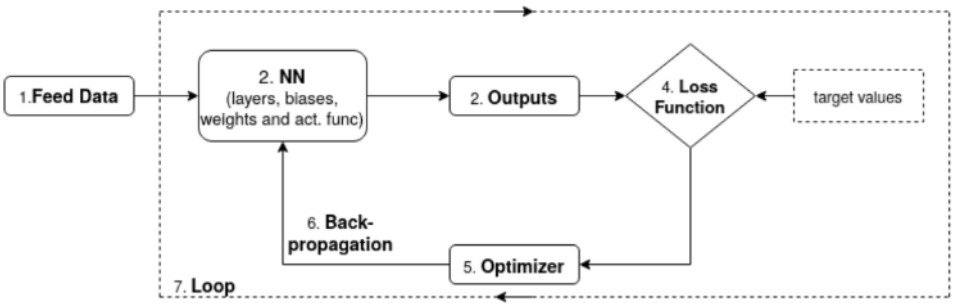


Figure 1.6: Backpropagation scheme

1.6.2 Gradient Descent

Gradient Descent is another important algorithm used in building a neural network. It is used to train models by minimizing the loss function that measures the difference between the predicted output and the true output.

The principle behind gradient descent is to iteratively change the model's parameters in the direction of the loss function's steepest decline. This is accomplished by computing the loss function's gradient with respect to the parameters and moving in the gradient's opposite direction.

The size of each step is controlled by the learning rate, a parameter. If the learning rate is too large, the algorithm may diverge, and if it is too small, the algorithm may converge too slowly or get stuck in a local minimum.

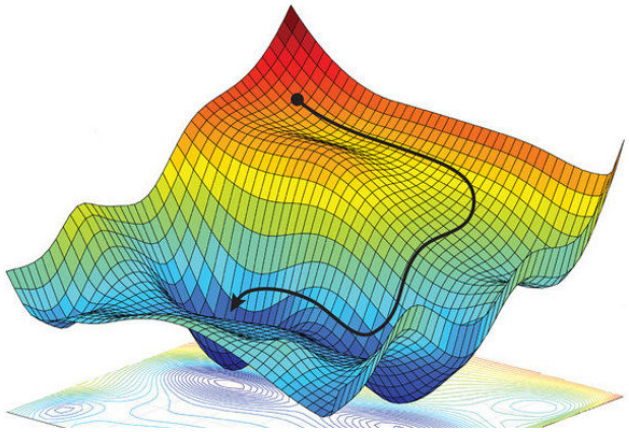


Figure 1.7: Path taken to reduce loss function by Gradient Descent algorithm

1.6.3 Activation Functions

An essential part of neural networks, activation functions allow the model to learn complicated relationships between inputs and outputs by introducing non-linearity into the system. An activation function uses a non-linear transformation to the weighted sum of inputs with bias to produce an output.

Some of the commonly used activation functions are:

1. Linear: A linear activation function gives a linear output based on the weighted sum of inputs and bias. It is defined as f(x) = x, where x is the input value.
2. Sigmoid: Any real-valued number can be converted by the sigmoid function to a number between 0 and 1. It is frequently used in binary classification problems' output layer.
3. Rectfied Linear Unit or ReLU:The ReLU function is defined as f(x) = max(0,x). Due to its ease of use and efficiency, it has emerged to be one of the most used

activation functions. It accelerates the convergence of gradient descent and aids in overcoming the vanishing gradient problem.

4. Tanh (Hyperbolic tangent): The tanh function maps any real-valued number to a value between -1 and 1. It is generally used in the hidden layers of neural networks.

5. Softmax: The output layer of problems involving multi-class classification uses the softmax function. It converts the results into a probability distribution whose values add up to 1.

6. Leaky ReLU: The leaky ReLU function is similar to ReLU except that it assigns a small negative value to negative inputs rather than setting all negative values to 0. This decreases the likelihood of the ”dying ReLU” problem, in which some neurons may stop functioning and cease to contribute to the output of the network.

7. ELU (Exponential Linear Unit): The ELU function is similar to the leaky ReLU function, but it smooths out the function and performs better during training since it utilises an exponential function for negative inputs.

Chapter 2

LITERATURE SURVEY

Despite being a relatively new field, Physics Informed Neural Networks has gathered quite a bit of momentum in the past half a decade. Numerous researchers have developed code to solve the Navier-Stokes equations and evaluated its performance in comparison to experimental and CFD solutions.

In their research, Raissi et al. [11] introduce Physics Informed Neural Networks as Neural Networks designed to solve supervised learning tasks in a way that adheres to a specific set of physical laws, as defined by general nonlinear partial differential equations. The paper demonstrates the effectiveness of this approach through several examples. In the first part of their research, the authors approximate the solutions for Burgers' Equation, Schrödinger Equation, Allen-Cahn Equation using PINNs. In the second part of their research [12], they explore the Data-driven discovery of Nonlinear Partial Differential Equations. They developed a neural network to approximate the Navier-Stokes equations for the problem of incompressible flow past a circular cylinder. The authors conclude that Physics-Informed Deep Learning models can outperform traditional numerical methods in solving PDEs.

The authors Raissi et al. [13], use deep learning and physical constraints to present a new method for reconstructing flow fields from sparse and noisy experimental data. The governing equations of fluid mechanics are incorporated into the neural network architecture by the authors, who also present a Navier-Stokes informed deep learning framework. The method is validated using experimental data for flow visualisation, and the findings demonstrate that it can precisely rebuild complex flow fields and capture key physical properties like vortices and flow separation. As examples, the paper considers two-dimensional flow past a circular cylinder, three-dimensional flow past a circular cylinder, transient flow over an obstacle in a two-dimensional channel, and 3D

physiologic blood flow in a realistic intracranial aneurysm. The authors conclude that the algorithm created in this research is independent of geometry, initial, and boundary conditions, making it extremely adaptable in terms of selecting the area of interest for data gathering as well as future training and prediction.

Mao et al. [9] use Physics Informed Neural Networks in their research to approximate the Euler equations that characterise high-speed aerodynamic flows. In both one- and two-dimensional domains, the forward and inverse problems are solved. The results of their research conclude that PINNs are proven to be less accurate when compared to traditional methods for forward problems. However, PINNs are a better choice for inverse problems which are difficult to solve with traditional methods.

In their paper, Sekar et al. [14] predict the flow over an airfoil using Machine Learning. A data-driven approach is used based on combining deep Convolutional Neural Networks (CNN) and Multilayer Perceptron (MLP). The flow over the airfoil is chosen to be incompressible, laminar and steady. Their approach consists of two steps. The CNN first extracts the geometrical parameters of the airfoil. This information and the Reynolds number and angle of attack is then passed on to the MLP, which predicts the flow. The authors use OpenFOAM solver to train the MLP. The authors achieve an accuracy of over 99.0

The authors Cai et al. [3], give an overview of the use of PINNs in solving fluid mechanics problems. The authors reiterate that PINNs are not so accurate when compared to higher-order CFD methods in approximating forward problems. The paper compares PINN and CFD solutions for 3D Incompressible Wake Flow past a Stationary Cylinder, 2D Steady Compressible Bow Shock Wave, and 2D Flow past a Thrombus (Biomedical Flows). The research paper comes to the conclusion that the PINN solutions and CFD data concur well.

In the paper, Cai et al. [4] propose a new approach for solving heat transfer problems utilising physics-informed neural networks. The authors suggest that traditional numerical methods for solving heat transfer problems can be enhanced by incorporating physical knowledge and constraints into the neural network architecture. This is accomplished by combining the principles of physics with data-driven approaches, resulting in models that can learn from data and apply physical principles simultaneously. The paper demon-

strates the usefulness of the proposed approach through several examples, including the one-dimensional heat equation and the two-dimensional convection-diffusion equation. The results suggest that physics-informed neural networks can outperform traditional numerical methods in terms of accuracy and computational efficiency. The authors also discuss the limitations and challenges of the proposed approach, the need for high amounts of training data and the sensitivity of the suggested approach to the selection of parameters. Overall, the paper suggests that physics-informed neural networks have the potential to enhance our ability to solve complex heat transfer problems and improve the efficiency and accuracy of numerical simulations.

Eivazi et al. [5], in their paper, explore PINNs while solving the Reynolds Averaged Navier Stokes (RANS) equations. The authors propose that by integrating physical knowledge and limitations into the neural network architecture, PINNs can be used to enhance the precision and effectiveness of RANS simulations. The methodology for implementing PINNs for RANS simulations is presented in the research, and it includes the use of a modified loss function to impose the RANS equations and the inclusion of a penalty term to make sure that the output of the neural network complies with the physical constraints. The method is put to the test on a number of cases, including two-dimensional Navier–Stokes equations for the Falkner–Skan boundary layer, ZPG turbulent boundary layer, APG turbulent boundary layer, the boundary layer developing on a NACA4412 airfoil, and the turbulent flow over a periodic hill. The outcomes show that PINNs can offer precise and effective solutions for RANS problems.

The authors Jina et al. [8]introduce a new neural network architecture called NSFnets by taking into account two distinct mathematical interpretations of the Navier-Stokes equations:the velocity-pressure (VP) formulation and the vorticity-velocity (VV) formulation. The authors test the NSFnets on benchmark problems to evaluate the precision, rate of convergence, computational cost, and flexibility. The authors use Analytical solutions and databases for direct numerical simulation (DNS) to give suitable initial and boundary conditions for the NSFnet simulations. The spatial and temporal coordinates are the inputs of the NSFnets. The authors use unsupervised learning, hence eliminating the need for inputting data to guide the neural network. The authors obtain moderate accuracy for VP-NSFnet and note that PINNs efficiency can be improved by developing a multi-node GPU code, which will make the training much quicker. They also note that

VV-NSFnet achieves better accuracy than the VP-NSFnet for laminar flows.

In their research, the authors Xiang et al. [16] propose a self-adaptive loss function technique, which updates the noise parameters with each iteration based on the maximum likelihood estimation and automatically assigns the weights of losses. They test the model on two-dimensional steady Kovasznay flow, two-dimensional unsteady cylinder wake, and three-dimensional unsteady Beltrami flow. The authors conclude that in their proposed method the loss function reduces quicker, and error is smaller as compared to original PINNs.

Yang et al. [17] introduce a predictive Large Eddy Simulation wall modeling using PINNs. The authors use a feed forward neural network architecture. The neural networks are then trained on data from fully developed channel flow at Re = 1000. The trained neural networks learn the law of wall although the network was not directly given information about it. The neural network performed better than the conventional equilibrium wall model. The authors also mention the challenges faced: lack of large amounts of DNS data on public domains, the lack of clarity about the amount of information that needs to be given an input, improving accuracy while keeping the computation costs low, and they also note that other ML techniques might perform better than the Wall Model LES.

In their research, Oldenburg et al. [10] propose Geometry Aware Physics Informed Neural Networks (GAPINN), which do not require any data and solve irregular geometries without parameterization. The GAPINN works only by reducing the residuals of the governing equations. The model had three network types. The dimensions of the asymmetric geometries are condensed to a latent representation by the first network, the Shape Encoding Network or SEN. The SEN and spatial coordinates act as input for the PINN, which trains a surrogate model. The third network is a Boundary Constraining Network, which is used to impose boundary conditions while the PINN is being trained. The GAPINN is tested in biofluidmechanics at Re = 500. The authors note that although the model was developed to solve the Navier-Stokes equations, adapting it to solve other PDEs should not be a challenge.

Arthurs et al. [1] put forward a novel approach to train a PINN to model Navier-Stokes pressure and velocity fields across a parametric domain. The parameters correspond

to the geometry of the domain and a fluid boundary condition. The authors use Finite Element Method solution data for pressure and velocity fields. However, their model is trained on solely velocity data. The authors show that the model works with small amounts of training data, chosen and generated during training by an Active Learning Algorithm. In order to accomplish that, the neural network training process was coupled with a mesh generator and a numerical solver for the Navier-Stokes equations. The authors illustrated that when looking for parameters that give specific physical properties, the network can execute exceptionally quick and accurate parameter sweeps.

The authors Arzani [2] use PINNs in blood flow dynamics. They test their model on one-dimensional advection-diffusion transport, two-dimensional blood flow in an aneurysm, two-dimensional blood flow in a stenosis, three-dimensional blood flow in an aneurysm, and Parameter identification (viscosity). The inlet and outlet boundary conditions are assumed to be unknown. The PINN results is in excellent agreement with the analytical solution for the 1D problem. The model performed very well for 2D cases and had only a small error in the 3D case. The authors note that there is a lack of generalizable information in 3D hemodynamic databases. The authors also mention that Convolutional Neural Networks can be used to improve PINN's convergence and accuracy. Finally, more research needs to be done to see how well the model works with real-world data.

Gladstone et al.[7] present FO-PINNs. These PINNs are trained using first order formulation of PDE losses. In order to guarantee compatibility between the predicted and exact derivatives, the second and higher-order PDEs are reconstructed as a collection of first order PDEs with additional compatibility equations. The loss function includes these compatibility equations as extra terms. Hence, automatic differentiation is used solely to calculator first-order derivatives. To impose the boundary conditions, the authors use the theory of R-functions to calculate the Approximate Distance Function to the boundaries and create a boundary-condition and geometry-aware solution. This boundary condition imposition method is made possible by FO-PINNs, which eliminate the need for second and higher-order derivative computations for more accurate solutions. The neural network is tested on Helmholtz equation, and flow through a parameterized cylindrical channel. The authors conclude that FO-PINNs can be trained 2.9 times faster compared to PINNs. The training is sped up due to the removal of additional backpropagation steps for calculation of higher-order derivatives, and by using Automatic Mixed

Precision for training. The authors also note that FO-PINNs provide significantly more accurate results for parameterized systems compared to PINNs. The FO-PINN can be generalized to problems involving higher order PDEs.

Chapter 3

OBJECTIVE AND METHODOLOGY

3.1 Objective

The objective of this project is to solve the Navier-Stokes using a PINN model. To assess their performance, the PINN models are tested on various problem statements, which are, one dimensional heat conduction, one dimensional convection-diffusion, one-dimensional unsteady heat conduction, one-dimensional nozzle flow, two-dimensional rectangular duct, and the Backward Facing Step Problem. The given order of problems was adopted because of their increasing complexity, with Backward Facing Step being the most complex out of them all.

3.2 Methodology

The Navier-Stokes equations are a set of partial differential equations that describe the motion of fluids. However, solving the Navier-Stokes equations for complex problems is often computationally expensive and requires significant computational resources. To make this process more manageable, the equations are broken down into smaller units that can be solved step-by-step.

Neural networks are a subset of machine learning models that are designed to mimic the behaviour of the human brain. They consist of interconnected nodes or neurons that are organized into layers. By adjusting the weights and biases of these neurons, a neural network can learn to recognize patterns in data and make predictions. In this project, a neural network was used to read handwritten digits. The code for this project was written using TensorFlow.

3.2.1 Tools Used

Google Colaboratory is a cloud-based platform offered by Google that provides an easy-to-use interface for developing neural networks and running simulations. It allows users to access powerful computing resources without the need for expensive hardware. TensorFlow with a Python interface was utilised to accomplish the data-driven methodology in this project. This project made use of TensorFlow, a Google open-source software library that had its initial version released in February 2017 and its 2.0 version released in January 2019. We also employed Keras, a Python-based deep learning API that is open source. TensorFlow can be used with Keras, which offers a straightforward and dependable user interface.

Figure 3.1: Integration of Keras and Tensorflow with Python

Ansys FLUENT is a widely used commercial software package for solving fluid mechanics problems. It provides a comprehensive suite of tools for pre-processing, solving, and post-processing fluid dynamics simulations. In this project, FLUENT was used to generate the data used for training and validating the neural network models. By using FLUENT to generate data, we were able to test the accuracy and reliability of their PINN models.

3.3 Physics Informed Neural Networks

Physics-informed neural networks (PINNs) are a type of neural network that incorporates physical laws into its architecture. In order to solve partial differential equations (PDEs), PINNs combine the strength of neural networks with a thorough understanding of physics.

Numerical techniques like finite element analysis and finite difference approaches are used in conventional PDE solutions. These techniques, however, can be computationally expensive and need a lot of computing power. By using neural networks to directly learn the solution to the PDEs, PINNs provide a more effective solution.

The physics-informed part of the network is achieved by incorporating the governing equations of the PDE into the loss function of the network. As a result, the network can learn both the PDE's solution and the fundamental physics that underlie it. As a result, PINNs can offer more precise answers to challenging issues.

Numerous real-world fields, such as fluid dynamics, materials science, and geophysics, can benefit from the use of PINNs. They can be used, for instance, to simulate fluid flow through porous medium, which is crucial in the oil and gas sector. The behaviour of materials under various situations, such as temperature and pressure, can also be modelled using them.

The capacity of PINNs to learn from sparse data is one of their benefits. For traditional PDE methods to work, a lot of data must be available. On the other hand, PINNs can learn from just one piece of data. They become especially helpful in circumstances where data is limited or challenging to collect because of this.

Another advantage of PINN is that they are good in handling complex geometries, which can be challenging for traditional numerical methods. This makes them particularly useful in applications such as aerospace engineering and medical imaging, where the geometry of the problem can be complex.

The architecture followed by physics informed neural networks is given in figure below.

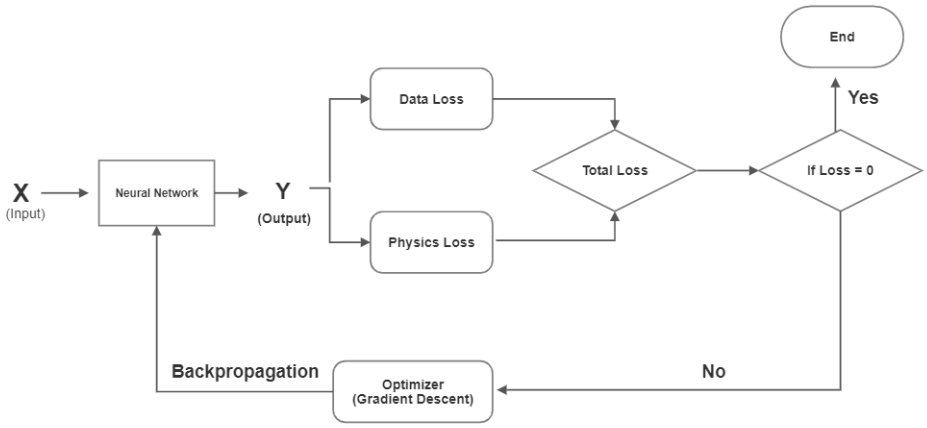


Figure 3.2: PINN architecture Flow chart

3.3.1 Autodifferentiation

In machine learning and scientific computing, autodifferentiation is a method for calculating function derivatives. The fundamental goal of autodifferentiation is to automatically calculate a function's derivative by dividing it into a series of simple operations, each of which has a known derivative.

There are two main types of autodifferentiation: forward-mode and reverse-mode. In forward-mode autodifferentiation, the function is evaluated at a particular point in the domain, and the derivatives are computed by applying the chain rule to each operation in the function, starting from the input variables and proceeding forward through the computation graph.

Chapter 4

PREDICTION OF THERMO-FLUID PROPERTIES USING PINN

4.1 1D Heat Conduction Equation

4.1.1 Problem Formulation

The problem was taken from textbook [15]. The domain in this problem (Figure 4.1) is a cylindrical fin. It has a uniform cross-sectional area A. The temperature at the base is 100°C and the fin has an insulated end.There is an ambient temperature of 20°C around the fin. Using PINN we will predict out temperature distribution across the fin.

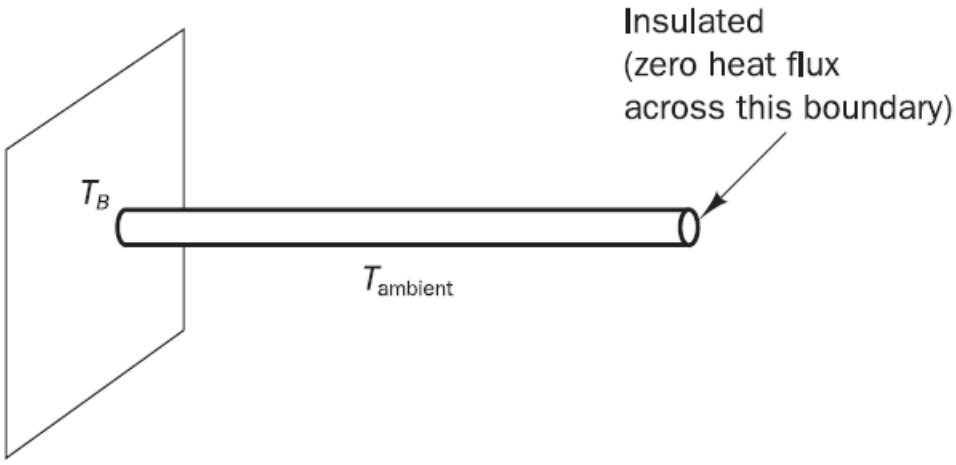


Figure 4.1: 1D Heat Conduction Problem formulation [15]

4.1.2 Governing Equations

The above problem statement is governed by the following governing equation:

$$kA\frac{\mathrm{d}^2T}{\mathrm{d}x^2} - hP(T - T_\infty) = 0 \tag{4.1}$$

where k is thermal conductivity, A is cross sectional area, h is convective heat transfer coefficient, P is perimeter, T_∞ is the ambient temperature.

4.1.3 Boundary Condition and Constant Values

Parameter	Value
Base Temperature(TB)	100 C
Free End Temperature(TA)	20 C
L	1 m
$hP/(kA)$	$25m^{-2}$

Table 4.1: BC and Constant Values of Problem 4.1

4.1.4 Neural Network Parameters

Parameter	Value
Number of Hidden Layers	3
Number of Neuron in Hidden Layer	20
Number of Input in Input Layer	1
Number of Output in Output Layer	1
Activation Function	tanh

Table 4.2: 1D Heat Conduction NN Parameters

4.1.5 PINN Loss

The total loss comprise of two loss functions Physics loss and Boundary layer loss The physics loss is given as :

$$kA\frac{\mathrm{d}^2T}{\mathrm{d}x^2} - hP(T - T_\infty) = \text{Physics Residue} \tag{4.2}$$

$$\text{Physics Loss} = \frac{1}{n}(\sum_{i=1}^{n}\sqrt{(\text{Physics Residue})^2}) \tag{4.3}$$

where n is the total number of collocation points

The boundary condition loss is given by:

$$\text{Boundary Condition Loss} = \frac{1}{n}(\sum_{i=1}^{n} \sqrt{(Y_{\text{pred}} - Y_{\text{bc}})^2}) \tag{4.4}$$

where Y_{pred} is the predicted property value in this case temperature and, Y_{bc} is the boundary condition that has to be applied

$$\text{Total Loss} = \text{Physics Loss} + \text{Boundary Condition Loss} \tag{4.5}$$

4.1.6 Analytical Solution

The analytical solution of equation is given by:

$$\frac{T - T_\infty}{T_B - T_\infty} = \frac{cosh[n(L-x)]}{cosh(nL)} \tag{4.6}$$

where T_∞ is the ambient temperature, T_B is the base temperature, L is the length of the rod

4.1.7 Results and Validation

The training took a large amount of epochs to train because the right boundary was not enforced using boundary condition.

The following table shows comparison between PINN solution and analytical solution.

X value	PINN Temperature	Analytical Temperature	% Error
0.2	49.244072	49.43889	0.39
0.4	30.712288	30.8531	0.675
0.6	23.958843	24.05573	0.403
0.8	21.461708	21.6634	0.931
1.0	20.5857	21.07802	2.33

Table 4.3: Error in PINN solution

From the table it is clear that error is less in the location where boundary condition were enforced.

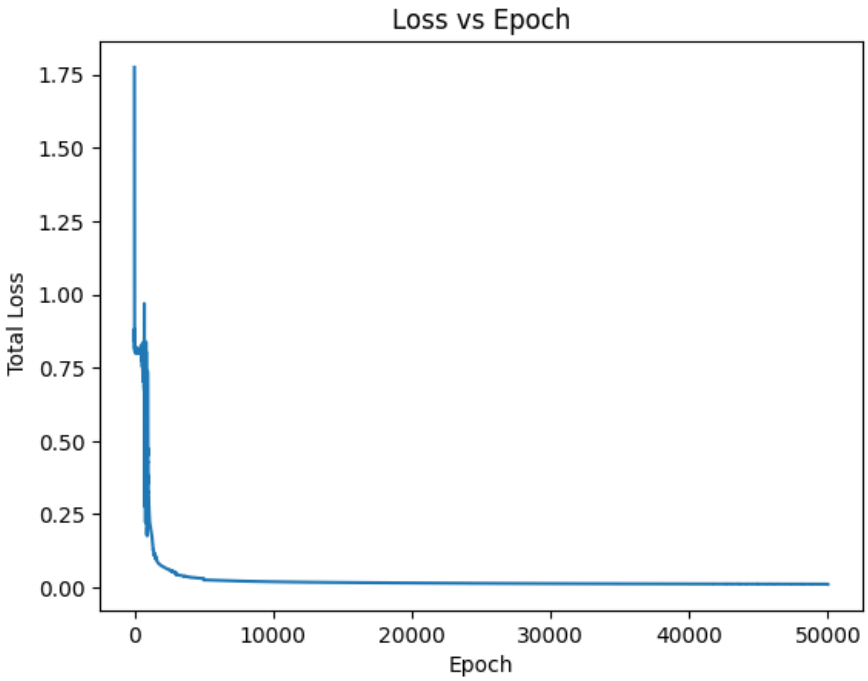


Figure 4.2: 1D steady heat conduction loss function vs epoch

4.2 1D Unsteady Heat Conduction

4.2.1 Problem Statement

In the problem, a rod is initially at temperature given by $200sin(x)$. Temperature is calculated using PINN on how temperature will change with passage in time.

4.2.2 Governing Equations

The above problem statement is governed by the following governing equation:

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = 0 \tag{4.7}$$

where α is thermal diffusivity, x is location, T is temperature and t is time.

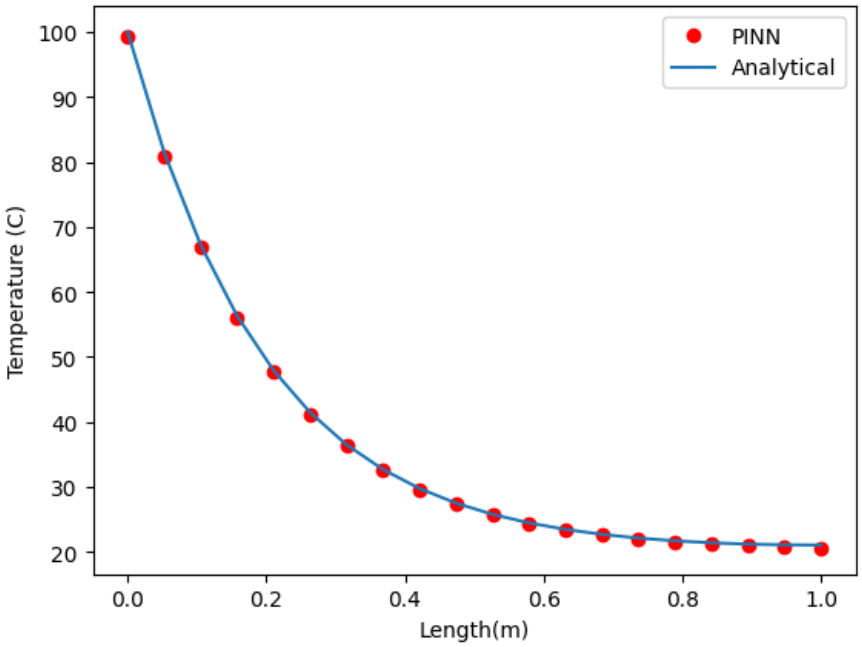


Figure 4.3: Temperature plot of 1D heat conduction equation

4.2.3 PINN Domain

The following domain is created to solve the problem.

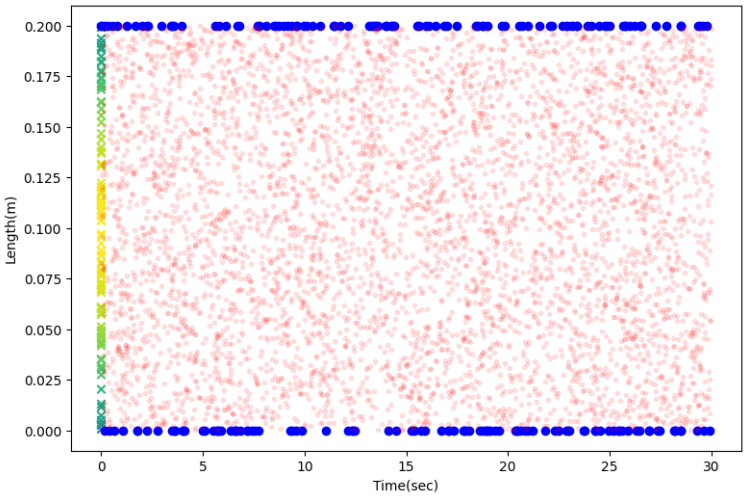


Figure 4.4: PINN Domain for Problem 4.2

4.2.4 Boundary Condition and Constant Values

Parameter	Value
Initial Temperature Distribution	sin(x)
Temperature at rod ends	0 C
alpha	1.77e-5

Table 4.4: Boundary and Initial Condition for Problem 4.2

4.2.5 Neural Network Parameters

Parameter	Value
Number of Hidden Layers	4
Number of Neuron in Hidden Layer	20
Number of Input in Input Layer	2
Number of Output in Output Layer	1
Activation Function	tanh

Table 4.5: 1D Unsteady Heat Conduction NN Parameters

4.2.6 PINN Loss

The PINN loss is same as 1D steady heat conduction, the only difference is in the Physics loss which is given below.

$$\rho c \frac{\partial T}{\partial t} - K \frac{\partial^2 T}{\partial x^2} = \text{Physics Loss} \tag{4.8}$$

The boundary condition loss is same as in equation (eqn no)

4.2.7 Results

From the figure 4.5 given below it we can see that the solution converged at around 4000 epoch.

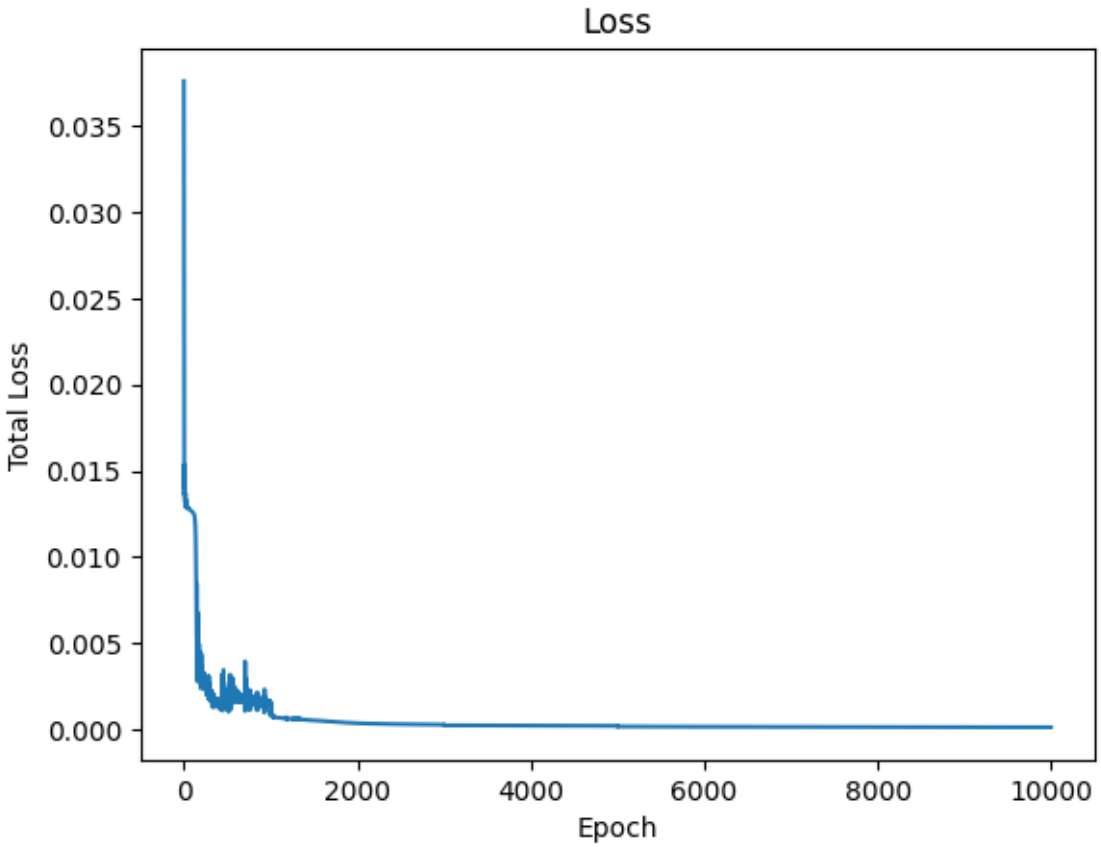


Figure 4.5: 1D unsteady heat conduction loss function vs epoch

The PINN solution is deviated from CFD solution at around x = 0.1 m. The table below shows the deviation from the CFD results.

X value	PINN Temperature	CFD Temperature	% Deviation
0.05	121.970535	124.6299	2.188
0.10	178.271	175.1774	1.765
0.15	121.9728	123.3612	1.112

Table 4.6: PINN and CFD result comparison on 1D unsteady heat conduction

The training time was 70.4 sec to run 10000 epoch on GPU accelerated Google Colab Notebook.

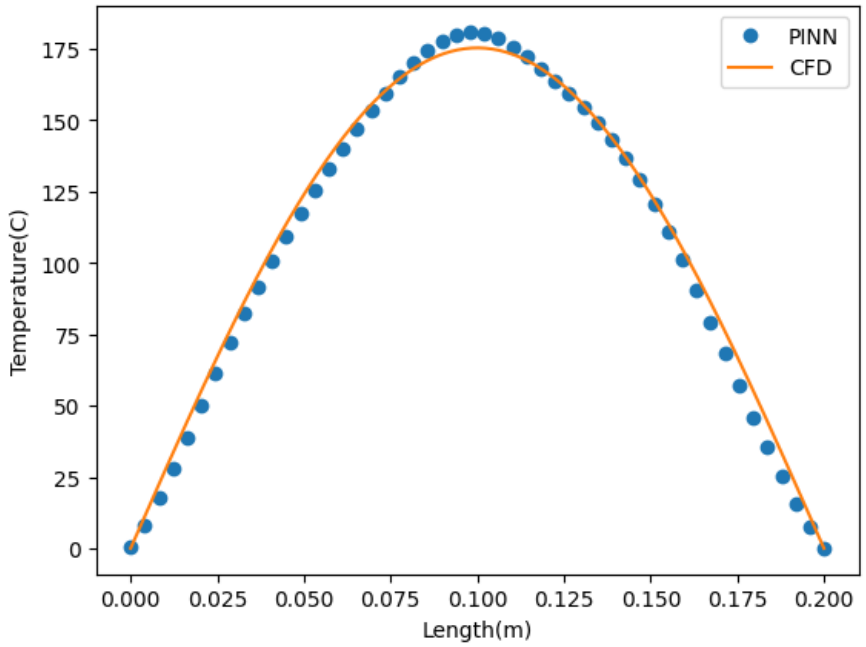


Figure 4.6: Temperature vs Length graph at t=30 sec

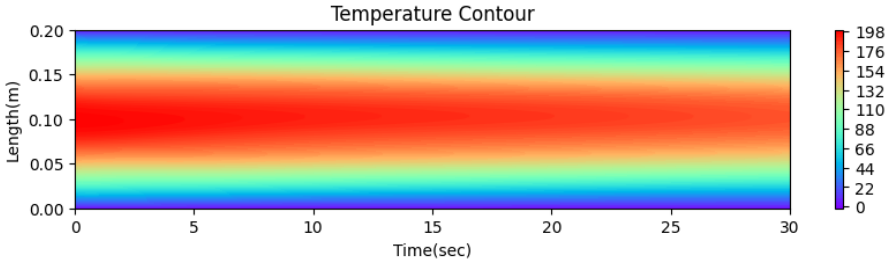


Figure 4.7: Temperature contour of 1D unsteady heat conduction

4.3 1D Convection Diffusion Equation

4.3.1 Problem Statement

This problem statement is taken from textbook [15]. A property ϕ is carried through convection and diffusion in the one-dimensional domain sketched in Figure 4.8.

4.3.2 Governing Equation

$$\frac{\mathrm{d}}{\mathrm{d}x}(\rho u \phi) = \Gamma \frac{\mathrm{d}^2 \phi}{\mathrm{d}x^2} \tag{4.9}$$

$$\frac{\mathrm{d}\rho u}{\mathrm{d}x} = 0 \tag{4.10}$$

For incompressible flow we can ignore equation 4.10

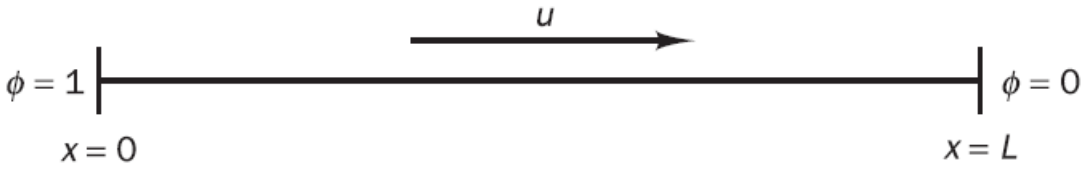


Figure 4.8: 1D Convection Diffusion Problem Statement [15]

4.3.3 Boundary Condition and Constant Values

Parameter	Value
ϕ_0	1
ϕ_L	0
Length(L)	1 m
ρ	$1kg/m^3$
Γ	0.1 kg/ms

Table 4.7: Boundary and Initial Condition for Problem 4.2

4.3.4 Neural Network Parameters

Parameter	Value
Number of Hidden Layers	4
Number of Neuron in Hidden Layer	20
Number of Input in Input Layer	1
Number of Output in Output Layer	1
Number of Calculation points	100
Activation Function	tanh

Table 4.8: 1D Unsteady Heat Conduction NN Parameters

4.3.5 PINN Loss

The Boundary condition part of the PINN loss is same as the previous statements only values need to be changed. The physics part is given below:

$$\frac{\mathrm{d}}{\mathrm{d}x}(\rho u\phi) - \Gamma\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = \text{Physics Residue} \tag{4.11}$$

4.3.6 Analytical Solution

The analytical Solution of 1D convection diffusion is given by the following equation:

$$\frac{\phi - \phi_o}{\phi_L - \phi_o} = \frac{exp(\rho u x/\Gamma) - 1}{exp(\rho u L/\Gamma) - 1} \tag{4.12}$$

where ϕ_o is ϕ at x = 0 and ϕ_L is ϕ at x = L

4.3.7 Results and Validation

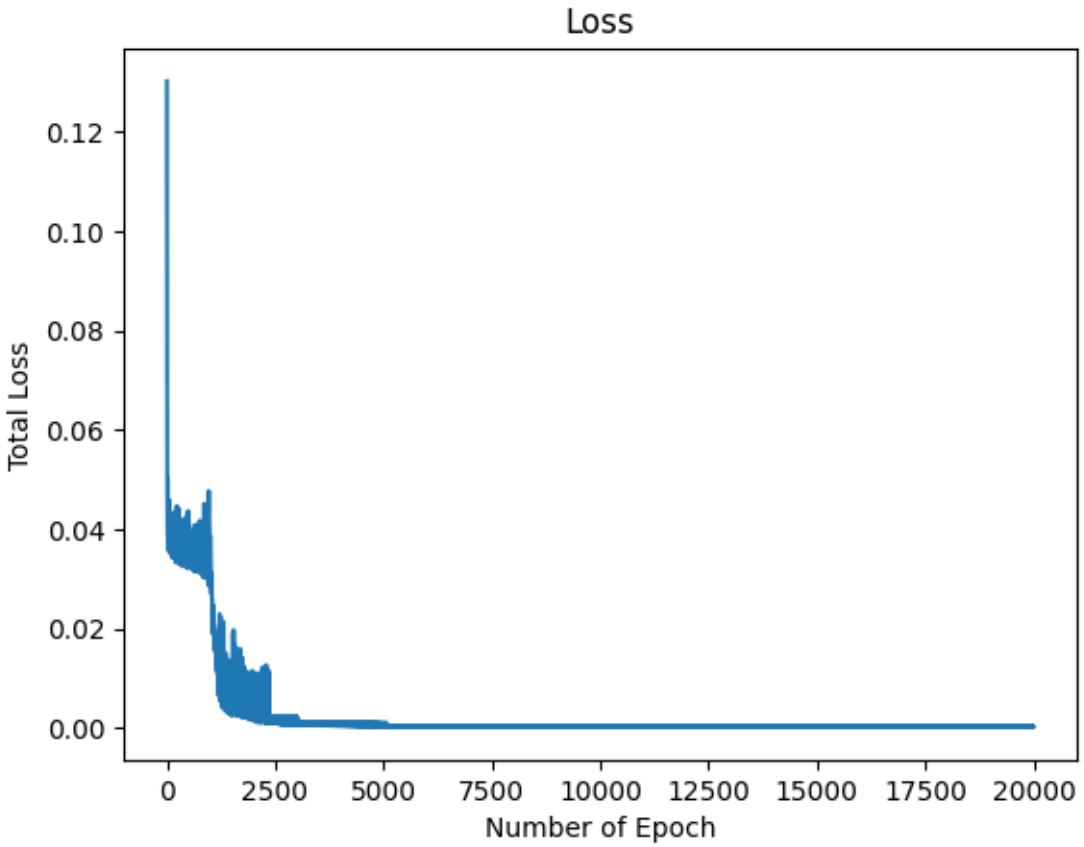


Figure 4.9: 1D Convection Diffusion Loss vs Epoch

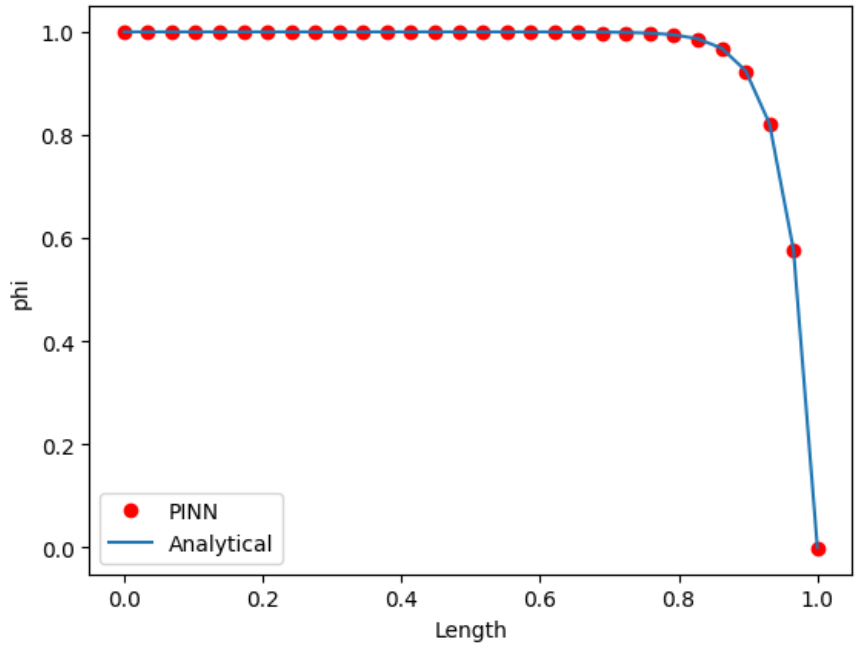


Figure 4.10: ϕ vs Length graph at u = 2.5 m/s

The PINN did results where in good accordance with the analytical even at high velocity where CFD usually fails if coarse grid is used. However PINN did take more time to converge at higher velocity.

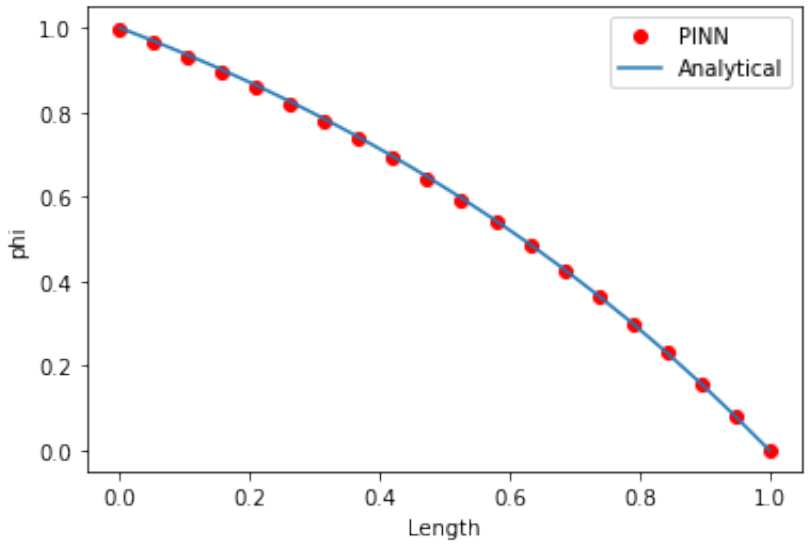


Figure 4.11: ϕ vs Length graph at u = 0.1 m/s

4.4 1D Nozzle Flow

4.4.1 Problem Statement

This problem is also taken from textbook [15]. A steady, frictionless flow is introduced in the nozzle as shown in figure below. Using PINN predict pressure and velocity and compare the prediction with analytical solution.

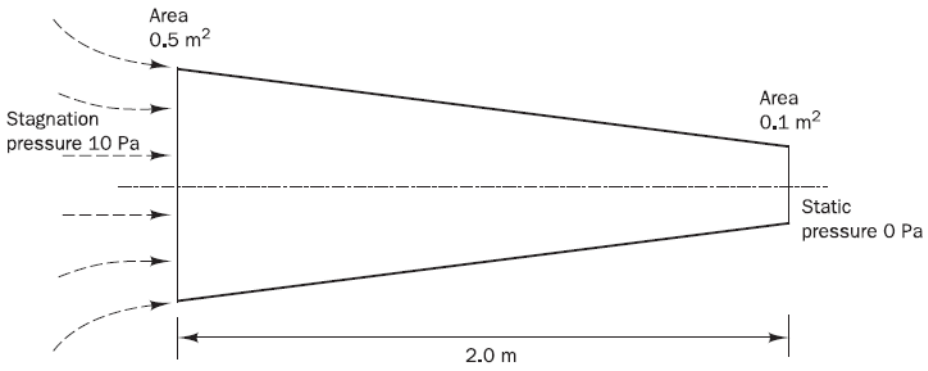


Figure 4.12: Nozzle [15]

4.4.2 Governing Equation

The problem 4.4 is governed by 1D Navier Stokes Equation which is given by:

$$\frac{\mathrm{d}}{\mathrm{d}x}(\rho u A) = 0 \tag{4.13}$$

$$\rho u A \frac{\mathrm{d}u}{\mathrm{d}x} = -A \frac{\mathrm{d}p}{\mathrm{d}x} \tag{4.14}$$

where u is the velocity in the x direction, A is the cross sectional area of the nozzle and p is the pressure.

4.4.3 Boundary Condition and Constant Values

In this problem statement statement stagnation pressure of 10 Pa was given. But PINN as of yet does not have the mathematical ability to correct pressure while training. CFD on the other hand uses pressure velocity coupling algorithm such as SIMPLE, SIMPLEC,

Parameter	Value
Inlet Pressure	0.5 Pa
Outlet Pressure	0 Pa
Length(L)	1 m
Inlet Area	$0.5m^2$
Outlet Area	$0.1m^2$

Table 4.9: Boundary Condition and Constant Values for Problem 4.4

COUPLE etc to correct pressure. Further research is required to incorporate a pressure correction feature into PINN. However if we use unsteady equations PINN will be able to correctly determine pressure.

As PINN is unable to correct pressure, we have to enforce corrected pressure and velocity at the boundaries. The corrected pressure and velocity is obtained using Bernoulli Equation given is Equation 4.20 and is mentioned in table 4.10

Parameter	Value
Inlet Pressure	0.48 Pa
Outlet Pressure	0 Pa
Inlet Velocity	0 m^2
Outlet Velocity	1 m^2

Table 4.10: Corrected Boundary Condition for Problem 4.4

4.4.4 Neural Network Parameters

Parameter	Value
Number of Hidden Layers	3
Number of Neuron in Hidden Layer	40
Number of Input in Input Layer	1
Number of Output in Output Layer	2
Number of Calculation points	100
Activation Function	tanh
Total Epoch	50000

Table 4.11: 1D Nozzle Flow NN Parameters

4.4.5 PINN Loss

The physics part of the loss function is given by the following two equations:

$$\frac{\mathrm{d}}{\mathrm{d}x}(\rho u A) = \text{Continuity Residue} \tag{4.15}$$

$$\rho u A \frac{\mathrm{d}u}{\mathrm{d}x} + A \frac{\mathrm{d}p}{\mathrm{d}x} = \text{Momentum Residue} \tag{4.16}$$

$$\text{Continuity Loss} = \frac{1}{n}(\sum_{i=1}^{n} \sqrt{(\text{Continuity Residue})^2}) \tag{4.17}$$

$$\text{Momentum Loss} = \frac{1}{n}(\sum_{i=1}^{n} \sqrt{(\text{Momentum Residue})^2}) \tag{4.18}$$

where n is the total number of data points where Navier Stokes is applied.

So the total Physics Loss is given by:

$$\text{Physics Loss = Momentum Loss + Continuity Loss} \tag{4.19}$$

4.4.6 Analytical Solution

The Analytical solution to this problem is given by Bernoulli's Equation

$$p_0 = p_n + \frac{1}{2}\rho u_n^2 = p_n + \frac{1}{2}\rho \dot{m}^2/(\rho A_n^2) \tag{4.20}$$

where $\dot{m}$ is mass flow rate and A is the area

4.4.7 Results and Validation

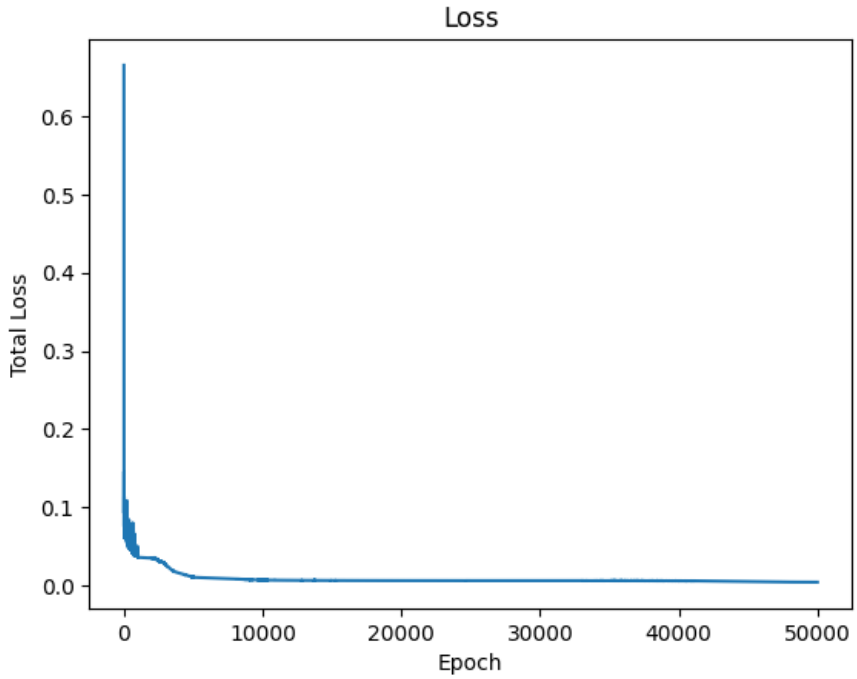


Figure 4.13: 1D Nozzle Loss vs Epoch

Even at 50000 epoch, the total loss was around 0.0032 which can be further improved by running more number of epoch.

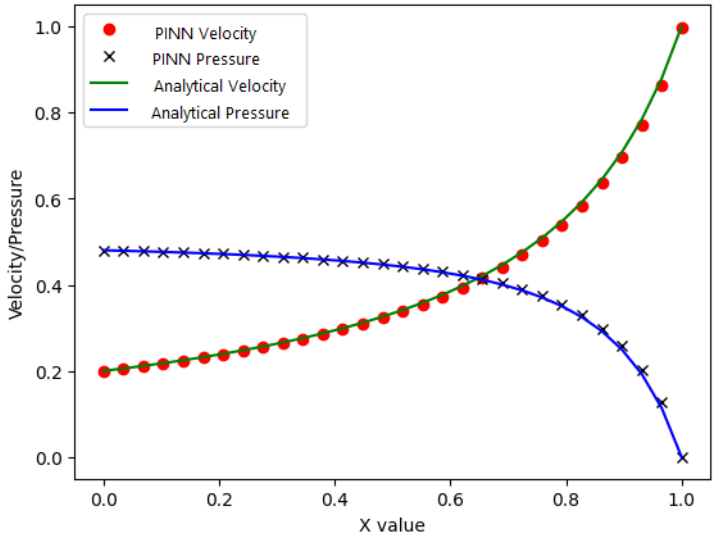


Figure 4.14: 1D Nozzle Velocity (m/s) and Pressure (Pa) Graph

4.5 2D Rectangular Duct

4.5.1 Problem Statement

Using PINN, we predicted the velocity profile and pressure of the flow through a rectangular duct as shown in Figure 4.15.

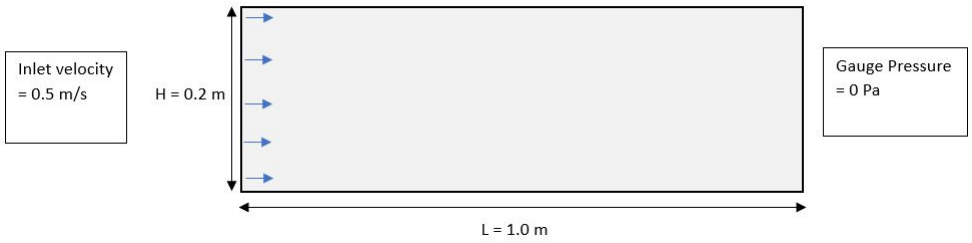


Figure 4.15: Rectangular Duct

4.5.2 Governing Equation

The problem 4.5 is governed by 1D Navier Stokes Equation which is given by:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{4.21}$$

$$\rho(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}) = -\frac{\partial p}{\partial x} + \mu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{4.22}$$

$$\rho(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}) = -\frac{\partial p}{\partial y} + \mu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) \tag{4.23}$$

4.5.3 Constant Values

Parameter	Value
Density(ρ)	1.225
Dynamic Viscosity(μ)	1.7894e-5

Table 4.12: Constant Values for Problem 4.5

Parameter	Value
Inlet Velocity	0.5 m/s
Outlet Pressure	0 Pa
Velocity at Walls(Top and Bottom)	0 m/s

Table 4.13: Boudary Conditions for Problem 4.5

Parameter	Value
Number of Hidden Layers	7
Number of Neuron in Hidden Layer	100
Number of Input in Input Layer	2
Number of Output in Output Layer	3
Number of Calculation points	5600
Activation Function	tanh
Total Epoch	50000

Table 4.14: 1D Nozzle Flow NN Parameters

4.5.4 Boundary Conditions

4.5.5 Neural Network Parameter

4.5.6 CFD Simulation

In order to validate the PINN results, a CFD simulation was performed on the problem domain. The parameters chosen for this problem result in a flow that is in the turbulent regime. The Reynolds number was taken to be Re = 1369.

Mesh

The mesh was created using ANSYS Fluent's meshing software. A simple structured mesh was given with mesh size of 2 mm all over the domain. The resultant mesh parameters are given in Table 4.15

Mesh Parameter	Value
Number of Nodes	50399
Number of Elements	49800
Average Aspect Ratio	1.004
Average Orthogonal Quality	0.9999

Table 4.15: Mesh Parameters

4.5.7 Simulation Setup

The following settings were used to setup the simulation:

Parameter	Value
Solver	Pressure Based
Turbulence Model	SST k-omega
Convergence Tolerance	1e-6
Pressure Velocity Coupling	SIMPLE
Discretization Scheme	Second Order Upwind

Table 4.16: CFD Simulation Settings

4.5.8 PINN Domain

The domain created for PINN is given below:

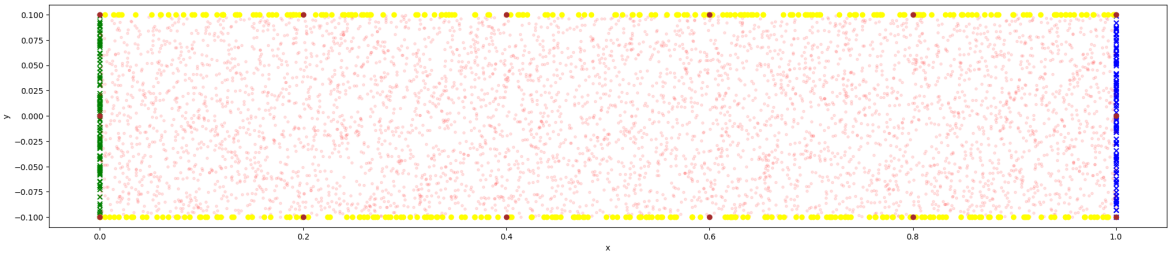


Figure 4.16: PINN Domain

The red points are where no external data is available other than Navier-Stokes, green points denote inlet, blue points denote outlet, yellow denote wall and brown points are where the pressure data was given.

4.5.9 PINN Loss

This problem is governed by 2D Navier-Stokes equation, therefore, the physics loss is calculated as:

$$\rho(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}) + \frac{\partial P}{\partial x} - \mu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) = \text{X Momentum Residue} \tag{4.24}$$

$$\rho(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}) + \frac{\partial P}{\partial y} - \mu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) = \text{Y Momentum Residue} \tag{4.25}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = \text{Continuity Residue} \tag{4.26}$$

$$\text{Physics Residue} = \text{X Momentum Residue} + \text{Y Momentum Residue} + \text{Continuity Residue} \tag{4.27}$$

$$\text{Physics Loss} = \frac{1}{n}\left(\sum_{i=1}^{n} \sqrt{(\text{Physics Residue})^2}\right) \tag{4.28}$$

4.5.10 Results and Validation

The PINN results were validated against CFD results.

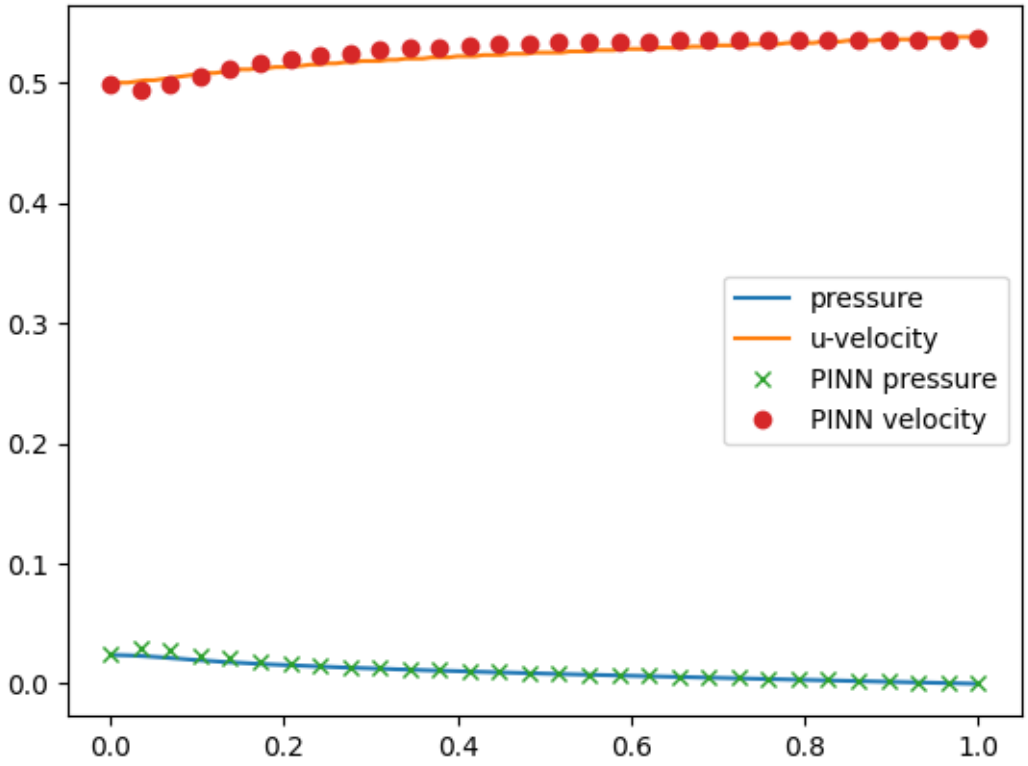


Figure 4.17: Plot of u-velocity at y = 0 m

From Figure 4.17, it can be concluded that the PINN behaves as good as CFD, but is deviating from CFD results at entry region.

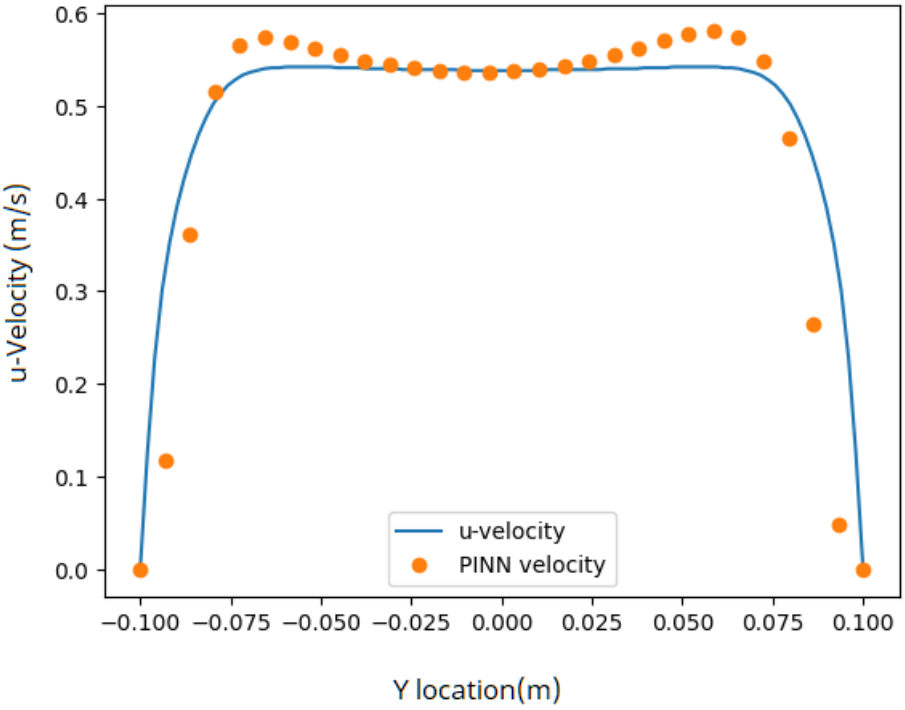
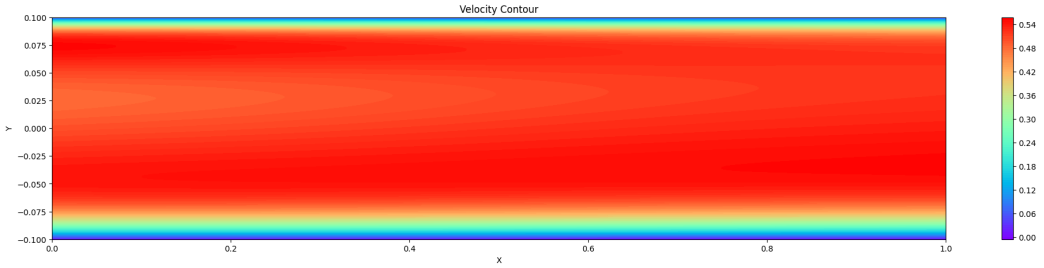


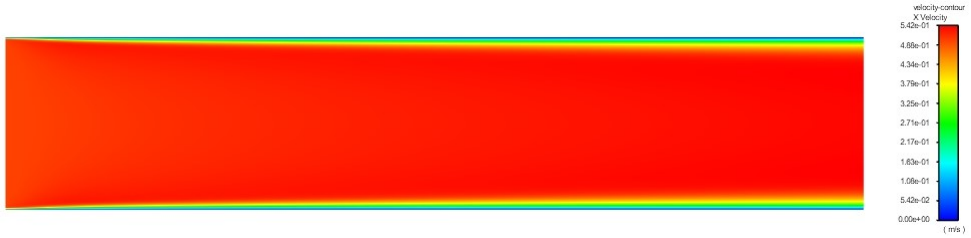
Figure 4.18: U-velocity at outlet

The velocity prediction is deviated from CFD results with significant amount. Only at point where pressure data was enforced as data in loss function was predicting with acceptable error.

The PINN prediction fails where there is developing flow as it is clear from Figure 4.17, 4.18, 4.19. Moreover it required pressure data to be enforced on the boundaries even at the boundaries. The training time was also around 40 minutes with GPU acceleration.



(a)



(b)

Figure 4.19: Contour Plot of u-velocity (a) PINN plot (b) CFD plot

4.6 Backward Facing Step

4.6.1 Problem Statement

The Backward Facing Step Problem is a classical benchmarking problem. The problem itself was taken from Gartling [6]. The computational domain is taken to have height H equals 1 m and a step height and upstream inlet of 0.5 m (H/2). The downstream channel was given the length L = 30m. The total length of the domain was taken to be 60 m. The inlet velocity field was defined to be a parallel flow with a parabolic velocity profile given by u(y)= 24y (0.5 - y) for 0 <y <0.5. The maximum inflow velocity is 1.5 and the average inflow velocity of 1. The boundary condition at the outlet also has a parallel flow and a constant total stress that is normal to the boundary. The outflow pressure was set to 0 Pa. Using PINN we predict the velocity and pressure at the downstream channel.

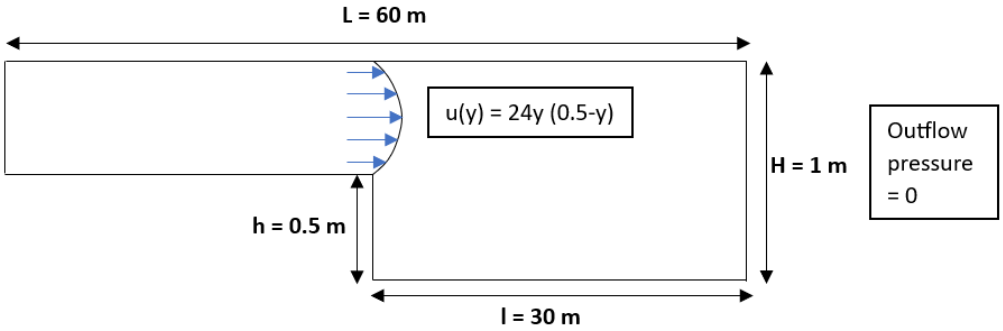


Figure 4.20: Backward Facing Step

4.6.2 Governing Equation

The governing equation is same as the equations used for Problem 4.5 that is 2D Navier-Stokes equation given by Equation 4.21, 4.22, and 4.23.

4.6.3 Boundary Condition

Parameter	Value
Inlet Velocity	$u(y) = 24y(0.5 - y)\ m/s$
Outlet Pressure	0 Pa
Velocity at Walls	$0\ m/s$

Table 4.17: Boundary Conditions for Problem 4.6

4.6.4 Constant Values

In order to remain in the laminar regime, Re number was set to 800 and then the constants were calculated using equation $Re = \rho u_{avg} D/\mu$.

Parameter	Value
ρ	$u(y) = 24y(0.5 - y)\ m/s$
μ	0.00125
Downstream Channel Length	$30\ m$

Table 4.18: Boundary Conditions for Problem 4.6

4.6.5 CFD Simulation

The Neural Network results were validated against CFD results and CFD results from reference [6].

Mesh

In order to save on the computational resources, the simulation was done on the downstream channel only. The following mesh was obtained:

Mesh Parameter	Value
Number of Nodes	303101
Number of Elements	300000
Average Aspect Ratio	1.024
Average Orthogonal Quality	0.9999

Table 4.19: Mesh Parameters

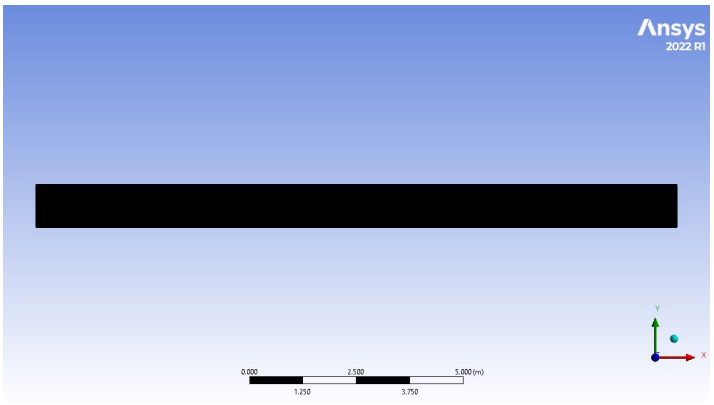


Figure 4.21: (a)

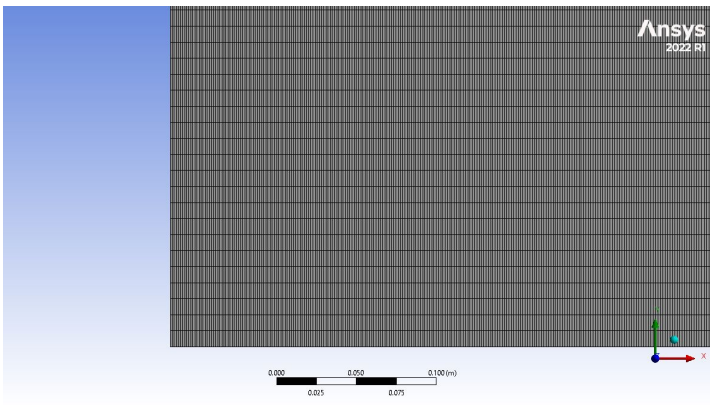


Figure 4.22: (b)

Figure 4.23: Backward Facing Step Mesh 4.6 (a) The Mesh (b) Zoomed view at the inlet

4.6.6 Simulation Setup

The following settings were used to setup the simulation:

Parameter	Value
Solver	Pressure Based
Turbulence Model	Laminar
Convergence Tolerance	1e-3
Pressure Velocity Coupling	SIMPLEC
Discretization Scheme	Second Order Upwind

Table 4.20: CFD Simulation Settings

4.6.7 PINN Domain

Figure 4.24 shows the domain created to training PINN model:

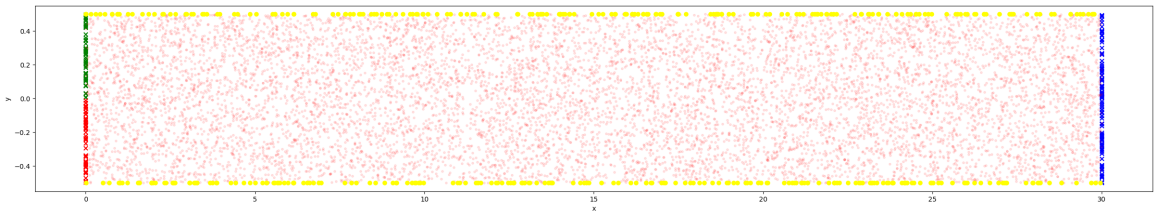


Figure 4.24: PINN Domain for Problem 4.6

The green points are where boundary condition is applied, red and yellow points are walls, and blue point is outlet.

4.6.8 PINN Loss

The physics part of the PINN loss is same as in Problem 4.5.

4.6.9 Neural Network Parameter

4.6.10 Results and Validation

The PINN prediction is in accordance with the CFD results with negligible deviation as can be seen in Figure 4.25

Parameter	Value
Number of Hidden Layers	10
Number of Neuron in Hidden Layer	100
Number of Input in Input Layer	2
Number of Output in Output Layer	3
Number of Calculation points	10000
Activation Function	tanh
Total Epoch	20000

Table 4.21: Neural Network Parameters for flow over backward facing step

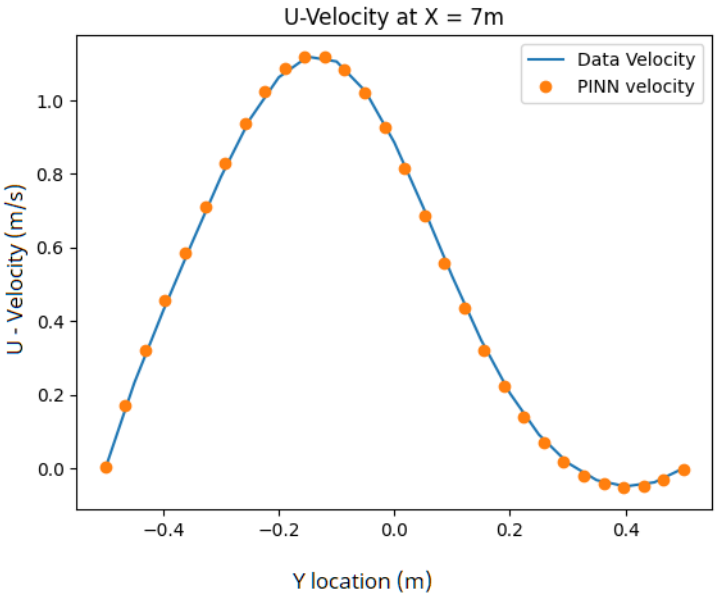
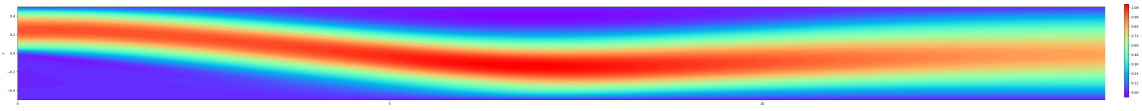
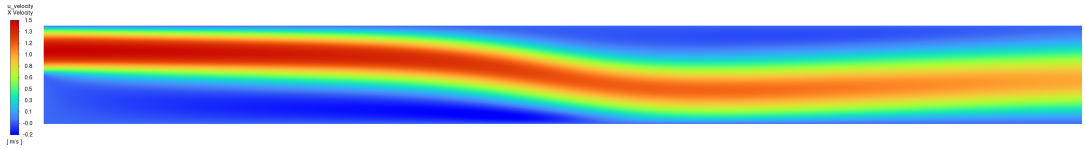


Figure 4.25: U-velocity at x = 7 m



(a) PINN



(b) CFD

Figure 4.26: U-velocity contour

Chapter 5

CONCLUSION

In this study, we investigated the accuracy PINN obtains on benchmark problems and simple geometries. Specifically, PINN models were used to predict six different problems: one-dimensional heat conduction, one-dimensional convection-diffusion, one-dimensional unsteady heat conduction, one-dimensional nozzle flow, two-dimensional rectangular duct, and the Backward Facing Step Problem. To assess the accuracy of the PINN-generated solutions, we compared them to analytical solutions and solutions produced by computational fluid dynamics (CFD). The results showed that the solutions predicted by PINN were in good agreement with the actual solutions, indicating that PINNs have the potential to be a viable alternative to CFD in the future.

The ability of PINNs to accurately predict solutions for these benchmarked problems and simple geometries is a promising development in the field of computational fluid dynamics. It suggests that PINNs may be able to provide accurate and efficient solutions for a wide range of fluid flow problems, which could have significant practical applications in areas such as aerospace engineering, ocean engineering, and environmental engineering.

One of the challenges in training the PINNs, was scaling the inputs and outputs to an appropriate range. The PINN output is required to be within the range of [-1,1] because of activation function used in the output layer. Tanh is an extremely handy function used widely in neural networks. However, the output of the PINN didn't always stay in the range, which resulted in distortion of the solution. The need for a different activation that does not have such limitations arises.

The PINN model is unable to predict accurate solutions for certain equations when proper support in terms of data is not provided. Even though the loss was being driven to zero, the model had trouble predicting the Backward Facing Step correctly. The model

needed data to predict the solution accurately. The reason for this behaviour could be the reattachment and circulation of the flow.

While it is true that the training of PINN models can be computationally expensive, the use of GPUs can significantly reduce the training time and make it feasible to use these models for practical applications. GPUs are well-suited for the type of parallel computations involved in training neural networks and can greatly speed up the computation compared to using only CPUs. However, it is also true that at the moment the computational cost of PINNs is still higher than that of traditional computational fluid dynamics (CFD) methods. PINNs may not always have an advantage over traditional CFD methods in terms of computational cost, upon high amounts of training they offer an advantage in terms of handling complex geometries and sparse data.

Chapter 6

FUTURE ENHANCEMENT

There are several future improvements that could be made to this project, including:

1. Extension to higher Reynolds numbers: As previously mentioned, the project is only currently capable of handling flows with low Reynolds numbers. Nevertheless, expanding the study to higher Reynolds numbers would allow the investigation of more complex and challenging fluid dynamics problems. This would demand the creation of more complicated neural network models capable of handling additional complexity.

2. Multi-phase flow simulations: Simulations of multi-phase flows are common computational fluid dynamics problems including the flow of fluids with suspended particles or the interaction of fluids with solid surfaces. The need for a PINN model that can handle multi-phase flows arises.

3. Optimization of Neural Network architecture: The performance of Neural Networks is highly dependent on the architecture of the model. Changes are to be made to the Neural Network architecture to make it more efficient and accurate.

4. Application to real-world problems: Despite being a well-known benchmarking issue, the Backward Facing Step problem is still a simplified version of actual fluid dynamics problems. The practical uses of the PINN and its potential might be shown by applying it to real-world. However, that would still take a very long time and extensive research.

References

[1] Christopher J Arthurs and Andrew P King. Active training of physics-informed neural networks to aggregate and interpolate parametric solutions to the navier-stokes equations. *Journal of Computational Physics*, 438:110364, 2021.

[2] Amirhossein Arzani, Jian-Xun Wang, and Roshan M D'Souza. Uncovering near-wall blood flow from sparse data with physics-informed neural networks. *Physics of Fluids*, 33(7):071905, 2021.

[3] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

[4] Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6), 2021.

[5] Hamidreza Eivazi, Mojtaba Tahani, Philipp Schlatter, and Ricardo Vinuesa. Physics-informed neural networks for solving reynolds-averaged navier–stokes equations. *Physics of Fluids*, 34(7):075117, 2022.

[6] David K Gartling. A test problem for outflow boundary conditions—flow over a backward-facing step. *International Journal for Numerical Methods in Fluids*, 11(7):953–967, 1990.

[7] Rini J Gladstone, Mohammad A Nabian, and Hadi Meidani. Fo-pinns: A first-order formulation for physics informed neural networks. *arXiv preprint arXiv:2210.14320*, 2022.

[8] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.

[9] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.

[10] Jan Oldenburg, Finja Borowski, Alper Öner, Klaus-Peter Schmitz, and Michael Stiehm. Geometry aware physics informed neural network surrogate for solving navier–stokes equation (gapinn). *Advanced Modeling and Simulation in Engineering Sciences*, 9(1):8, 2022.

[11] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[12] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. 2017.

[13] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: A navier-stokes informed deep learning framework for assimilating flow visualization data. *arXiv preprint arXiv:1808.04327*, 2018.

[14] Vinothkumar Sekar, Qinghua Jiang, Chang Shu, and Boo Cheong Khoo. Fast flow field prediction over airfoils using deep learning approach. *Physics of Fluids*, 31(5):057103, 2019.

[15] Henk Kaarle Versteeg and Weeratunge Malalasekera. *An introduction to computational fluid dynamics - the finite volume method.* Addison-Wesley-Longman, 1995.

[16] Zixue Xiang, Wei Peng, Xiaohu Zheng, Xiaoyu Zhao, and Wen Yao. Self-adaptive loss balanced physics-informed neural networks for the incompressible navier-stokes equations. *arXiv preprint arXiv:2104.06217*, 2021.

[17] XIA Yang, Suhaib Zafar, J-X Wang, and Heng Xiao. Predictive large-eddy-simulation wall modeling via physics-informed neural networks. *Physical Review Fluids*, 4(3):034602, 2019.