DEEP LEARNING ENHANCED CFD FOR MODELING FLUID FLOW

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

B.Tech

in

Mechanical Engineering

By

Adithya B (111118001)

Gokulnath S (111118033)

Hirthick Kumaran N (111118039)

MECHANICAL ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI – 620015

MAY 2022

BONAFIDE CERTIFICATE

This is to certify that the project titled **DEEP LEARNING ENHANCED CFD FOR MODELING FLUID FLOW** is a bonafide record of the work done by

Adithya B (111118001)

Gokulnath S (111118033)

Hirthick Kumaran N (111118039)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Mechanical Engineering** of the **NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI**, during the year 2018-19.

PROJECT GUIDE

Dr R B Anand

DEPARTMENT HOD

Dr A R Veerappan

Project Viva-voce held on _____

Internal Examiner

External Examiner

ABSTRACT

Machine learning is quickly becoming a key tool in scientific computing, with several applications in computational fluid dynamics. This study focuses on some of the most promising topics, such as accelerating direct numerical simulations. As a test case, the steady covection diffusion issue was utilised to determine the amount of complexity to which a technique may give a solution. Finally, the focus is on physics-based machine learning algorithms that allow for the solution of differential equations without the need to get high resolution computed data.

Regardless of critical advancement in recreating stream issues utilizing mathematical discretization of the Navier-Stokes conditions (NSE) throughout the course of recent years, we actually can't integrate loud information consistently into existing calculations, network age is troublesome, and high-layered issues administered by parametrized NSE are difficult to settle. Furthermore, handling inverse flow issues is sometimes prohibitively expensive and necessitates complicated and costly formulations as well as novel computer codes. We examine flow physics-informed learning, which integrates data and mathematical models easily and is implemented using physics-informed neural networks (PINNs). We show how PINNs may be used to solve inverse issues.

Keywords: Machine Learning, Computational Fluid Dynamics, Neural Network, PINN, Data-Driven, Navier-Stokes, Convection-Diffusion

ACKNOWLEDGEMENTS

We would like to express our deepest gratitude to the following people for guiding us through this course and without whom this project and the results achieved from it would not have reached completion.

We thank our guide, **Dr R B Anand**, Professor, Department of Mechanical Engineering, for helping us and guiding us in the course of this project. Without his guidance, we would not have been able to successfully complete this project. His patience and genial attitude is and always will be a source of inspiration to us.

Dr A R Veerappan, the Head of the Department, Department of Mechanical Engineering, for allowing us to avail the facilities at the department.

We are also thankful to the faculty and staff members of the Department of Mechanical Engineering, our individual parents and our friends for their constant support and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Below contains the exhaustive list of symbols used in this literature

Symbol	Meaning
u	Transported property
x	Spatial variable
c	Advection strength
D	Diffusion strength
Pe	Peclet number
u_L	u @ x = 1
u_0	u @ x = 0
ϕ	Transported variable
x, y	Spatial variables
u	Advection strength in x-direction
v	Advection strength in y-direction
K	Isotropic Diffusion strength
x, y	Spatial variables
u	x-component of velocity
v	y-component of velocity
p	Pressure
Re[$= \frac{\rho u L}{\nu}$]	Reynolds number

Table 1: Symbols

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Computational Fluid Dynamics (CFD) is the discipline of simulating, predicting, and analysing fluid flows, heat transfer, mass transfer, chemical reactions, and other related phenomena by solving physics-based governing equations for a large number of finite points or finite volume cells. Ordinary differential equations (ODEs) and partial differential equations (PDEs) are the most common governing equations (GDEs). The exact solution to the majority of ODEs and PDEs remains unknown. Conventionally, various numerical approximation approaches are utilised to solve a large number of such differential equations. Differential equations are discretized to get a algebraic equation, which is a close approximation of itself. Solving such sets of equations for flow field variables is a time-consuming and computationally exhaustive step.

Machine learning is a notion that can assist us in addressing the challenges listed above. In fact, machine learning models could be used to perform a variety of other tasks, such as generating a high resolution solution from a coarse grid solution or a corrupt solution, predicting correction, predicting unpredictable parameters, and performing data-driven flow-field analysis, to name a few. Here, we focus on the potential for machine learning to enhance CFD, such as increasing the speed of high-fidelity simulations, developing turbulence models with varied degrees of fidelity, and producing lower order models beyond what is possible with traditional techniques.

1.2 NEURAL NETWORK

A neural network is a set of algorithms that attempts to detect underlying relationships in a batch of data using a method that mimics how the human neural works. Neural networks, in this context, refer to systems of neurons that can be biological or artificial in nature.

Since neural network can adjust to evolving input, they can create the most ideal result without requiring the result rules to be upgraded. The computerized reasoning based idea of neural networks is rapidly building up some forward movement in the making of exchanging frameworks.

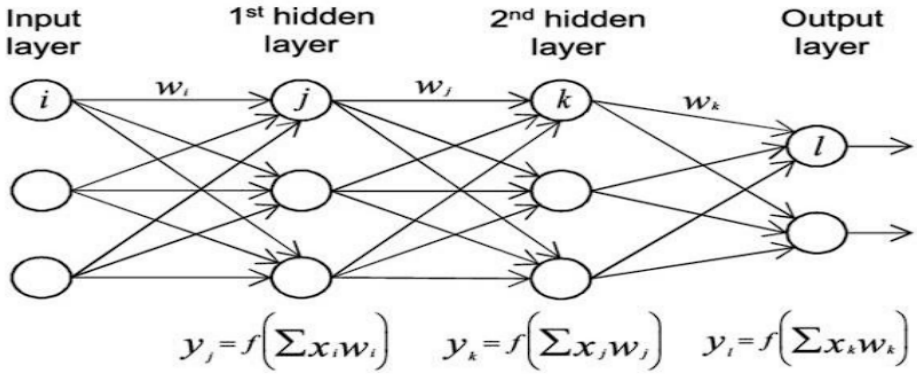


Figure 1.1: Neural Network Architecture

The architecture's complexity enables for a higher number of weights and biases to be adjusted. Because it has more controllers, the network is more tuneable. We train the network to discover the values that minimise the output error because the weights and biases cannot be set manually. It's akin to figuring out the weights and biases for fitting the function to our data (X, Y).

1.3 NEURAL NETWORK AS FUNCTION APPROXIMATORS

Although neural networks have a wide range of applications, the problems described in this thesis fall within the regression or function approximation group. The general guess hypothesis says that, expecting moderate presumptions on the enactment work, a feed-forward network with a solitary secret layer containing a set number of neurons might rough consistent capacities on conservative subsets of Rn. In 1989, George Cybenko proposed the primary sigmoid initiation capacities. Truly, Kurt Hornik[1] exhibited in 1991 that it is the multi-facet feedforward design, not the particular decision of the actuation work, that gives the possibility to be an all inclusive approximator.

1.4 DEEP LEARNING

The growing usage of neural networks as function approximators [1] and numerical solvers has opened up new possibilities for machine learning and numerical analysis. As a result, a rudimentary grasp of neural networks is required for the objectives of this study. We may acquire intuition for the inner workings of a deep neural network by understanding how a single perceptron - single artificial neuron - generates predictions and gets taught.

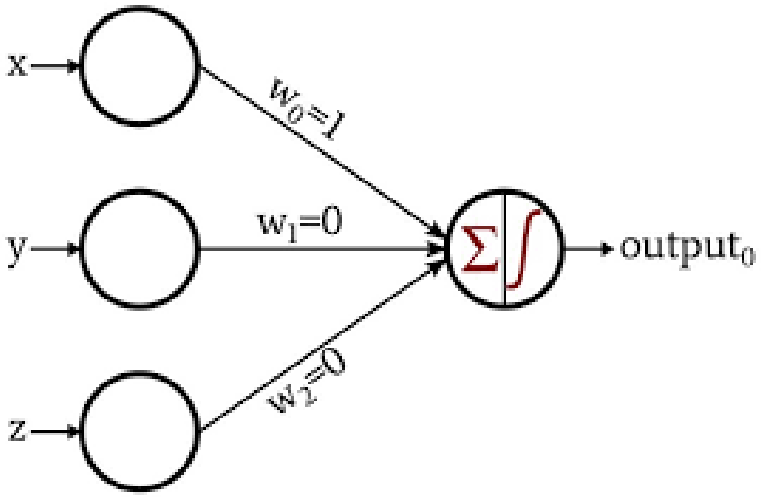


Figure 1.2: The structure of the perceptron.

The most basic neural network architecture is the perceptron. It is made up of a single neuron with a function of activation [2]. Let x represent the perceptron's inputs, w represent the weights, b represent the bias, and g represent the activation function. Before applying g, the perceptron takes input x, computes the weighted sum wx—, and adds the bias, as shown in Figure 1. As a result, our perceptron predicts the following for inputs x:

$$h(x) = g(wx + b), \tag{1.1}$$

where h (x) represents the output for the perceptron's given input x and w denotes weight of the model and b denotes bias.

A layer in a neural network is characterized as an assortment of neurons that each independently cycle the information sources and gives a particular result. Thusly, the result of a layer is a vector whose dimensionality is equivalent to the quantity of neurons in that given layer. While only one layer can be utilized to make expectations, scientists frequently select to make their neural organizations "profound" to work on the adaptability and demonstrating capacities of the organization. Whenever a neural network is alluded to as "profound" it suggests that there are various layers between the info and result layer [2].
Considering that every neuron has a weight vector used to compute its result, we make a "weight network" to numerically depict the 5 assortment of weight vectors in each layer. The instinct behind utilizing profound neural networks rather than simple designs is that lower layers recognize low-level examples in the information and as data is engendered through the neural organization, higher layers can distinguish the high level designs in the information that are important for exact foreseeing.

Similarly vital to the construction of the neural network is the interaction it goes through to be prepared and decide the ideal assortment of loads for our concern.

1.5 WORKING OF THE MODEL

A training set and a loss function are both required to train a neural network. The training set consists of training inputs as well as possible goal values for each associated input. The training process is known as supervised learning if a training set has goal values for its inputs, otherwise it is known as unsupervised learning [2]. When goal values are supplied, we shall elaborate on the optimization procedure for the purposes of our article.

We want to give an outline of how an arbitrary loss function is optimised because each job has a distinct set of acceptable loss functions. The weights of a neural network must be optimised in two steps: forward and backward passes. The inputs are sent to the neural network and transmitted through the network in the forward pass, resulting to a final prediction. Following the prediction, the error is calculated using the chosen loss function in relation to the true (target) value. The loss function's gradient is then computed with respect to each weight, and the weights are modified in the direction specified by the gradient.

To minimise our loss function, the gradient indicates the "direction" in which the weights should be modified [2]. The weights are continually updated while this procedure is repeated, and the loss function's gradient with respect to the weights is recalculated. As a result, we anticipate eventually arriving at a set of weights that suitably minimises our loss function. Stochastic gradient descent is the process of computing gradients and modifying weights based on the gradient of the loss function. Before adjusting the weights, it is usual in machine learning to scale the gradient by a factor, commonly known as the learning rate. The rate at which the model adapts to the challenge is controlled by the learning rate [2].

One of the most well-known ways to deal with working on the intricacy of the neural network is expanding the quantity of layers. While we can likewise expand the quantity of neurons per layer, it is regularly acknowledged that rising the quantity of layers instead of the quantity of neurons per layer will be more remunerating [2]. In any case, given the intricacy of physical science informed machines, we will differ both building parts and rethink the model. In this part we want to decide the ideal number of layers and neurons for which the model accomplishes high precision however, not at the expense of high intricacy.

1.6 OBJECTIVE OF THE PROJECT

We concentrate on speeding up computational techniques and developing novel methods to reduce computational power requirements for general flow related problems with the aid of Deep Learning models without compromising on accuracy. We no longer see deep learning models as black boxes. Now that the AI engine depends on notable, confirmed equations, the framework's inward functions are as of now not a total secret. This produces another entity, a "grey box," in which it is as yet difficult to depict the exact thing is happening inside, yet it is likewise not entirely obscure, because of the way that it depends on notable actual principles of physics. With this tool at our disposal, we aim to improve computational efficiency and reduce computational power requirements to help fellow researchers push their works in flow related problems beyond the "limit".

CHAPTER 2

LITERATURE REVIEW

2.1 NUMERICAL SOLUTIONS

It's vital to grasp conventional methods for generating numerical solutions to PDEs before diving into a full explanation of a physics-informed approach to the convection diffusion equation. The finite difference and element approaches are the most often utilised. Both approaches approximate numerical solutions to PDEs by breaking the issue down into smaller pieces that, when solved, numerically approximate the PDE solution. The approaches are different because the former employs Taylor's series approximation while the later takes an integral technique to approximate the function [3].

Consider the capacity f, which has just a single variable (x). We should assess the subordinates of f while planning a limited contrast procedure. We really want to parcel the area into uniform - or non-uniform - time frames before we can give evaluations to the subordinates of f. The cross section is the result of isolating the space, time, and other aspects' area in higher-layered PDEs [4]. The progression size or cross section width is the size of the spans in each aspect. While parting the area, it's basic to decide step-estimates accurately since, as we'll see later, they could influence the mathematical technique's execution and rightness [5].

How about we call the lattice width h. We can gauge the subsidiaries of f utilizing this technique. We might utilize three unmistakable limited contrast ways to deal with infer the subsidiaries of f: forward, in reverse, and focused approximations. Every strategy indicates whether we are pushing ahead or in reverse in each aspect, or on the other hand on the off chance that we are adopting a focused strategy with no forward or in reverse "development." Essentially, to ascertain the subordinate of f at a given area x, the focused procedure will use a worth of f to the right and left of x rather than f (x) in every calculation. Accordingly, the three methodologies for assessing the principal subsidiary of f are:

$$\frac{\partial f}{\partial x+} = \frac{f(x+h) - f(x)}{h} \tag{2.1}$$

$$\frac{\partial f}{\partial x-} = \frac{f(x) - f(x-h)}{h} \tag{2.2}$$

$$\frac{\partial f}{\partial x+} = \frac{f(x+h) - f(x-h)}{h} \tag{2.3}$$

The forward, backward, and centred methods are denoted by (2.1), (2.2), and (2.3), respectively. The greatest difference between the centred finite difference approach and the other two methods is that (2.1) and (2.2) employ the value of f (x) in their computations, whereas (3) does not. More accurate estimates exist, but they are typically accepted if the step-size is small enough [6]. (2.1), (2.2), and (2.3) are known as first order accurate approximations. Higher-order derivatives have similar formulas that may be used to numerically solve the PDE. It's critical to remember that finite difference approaches can only estimate the function value at grid - mesh - points determined by our mesh width, not at all points in the domain (s).

Limited contrast techniques are generally used to tackle PDEs and have likewise been applied to choices valuing. Straightforward limited distinction strategies, for example, (2.1), (2.2), and (2.3) and their more elevated level reciprocals can be utilized for guess however specialists have consistently looked to work on these essential mathematical techniques for more convoluted PDEs, like free limit issues. Also, while limited contrast techniques have been utilized to cost American choices, dependability also, union issues can emerge from inappropriate cross section age [10] or space definition. The issues that emerge from limited distinction techniques added to the advancement of a comparable strategy known as the limited component technique.

2.2 MACHINE LEARNING

Machine learning (ML) gives an assortment of techniques to removing data from information, which may therefore be transformed into information about the fundamental material science. As a result, machine learning algorithms can supplement domain knowledge and automate tasks related to flow control and optimization, resulting in a powerful information processing framework that can complement, if not completely transform, current lines of fluid dynamics research and industrial applications [6].The current scenario might be described as "data-rich, knowledge-poor": the underlying physics is so complicated that learning and extracting new information from a vast amount of observable data is nearly difficult [7].

When high-fidelity data is used to reduce uncertainties and/or errors in low-fidelity simulations, machine learning is used. In other words, using high-fidelity codes to calibrate low-fidelity codes is a DD technique [8]. This framework necessitates the availability of sufficient high-fidelity data (or other information about a solution with higher accuracy), which is a constraint for real engineering applications

(e.g., Direct Numerical Simulations (DNS) are still computationally expensive), as well as a substantial knowledge base for selecting appropriate CRs (if applicable) to inform the low-fidelity simulations (e.g., an appropriate turbulence model for the Reynolds-Avera equation).

Essentially, one can address this PDE by utilizing mathematical methodologies, for example, e.g., limited volume or limited component strategies. For a cross section free profound learning technique, we intend to assess the mathematical arrangement u(t, x) by planning the underlying issue as a streamlining issue and inferring the misfortune work L() related to the PDE issue with the comparing limitations. We then develop a profound brain organization to estimated the ideal arrangement of the misfortune work by a fitting improvement procedure, for example, the Stochastic Gradient Descent calculation. It implies that we need to assess an answer u(t, x) by an approximated work f(t, x), not set in stone by a profound brain network with the relating set of boundaries remembering the loads and inclination terms for each layer. Here, t and x are characterized as the information of such a brain organization.

CHAPTER 3

PHYSICS INFORMED LEARNING

3.1 PINN

PDEs (partial differential equations) describe the behaviour of a wide range of dynamical systems in our environment. PDEs are incredibly prevalent and helpful, but they are also quite difficult to solve. We need to utilise numerical analysis to approximate the solution values at various locations in the parameter space since not all PDEs have straightforward analytical solutions. Standard forward and backward finite difference methods, finite element methods, and Runge-Kutta methods are examples of traditional numerical analysis methods. These traditional approaches, although providing sufficient results, are computationally costly. Researchers have turned to physics-informed neural networks (PINNs) to generate numerical solutions to PDEs as our understanding of deep neural networks and automated differentiation has improved.

The traditional neural networks are based on data that is supplied into them. The model is just attempting to fit into the data. To train a neural network, you don't need to know anything about the physics of the problem. This is why there is little noise, such as oscillations, in the vicinity of the precise answer. It is critical that the neural network function learns the problem's physics.

Raissi et al. [20, 21] and Lagaris et al. [13] proposed PINNs as a method for obtaining numerical solutions to PDEs. The ability of deep neural networks to be universal function approximators [11, 15] and continual developments in automated differentiation [3] to compute gradients of the neural network's outputs with regard to its inputs enable us to model PDEs using this newly developed technique. Because they are obligated to preserve any conservation principles or invariances arising from the physical laws that constrain the observed data, PINNs increase our capacity to generate numerical solutions for PDEs. As we just indicated, PINNs expand our computational scientific capabilities and give more resources for mesh-free numerical solvers with enhanced data efficiency [20, 21].

An optimization problem involves solving a regression problem. The most straightforward technique to include physics in the issue is to include it in the optimization objective. In 1997, I.E. Lagaris et al. proposed a method for doing so. The solution of a differential equation is viewed as an optimization issue using a neural network function, rather than a training problem. The benefit of this technique

is that it does not necessitate any data in order to train the requisite neural network. For training, only the values of independent variables are used. The optimization target must also take into account the beginning and boundary conditions.

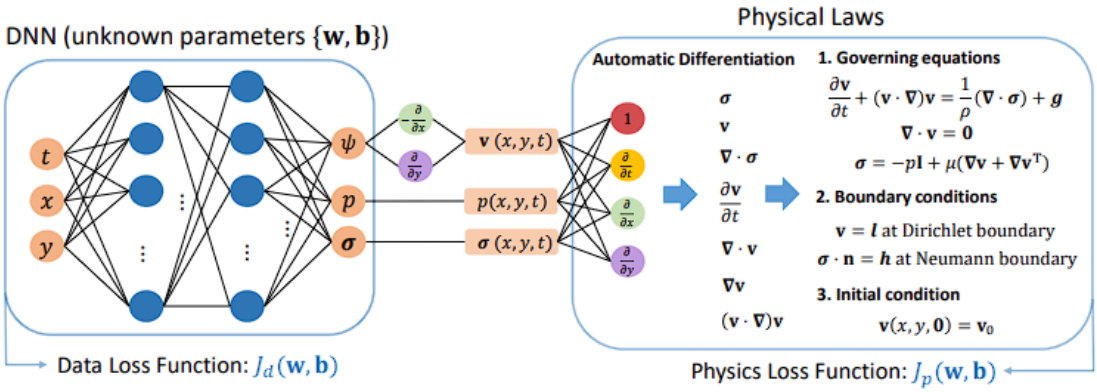


Figure 3.1: Architecture of the physics-informed neural network for fluid dynamics.

All the more as of late, Raissi et al. [5, 6] presented a general structure of PINN and exhibited its ability in demonstrating complex actual frameworks, for example, addressing/recognizing PDEs. A colossal contrast from a portion of the past examinations is that, notwithstanding the actual regulations, the PINN can likewise take advantage of the accessible estimation information, making it conceivable to find the frameworks whose material science are not completely perceived. In specific, fundamental commitments of utilizing PINN to model fluid flows have been made as of late. For instance, Kissas et al. [10] utilized PINN to foresee the blood vessel circulatory strain in view of the MRI information furthermore, the preservation regulations. Sun et al. [11] proposed a PINN approach for substitute demonstrating of fluid flows with practically no reenactment information. Zhu et al. [12] proposed a material science obliged convolutional encoder decoder organization and a generative model for displaying of stochastic fluid flowss

3.2 PHYSICAL CONSTRAINTS

In 2017, Maziar Raissi et al. offered an improvement to the approach described by Lagaris et al. The suggested solution overcomes the drawbacks of Lagaris' method, which were discussed at the end of the previous part. The following are the main enhancements:

To meet the boundary constraints, additional loss terms are calculated as mean squared error at borders. As a result, the approach may be used to nearly all types of ODEs and PDEs. As extra mean squared error loss terms, a variety of boundary constraints might be imposed. There's no need to change the trial function because the neural network can do it all by itself.

The second advantage of employing an extra loss term for border restrictions is that it allows for boundary relaxation during optimization, making convergence easier and simpler. Though there will be some departure from the boundary values, as the optimization progresses, the values begin to closely equal the precise values.

Initial conditions, similar to boundary restrictions, might be enforced by adding extra loss terms as mean of squared errors of the initial value data. This enhancement allows it to be used to tackle time-varying issues such as convection and burgers.

This model's design allows for deeper layers, allowing it to handle complicated and non-linear functions.

3.3 DEEP LEARNING

Here we focus on the connection with physical models, and there are lots of great introductions to deep learning. Hence, we'll keep it short: the goal in deep learning is to approximate an unknown function

$$f(x) = y \tag{3.1}$$

We regularly upgrade, for example train, with a stochastic inclination plunge (SGD) enhancer of decision, for example Adam analyzer. We'll depend on auto-differtiation to figure the slope of a scalar misfortune L w.r.t. the loads, $dL/d\theta$ We will likewise expect that e indicates a scalar mistake work (additionally called cost, or goal work). It is critical for the effective estimation of slopes that this capacity is scalar. For preparing we recognize: the preparation informational collection drawn from some dispersion, the approval set (from a similar circulation, however various information), and test informational collections with some unexpected dissemination in comparison to the preparing one. The last differentiation is significant. For the test set we ask for from dispersion information to check how well our prepared model sums up. Note that this gives a colossal scope of opportunities for the test informational collection: from small changes that will unquestionably work, up to totally various data sources that are basically ensured to fizzle. There's no highest quality level, yet test information ought to be produced with care

3.4 NAVIER STOKES

The following stage as far as intricacy is given by the Navier-Stokes conditions, which are a deep rooted model for fluids. Notwithstanding a condition for the preservation of energy (like Burgers), they incorporate a condition for the preservation of mass. This forestalls the development of shock waves, however presents another test for mathematical strategies as a hard-imperative for difference free movements.

$$u\frac{\delta u}{\delta x} + v\frac{\delta u}{\delta y} = -\frac{\delta p}{\delta x} + \frac{1}{Re}(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 v}{\delta y^2}) \tag{3.2}$$

$$u\frac{\delta v}{\delta x} + v\frac{\delta v}{\delta y} = -\frac{\delta p}{\delta y} + \frac{1}{Re}(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 v}{\delta y^2}) \tag{3.3}$$

$$\nabla.u = 0 \tag{3.4}$$

The incompressible Navier-Stokes equations with constant dynamic viscosity (eq no) is solved in this study utilising SIMPLE approach and Google's end-to-end open source ML platform Tensorflow (API version 1.15) on the Python 3.7 programming language.

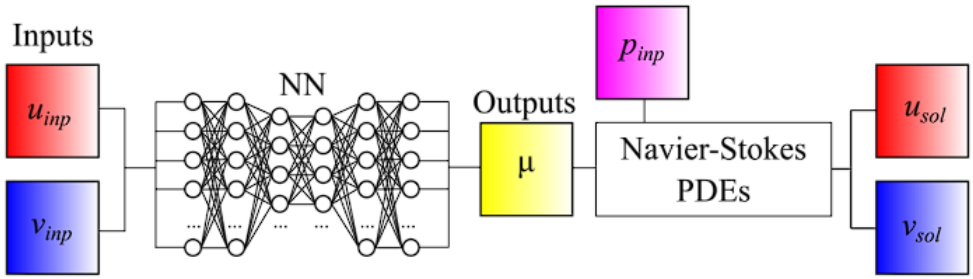


Figure 3.2: NN is embedded into the solution of the Navier-Stokes equations.

Several research groups in diverse fields have argued that solving the Poisson equation using deep learning can speed up CFD. The Poisson equation is often employed in operator-splitting approaches to discretize the Navier–Stokes equations [17], where the velocity field is first advected and the resultant field u does not fulfil the continuity equation (i.e. for incompressible flows, u does not satisfy divergence free condition). The second stage is an adjustment to guarantee that u is not divergent, resulting in the Poisson equation:

$$\frac{\nabla t}{p}\nabla^2 p = -\nabla.u \tag{3.5}$$

where t represents the simulation time step, f represents the fluid density, and p represents the pressure. The numerical solver's most computationally expensive step is usually solving this equation. As a result, developing alternative techniques to tackle it more effectively has a lot of promise. Ajuria et al. [2] suggested and tested utilising a convolutional neural network (CNN) paired with a CFD code to solve (1) in incompressible scenarios.

CHAPTER 4

MODELS AND PREDICTION

4.1 STEADY STATE 1D CONVECTION-DIFFUSION

In circumstances when there is both diffusion and convection or advection, the convection–diffusion equation explains the transport of heat, particles, or other physical quantities. Being time invariant and linear, solving the one-dimensional steady convection diffusion equation is a good place to start implementing the novel machine learning based algorithm. It is still valuable since it allows us to figure out and solve issues in unusual circumstances.
The steady state convection-diffusion equation can be written as

$$\nabla \cdot (D\nabla c) = \nabla \cdot (c \cdot u) \tag{4.1}$$

which is,

$$D\frac{d^2u}{dx^2} = \frac{d(c.u)}{dx} \tag{4.2}$$

for constant diffusivity.
To identify the relative strengths of convection and diffusion, we use the Peclet number, where

$$Pe = \frac{c\Delta x}{D} \tag{4.3}$$

higher values of Peclet number indicate a convection dominant flow and lower Peclet number for diffusion dominant flows.

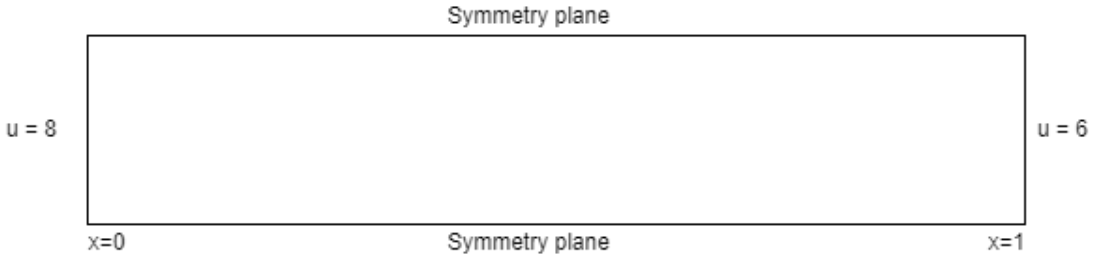


Figure 4.1: Domain of the 1-dimensional case

4.1.1 EXACT & NUMERICAL SOLUTION

Being an ODE, we have access to the exact solution for the variable of interest. Solving the ODE, we get the solution to be

$$\frac{u - u_0}{u_L - u_0} = \frac{e^{\frac{c.x}{\epsilon}} - 1}{e^{\frac{c.L}{\epsilon}} - 1} \tag{4.4}$$

As we can see from above, the exact solution can be computationally expensive with the exponential functions. In addition to implementing ML for understanding the dynamics and working, we can check if it can fix this computational power requirement to any extent. To obtain an approximate solution, the governing equations can be solved numerically by discretizing using finite difference method with the Central Differencing Scheme (CDS) as below.

$$c\frac{u_{i+1} - u_{i-1}}{2\Delta x} = D\frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} \tag{4.5}$$

The resulting set of algebraic equations are solved for using an iterative approach with calculated initial guesses for the transported property.

From observations we see the CDS scheme gets extremely unstable and faulty when made to predict for cases with Peclet number greater than 2, that is, convection dominant flow. In this section we try to fix this fallback by implementing a machine learning algorithm that maps the low fidelity solution that's obtained from the CDS scheme to high fidelity solution obtained from the exact solution.

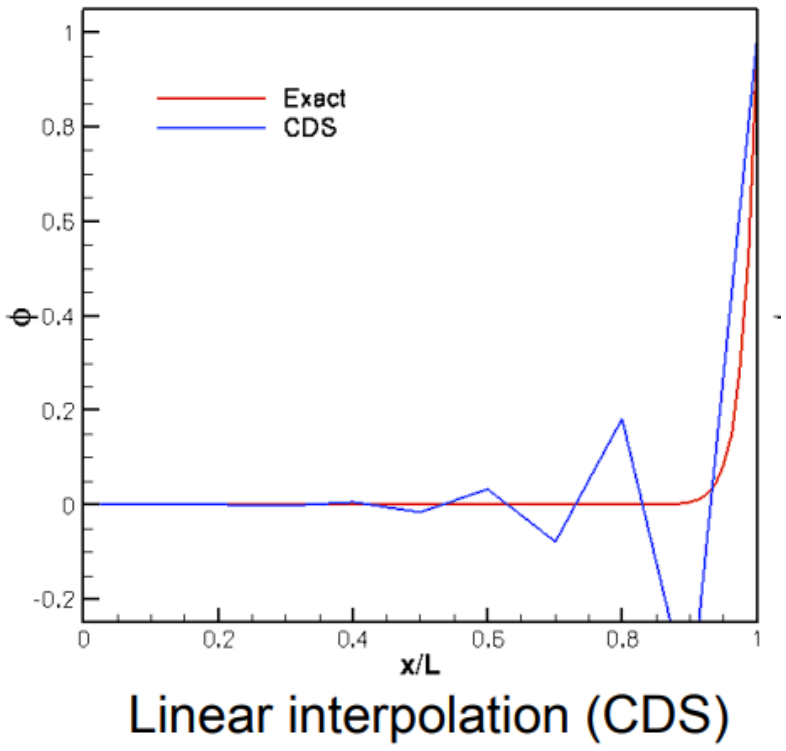


Figure 4.2: CDS graph

A function approximator for shift in weather conditions dissemination can like-

wise be formulated via preparing a solitary layered brain network with the specific arrangements or higher request precise arrangements. In the event that the portrayal for the guide is x (free factor) to u (subordinate variable), the relapse is a lot of wonderful however minor motions should have been visible overall around the specific arrangement. For high Pe cases, bigger number of cases were utilized to prepare the organization. The advantage of this organization is that it gives us a proxy capacity to the specific capacity of the overseeing differential condition. This capacity could be over and again utilized whenever it is prepared. The disadvantage of this technique is the necessity of profoundly exact answer for preparing. That implies the issue must be settled utilizing known definite capacity or utilizing higher request strategy. It isn't by any stretch of the imagination an elective guess technique in evident sense.

4.1.2 DATA-DRIVEN MODEL

The first approach was to build a data-driven approximation that maps the erroneous CDS solution to the accurate exact solution. In this approach, we train the neural network with CDS scheme data and the exact solution for a range of Peclet number cases. This trained network is then used to predict the corrected value of transported variable.

The resulting network serves as a correction mechanism that bridges the fallback inherent to the numerical scheme.

The network architecture chosen for this purpose is

- 3 hidden layers
- 32 neurons/layer
- Input: Solution from CDS scheme
- Output: Corrected field variable

The network architecture was arrived as a trade off between computational complexity associated with deeper and denser network, and accuracy of capturing detail by complex network architectures.

The mesh size of the domain is fixed to 20 in this case, ie. each training instance is a vector of 20 dimensions. In other words, each input is obtained from CDS scheme applied over the domain divided into 20 grids and similarly, each output is obtained from exact solution applied over the domain divided into 20 grids.

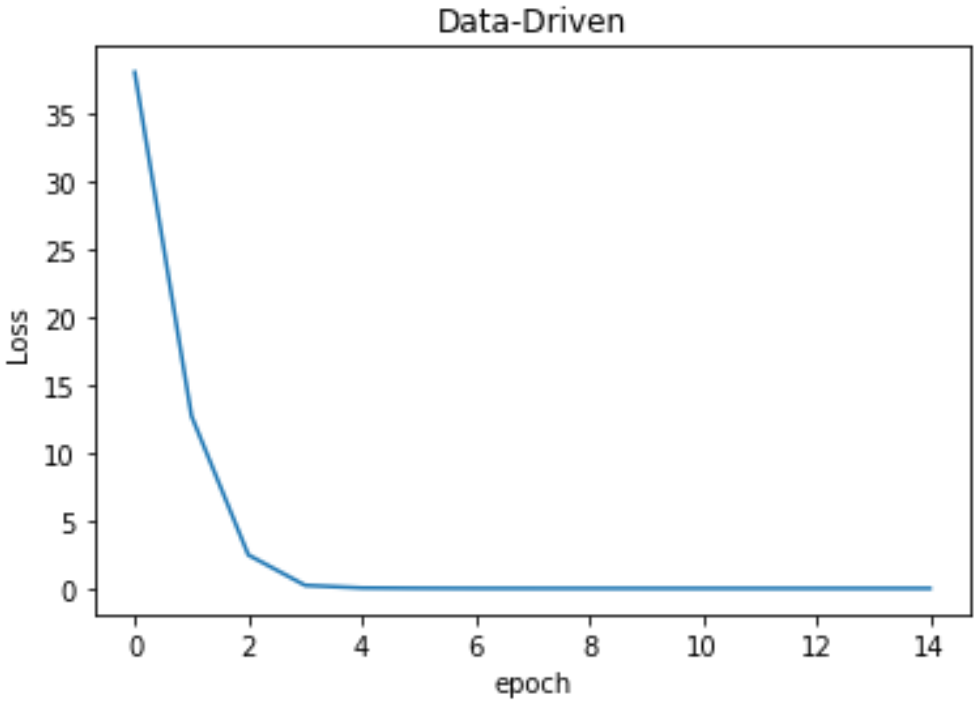
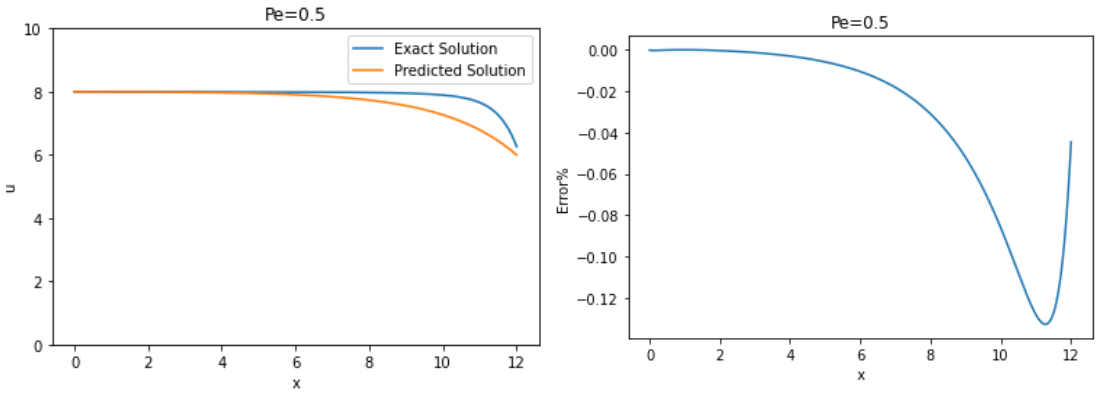


Figure 4.3: Data-driven graph

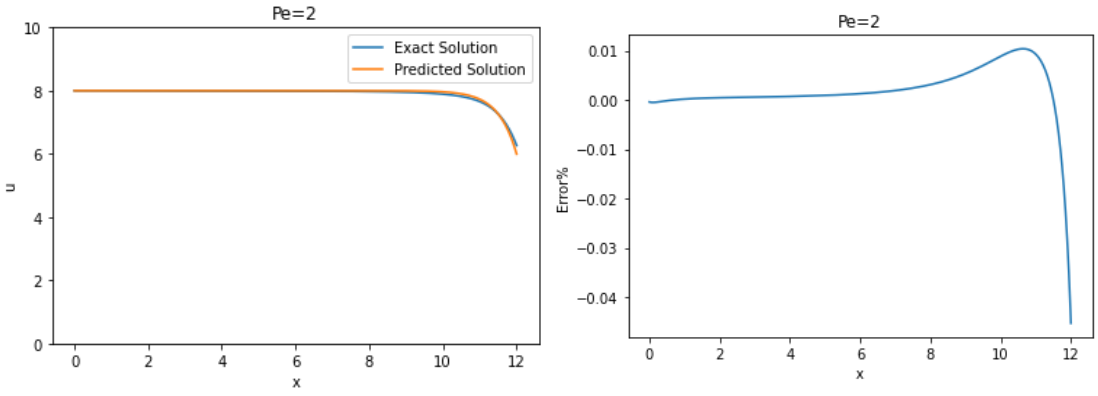
As observed from Figure 4.3 the number of iterations for the network to learn for the given case was chosen to be 14 as the overall loss got saturated beyond this point and we felt it was wastage of resources to run the training algorithm for longer.



(a) Exact vs Predicted

(b) Error percentage

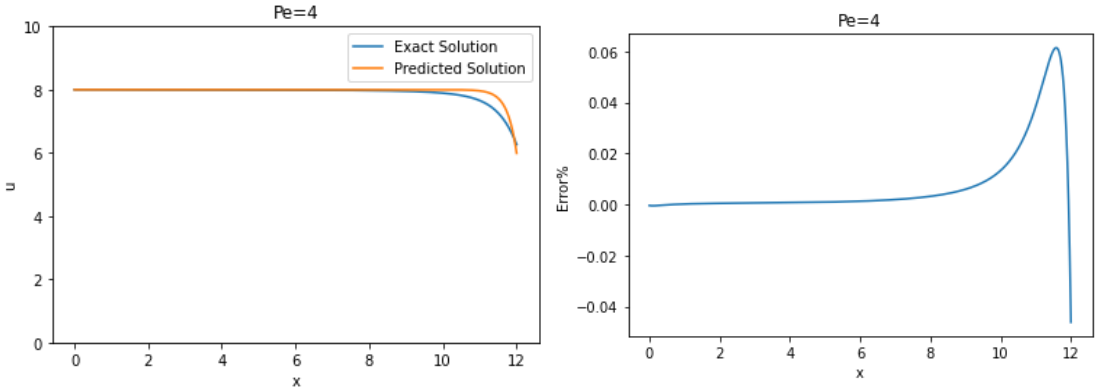
Figure 4.4: Data-Driven solution for Pe=0.5



(a) Exact vs Predicted

(b) Error percentage

Figure 4.5: Data-Driven solution for Pe=2



(a) Exact vs Predicted

(b) Error percentage

Figure 4.6: Data-Driven solution for Pe=4

Data-driven methods are helpful for arrangement improvement, work guess and mistake assessment. Almost certainly, these are solid capacities yet age of information expects us to utilize traditional strategies. The extension for direct decrease in computational cost is absurd assuming a strategy requires a lot of information to prepare upon. For an other estimation strategy, it is crucial for cut off the utilization of precise information. There is a requirement for looking for techniques that don't depend much on accessible arrangement. Along these lines, the physical science informed learning comes into picture.

4.1.3 PINN MODEL

Network Architecture

- 64 neuron/layer
- 7 hidden layers
- Input: X - coordinate

- Output: Exact solution

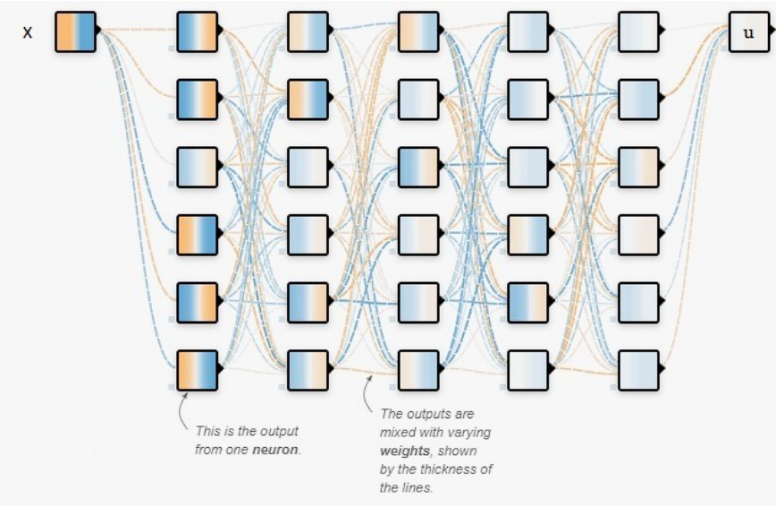


Figure 4.7: Represented PINN architecture for 1D Convection-Diffusion

The architecture was arrived at by means of trial and error performed on a wise choice of combinations. Neural networks are observed to perform better when the neurons per layer are in exponents of two. Considering this and the computation time taken for huge networks, our domain for trial was fixed as seen in the table below. Based on the results we chose the network architecture as described above.

Neurons/layer \ Hidden Layers	1	2	3	4	5	6	7	8	9
8	0.524258	0.359907	0.283909	0.191910	0.163470	0.288406	0.155962	0.208045	0.300889
16	0.504164	0.364898	0.256840	0.162470	0.167124	0.151242	0.166797	0.200214	0.220921
32	0.579731	0.353373	0.325204	0.201689	0.155867	0.165799	0.154105	0.354741	0.164206
64	0.444545	0.357300	0.340547	0.198566	0.161373	0.150040	0.090564	0.157436	0.225263
128	0.577974	0.363626	0.280424	0.207207	0.148019	0.137188	0.147777	0.167458	0.201977

Figure 4.8: Number of Neurons Vs Layers

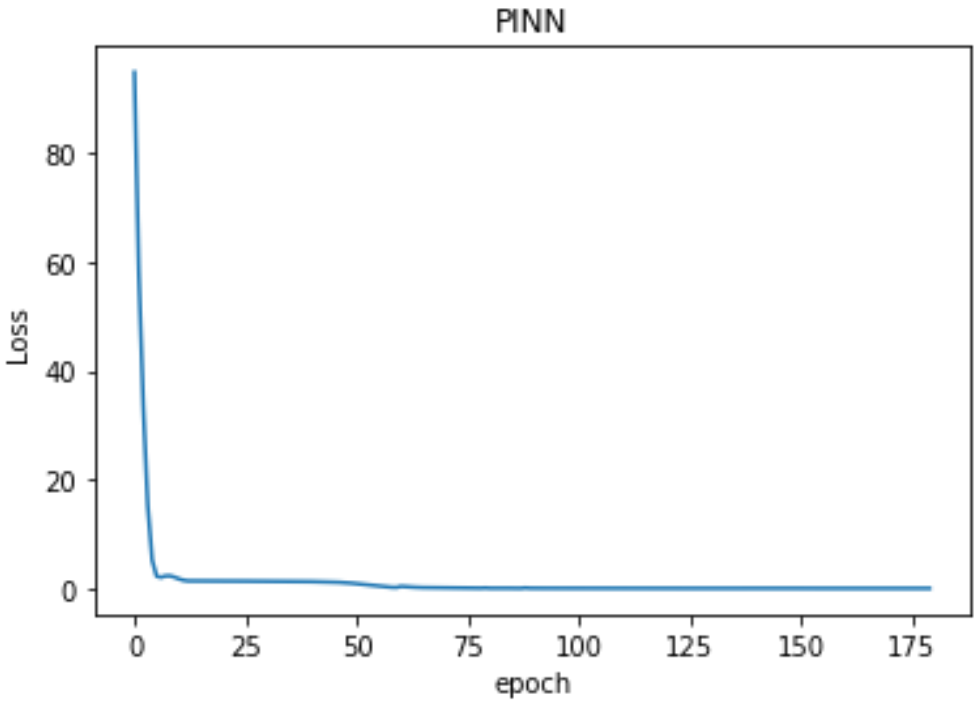


Figure 4.9: PINN graph

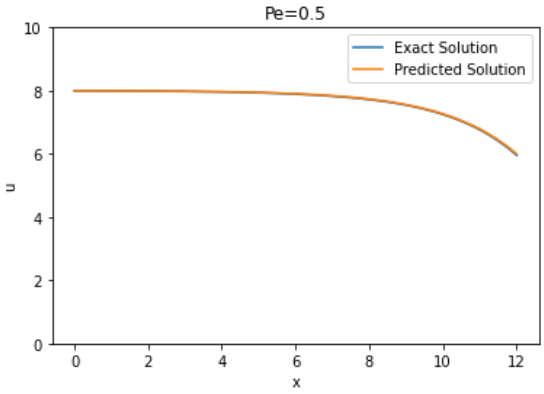
A custom weighted loss function was developed to regularize the neural network to look for prediction in a space that satisfies the physics constraints prevalent. Our loss function includes

- **boundary loss**, which penalises the network heavily if the boundary values don't match the constraints given for the domain.
- **functional losses**, which is used to transfer our knowledge of the field variables to the network.
- **governing equation loss**, which checks if the predicted field variables satisfy the governing equation.

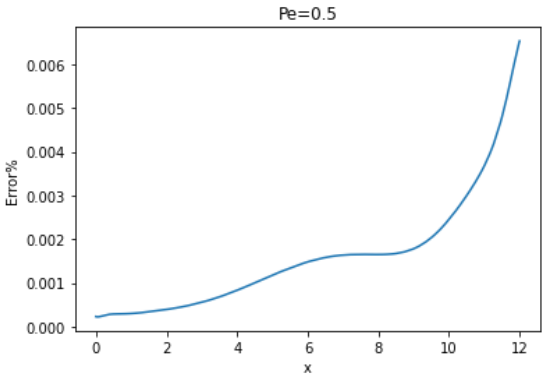
With necessary weights we can force the neural network to be stringent on essential conditions and relaxing slightly on other criteria to speed up the training and predicting time.

With the data from exact solution, we were able to approximate a function that maps the spatial co-ordinates directly to the exact solution in such a way that the approximated function satisfies the governing equation.

This section contains the predictions from the neural network trained for 1-dimensional cases.

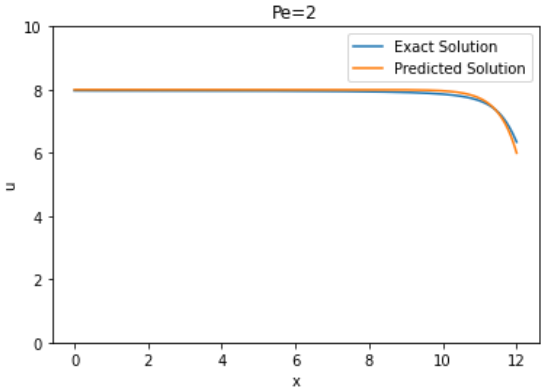


(a) Exact vs Predicted

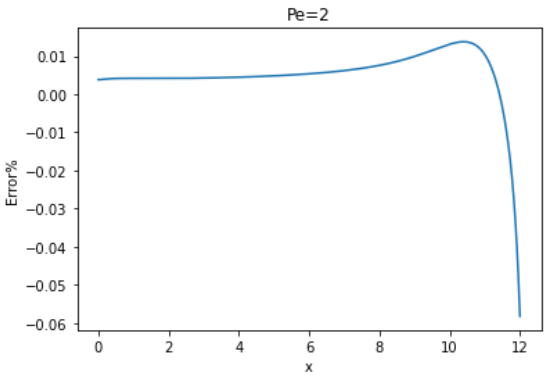


(b) Error percentage

Figure 4.10: PINN solution for Pe=0.5

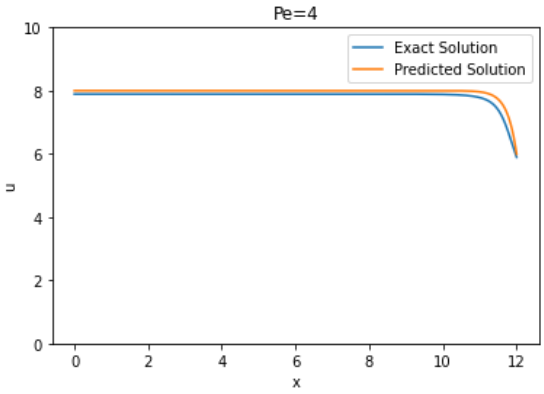


(a) Exact vs Predicted

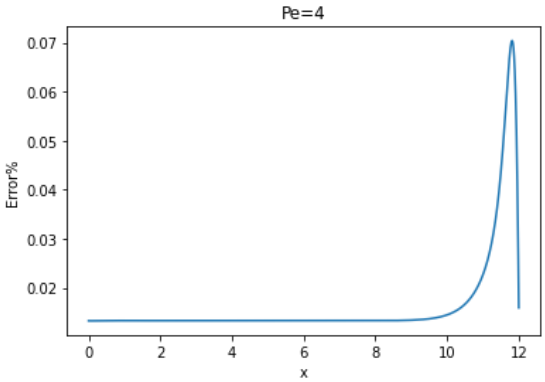


(b) Error percentage

Figure 4.11: PINN solution for Pe=2



(a) Exact vs Predicted



(b) Error percentage

Figure 4.12: PINN solution for Pe=4

From the results we can observe that the predicted field of the transported variable very closely agrees with the exact solution that we obtained by solving the ODE of interest.

	Data-Driven	PINN
Training data size	500*10	1*10
Data preparation time	9s	<1s
Epochs	15	180
Final error	0.1405	0.0309
Training Time	8s	14s

Table 4.1: Data-Driven vs PINN

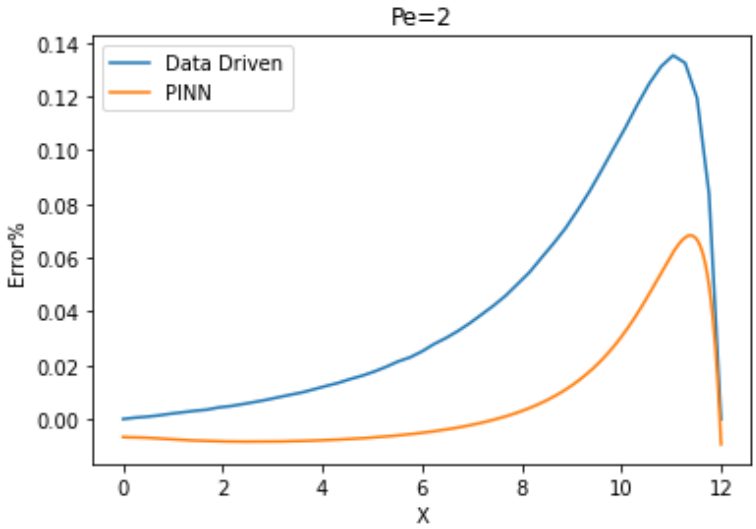


Figure 4.13: Data-Driven Vs PINN prediction for Pe=2

Figure 4.18 demonstrates how effective the data-driven model and PINN model are in predicting the transport variable field. Table 4.1 quantifies how much we save in terms of computational power and time and still get lower lower mean square loss by just including the physical constraints to the neural network. Although the improvement seems smaller, it will demonstrate significant enhancement when extended to higher dimensional cases. So, this is a positive sign and indicates we are progressing in the right direction with this approach.

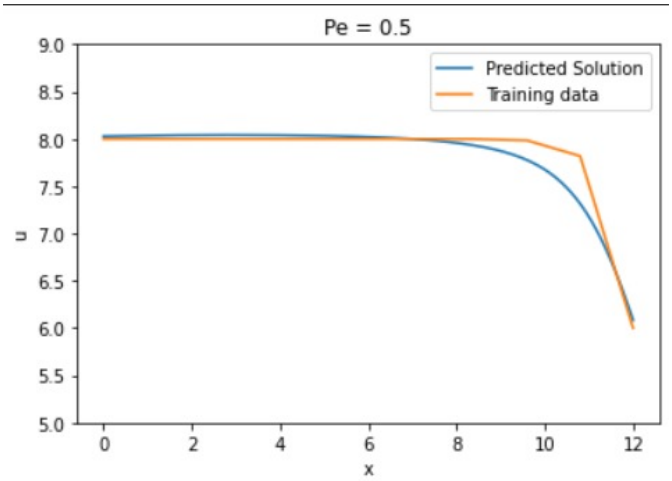


Figure 4.14: Turning low fidelity data to high fidelity data for Pe=0.5 ($L_2 error : 0.024$)

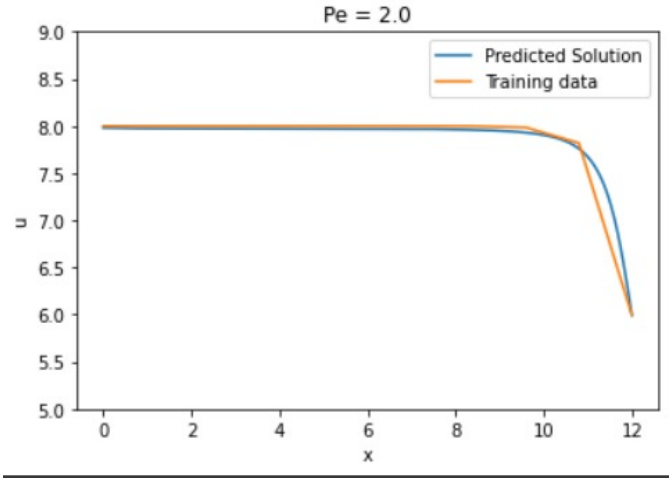


Figure 4.15: Turning low fidelity data to high fidelity data for Pe=2 ($L_2 error : 0.016$)

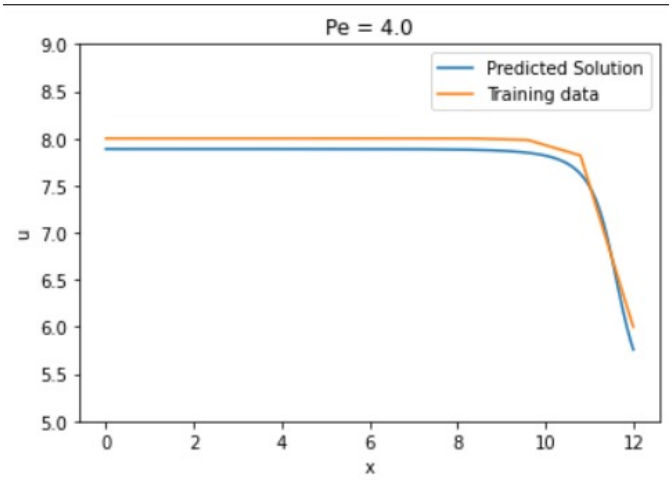


Figure 4.16: Turning low fidelity data to high fidelity data for Pe=4 ($L_2 error : 0.023$)

4.2 2D CONVECTION-DIFFUSION EQUATION

The generalized convection diffusion is extended to a 2-dimensional case. So we extended the governing equation to include both the space variables. The governing equation takes the form,

$$K(\frac{\delta^2\phi}{\delta x^2} + \frac{\delta^2\phi}{\delta y^2}) = u\frac{\delta\phi}{\delta x} + v\frac{\delta\phi}{\delta y} \tag{4.6}$$

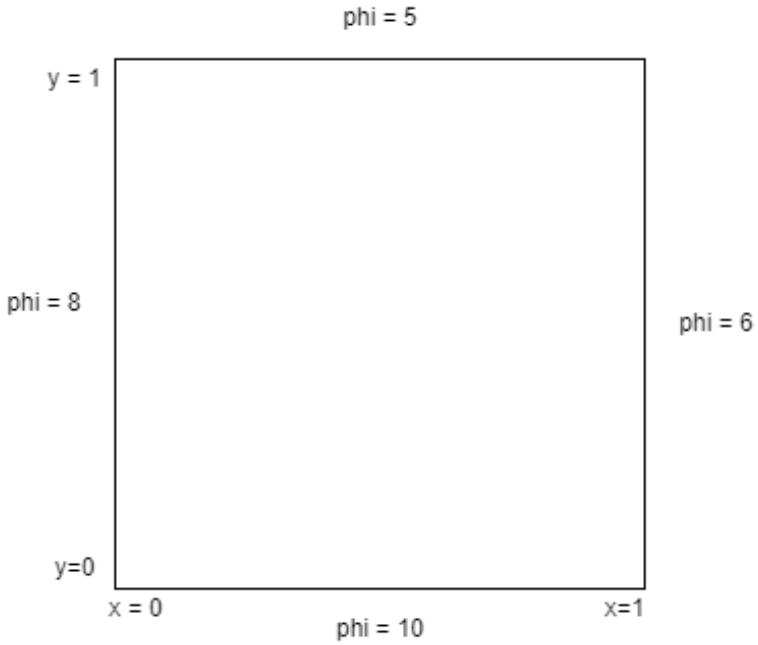


Figure 4.17: Domain for 2D case

4.2.1 NUMERICAL SOLUTION

To generate the training data required to train and compare the predictions of the model, we get flow field information by solving the governing equations using numerical techniques.

Hybrid Scheme is a hybrid of central difference scheme and the upwind scheme that takes the best parts of both schemes to avoid the instability associated with CDS and eliminate the excessive convection effect associated with the upwind scheme.

$$\phi_{hybrid} = \begin{cases} \phi_{CDS}, Pe < 2 \\ \phi_{Upwind}, Pe \geq 2 \end{cases}$$

4.2.2 PINN MODEL

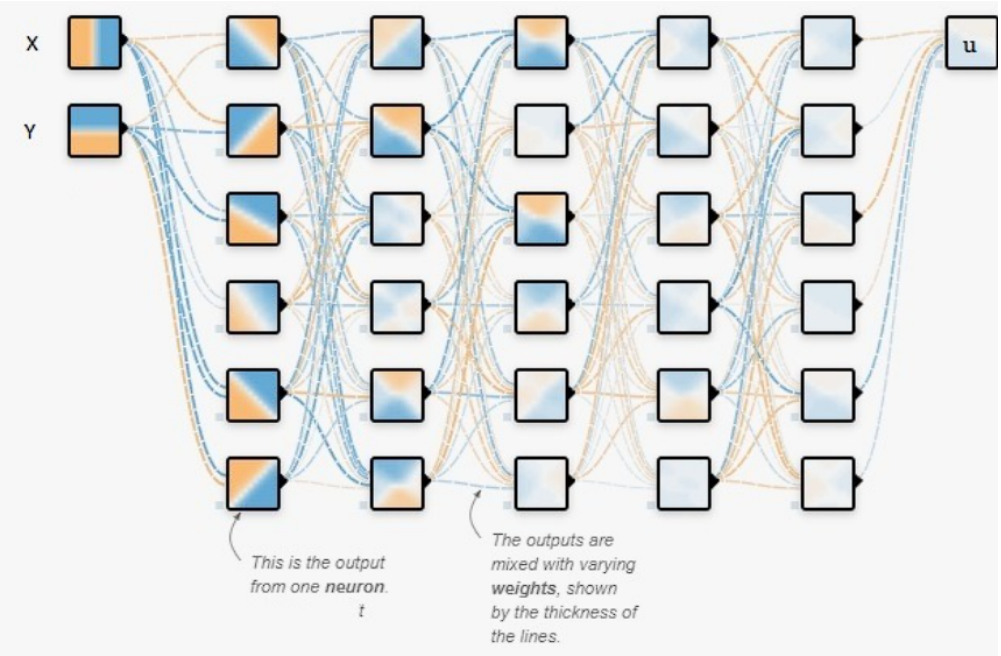
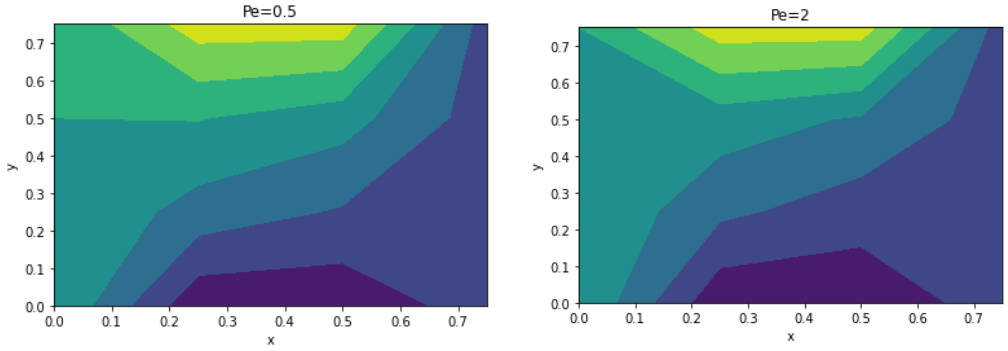


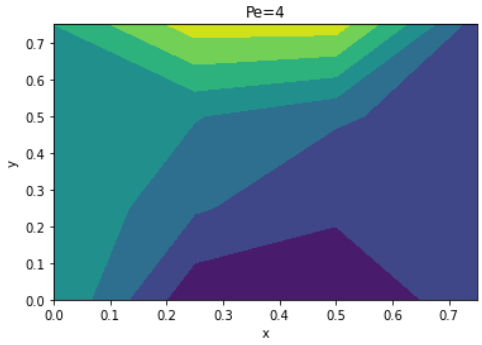
Figure 4.18: Represented PINN architecture for 2D Convection-Diffusion

4.2.3 RESULTS AND DISCUSSION



(a) Pe = 0.5

(b) Pe = 2

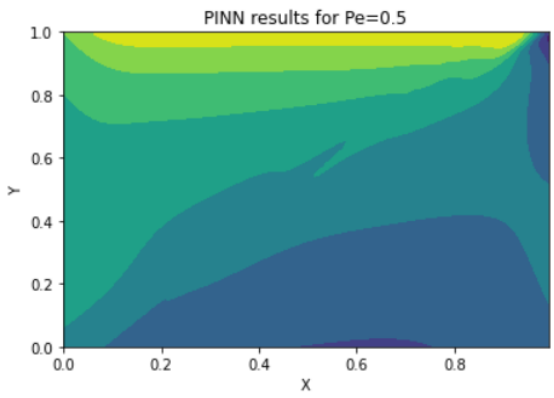


(c) Pe = 4

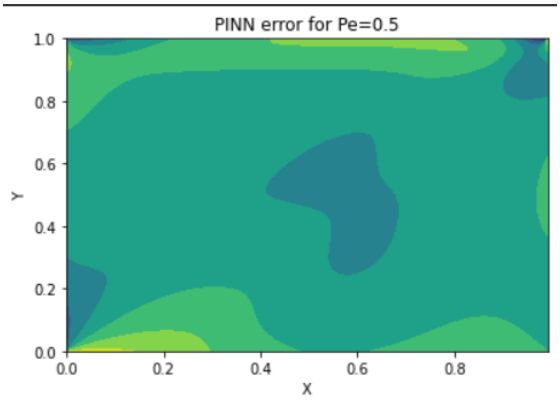
Figure 4.19: Training velocity field

Grid size

- Training Data - 20x20
- Output Data - 140x140

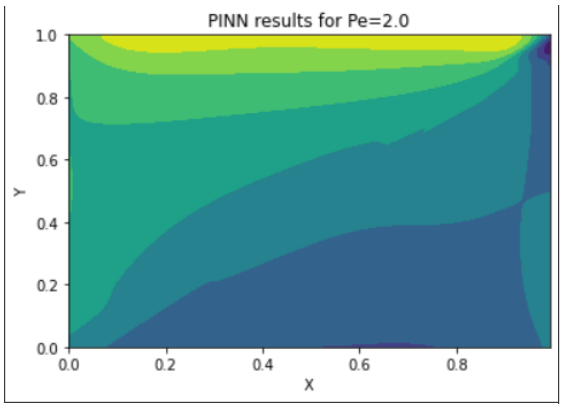


(a) Predicted velocity field for Pe = 0.5

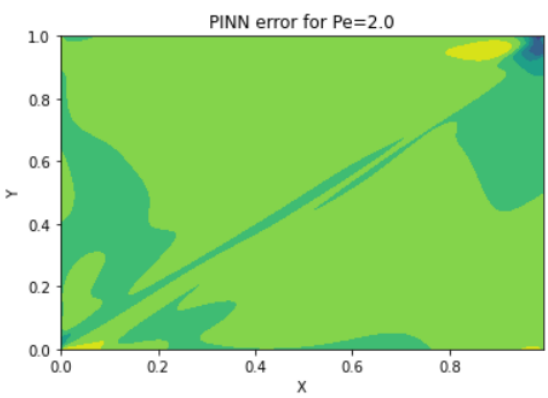


(b) Error in prediction for Pe = 0.5

Figure 4.20: Prediction for Pe=0.5

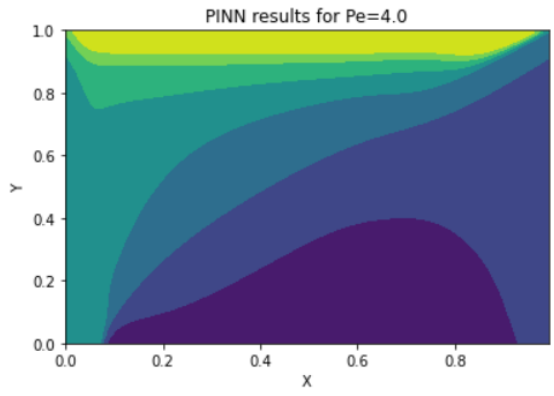


(a) Predicted velocity field for Pe = 2

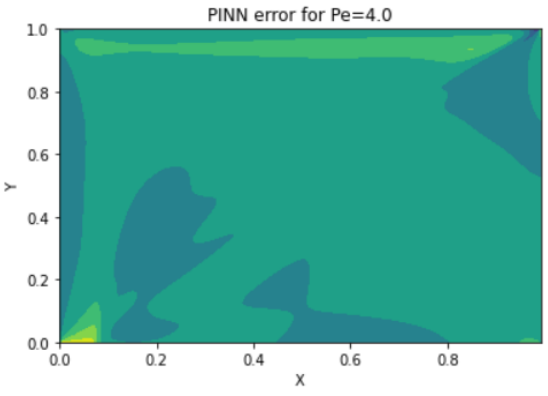


(b) Error in prediction for Pe = 2

Figure 4.21: Prediction for Pe = 2



(a) Predicted velocity field for Pe = 4



(b) Error in prediction for Pe = 4

Figure 4.22: Prediction for Pe = 4

4.3 NAVIER-STOKES EQUATION

We extend the approach to the fluid flow problem next. For simplicity, we chose the traditional lid-driven cavity problem, which involves a fluid exhibiting rotational flow as a result of being enclosed by three fixed walls and one movable wall as represented in Figure 4.23. The flow field variables here are the velocity components and pressure field, which are conventionally obtained by solving the Navier-Stokes equation and continuity equation.

Governing Equations

- Continuity eqn.

$$\frac{\delta u}{\delta x} + \frac{\delta v}{\delta y} = 0 \tag{4.7}$$

- x-momentum eqn,

$$u\frac{\delta u}{\delta x} + v\frac{\delta u}{\delta y} = -\frac{\delta p}{\delta x} + \frac{1}{Re}(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 v}{\delta y^2}) \tag{4.8}$$

- y-momentum eqn.

$$u\frac{\delta v}{\delta x} + v\frac{\delta v}{\delta y} = -\frac{\delta p}{\delta y} + \frac{1}{Re}(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 v}{\delta y^2}) \tag{4.9}$$

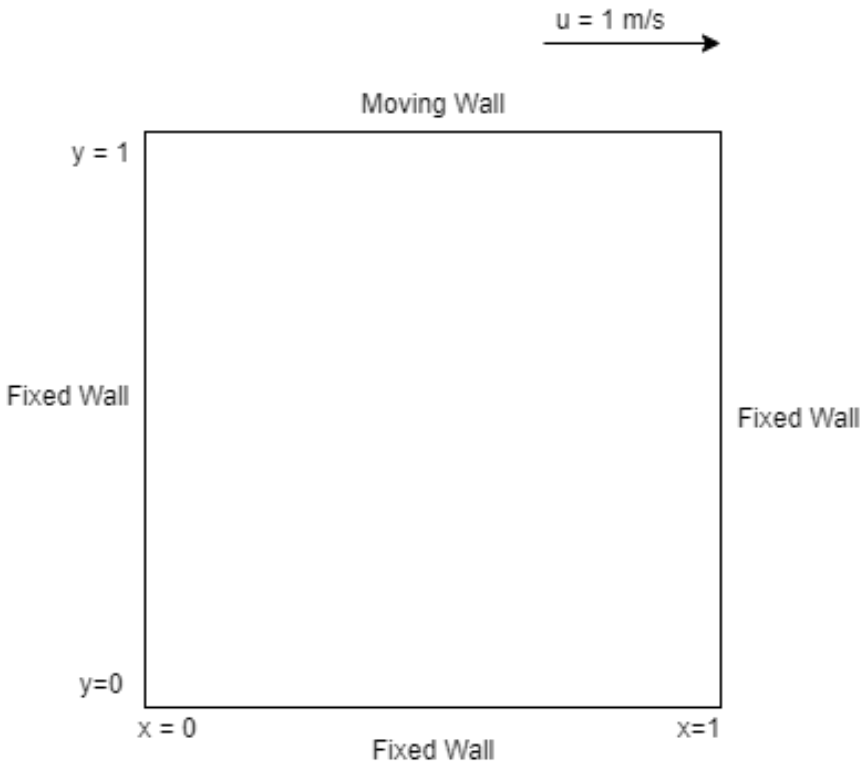


Figure 4.23: Domain of Lid-Driven Cavity

Boundary Conditions

- Velocity
  - Fixed Wall - No slip (u, v = 0)
  - Moving Wall - No slip (u = 0.5 m/s, v = 0)
- Pressure
  - Fixed Wall - $\frac{\delta p}{\delta \hat{n}} = 0$
  - Moving Wall - $p_x = \frac{1}{2}\rho v^2 = 0.125Pa, \frac{\delta p}{\delta y} = 0$

4.3.1 NUMERICAL SOLUTION

Similar to our 2D convection-diffusion case, Navier-Stokes equation cannot be solved for exact solution. Hence, we need to numerically approximate the solution. Fluid flow problems come with an additional difficulty inherent to their non-linear pressure-velocity coupled governing equations.

There exist several algorithms to solve this of which Semi-Implicit Method for Pressure Linked Equations **(SIMPLE)** algorithm is the most straightforward and yields close enough solution. The major steps of the algorithm are depicted in the flowchart in Figure 4.24.

The major steps involve guessing the initial values of the field variables and solving the momentum equations and pressure correction equation derived from the Poisson's equation which helps us correct the guessed values of the field variables. The corrected variables are checked for convergence in governing equations and are iteratively updated until convergence of residuals.

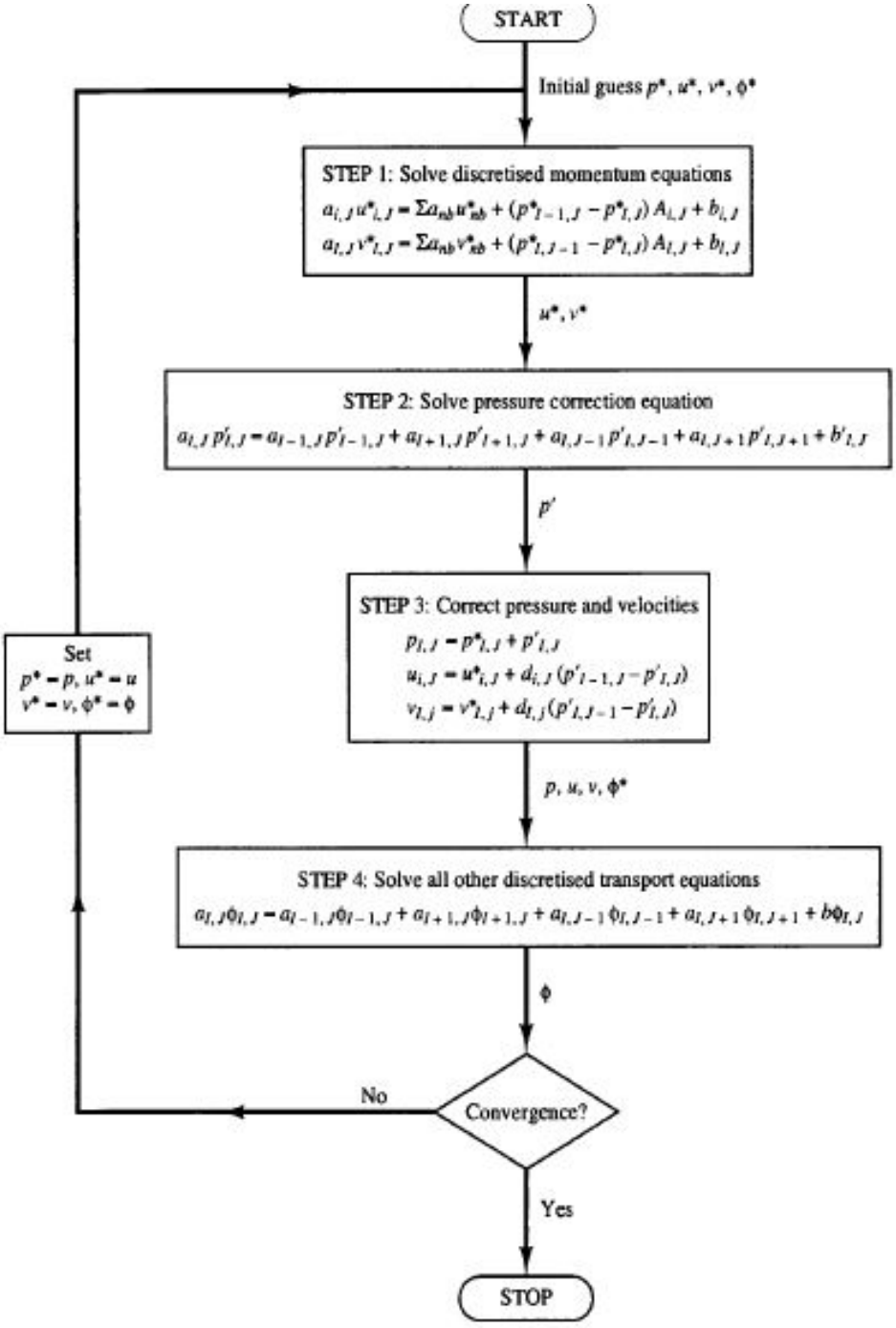


Figure 4.24: Simple Algorithm

With the code written for solving for flow field and pressure field using SIMPLE algorithm for a structured grid, we extract the training data and test data using different resolution of grid.

4.3.2 PINN MODEL

Our objective here is to predict the flow field variables using neural network that obeys physical laws governing the output variable. the network is designed to convert low-fidelity, low-resolution data to high-resolution, high-fidelity data without much computational power requirement. For this purpose, we develop a network that gets the spatial coordinates for input and outputs the flow field values at that position in space, having been trained with low-resolution flow field values. While doing all this, the network's loss function is designed to penalise any abnormal predictions that do not obey the governing equations.

With this we can understand that the low-resolution data is fed into the network in the form of training and high-resolution flow field is obtained by using the network to predict the field variables for various position in the domain.

We found the network architecture in an approach similar to that explained in previous sections. The final architecture obtained had

- 5 hidden layers
- 64 neurons/layer
- Activation function - tanh
- Optimizer - Adam with Learning rate of 0.005
- Input: Spatial position
- Output: Field variables

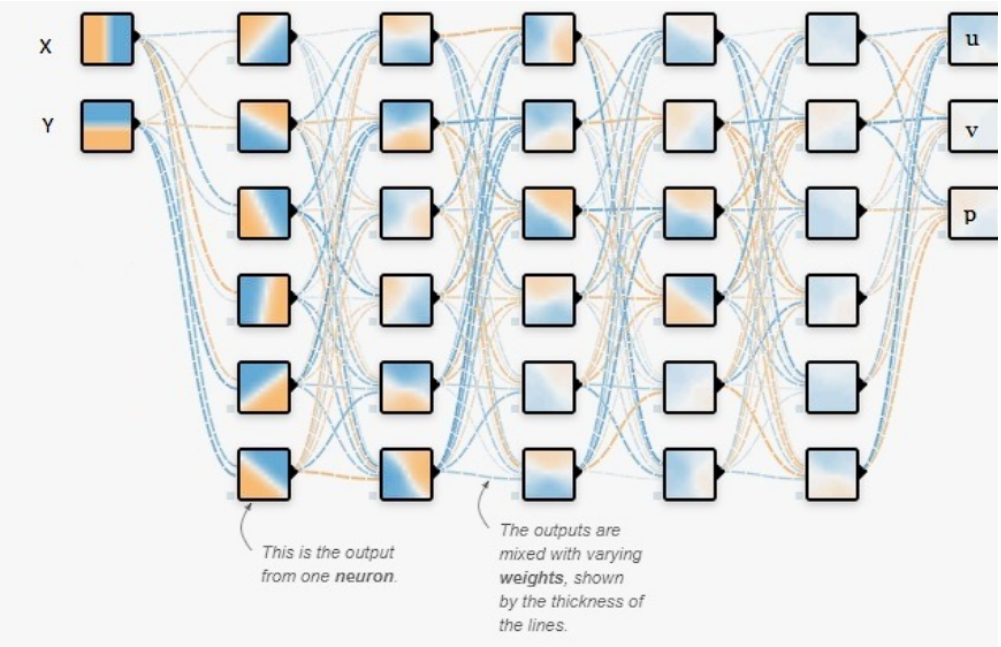


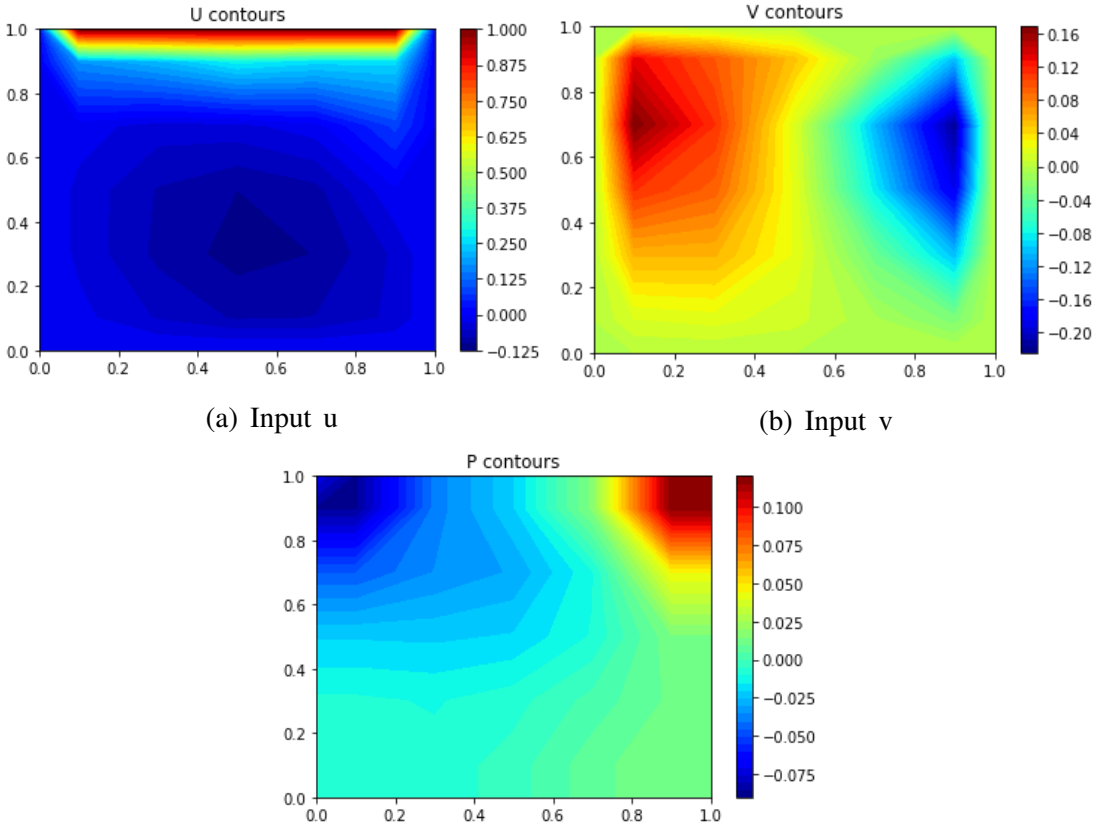
Figure 4.25: Represented PINN architecture for Fluid Flow

A schematic of the neural network is illustrated in Figure 4.25. It should be noted that the figure isn't an exact depiction of our model, but is a schematic

representation only. The reason being, our network has 64 neurons per layer which would be highly difficult to interpret. Hence, we used a simpler image with just 6 neurons per layer for the readers to get a better insight into our model.

The custom loss function imposes the physics onto the model through a regularization technique which narrows down the choices of the networks guesses to a limited set of values which satisfy the physical laws involved. If the choices for the network's prediction does not agree with the physics taught to the networks, the network is heavily penalised, hence, the network tries to drive down the loss by looking for values that almost satisfy the governing laws.

4.3.3 RESULTS AND DISCUSSION



(a) Input u

(b) Input v

(c) Input Pressure contour

Figure 4.26: Training velocity & pressure field

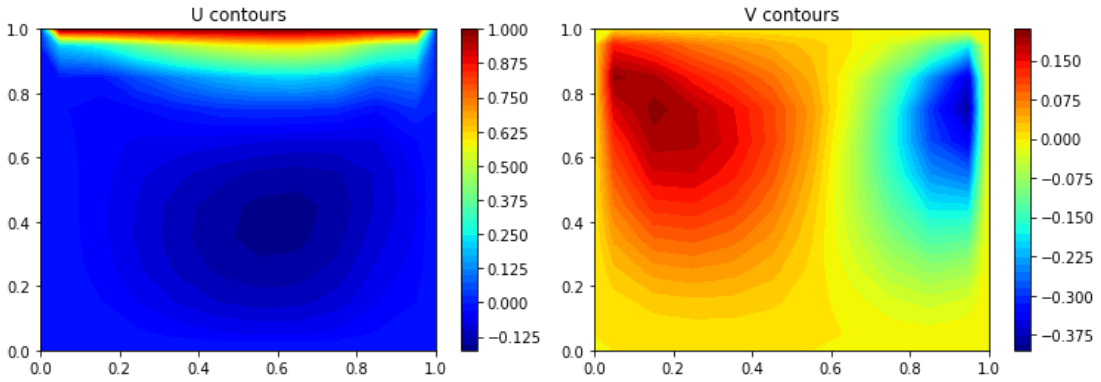
The coarse grid flow field variables are used to train the neural network and help us predict the flow field for a very fine grid with high resolution.

Grid size

- Training Data - 5x5
- Output Data - 10x10; 50x50

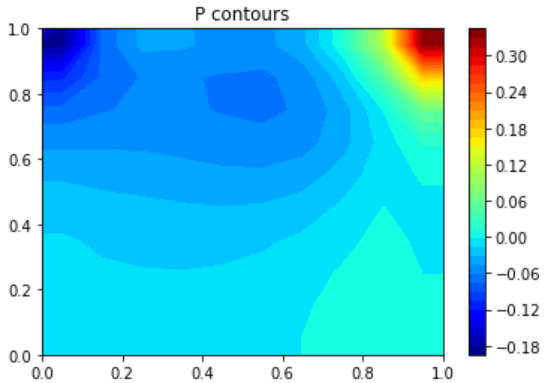
From the input training data space, we can see that the quality of the distribution is very low, that is, the resolution of the contours are poor. Although this isn't a quantifiable parameter, we can observe that in the edges of contours we are able to see mesh interfaces. this signifies the grid is extremely coarse for it to effectively capture the field variables. The only way to resolve this is by taking a finer mesh. The issue with finer mesh again is that the computational demand increases which is always unfavourable. So, we employ neural networks to undercut this situation.

Here we use the coarse mesh training data, to train the neural network to approximate a function that satisfies the Navier Stokes equation. Predictions were then done on finer meshes. This saves a huge amount of time and computational energy that would have been required had one attempted to produce the finer mesh results from the numerical solutions.



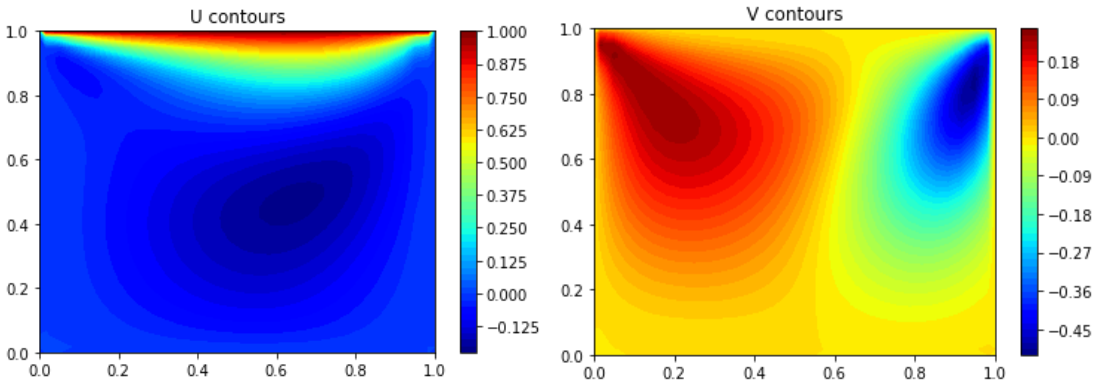
(a) Output u - Fine

(b) Output v - Fine



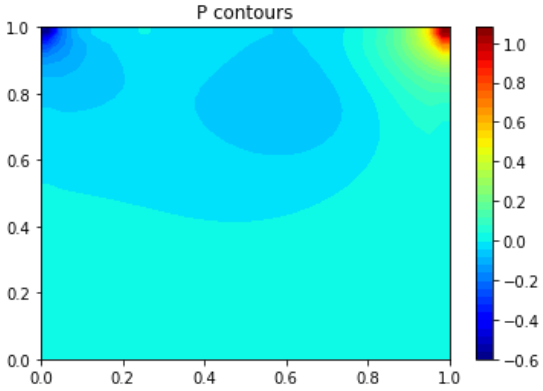
(c) Output Pressure contour - Fine

Figure 4.27: Output velocity & pressure fields - Fine



(a) Output u - Finer

(b) Output v - Finer



(c) Output Pressure contour - Finer

Figure 4.28: Training velocity & pressure fields - Finer

We used our neural network to output a slightly finer grid. The resulting flow field variable contours as seen in Figure 4.27 still suffer from pixelation although, the quality of contour had evidently improved. We observed that our neural network was able to smoothen the contours and predict without any difficulty for slightly (2X) finer grid itself. This shows that our approach is right and we proceed to progress in the same manner.

We then tried extending it to see how much the more of a refined, high-resolution flow field we can obtain from the very coarse set of data points we initially had.

Figure 4.28 shows a 50x50 solution space which is five times finer than our input to neural network. This resulted with much more distinct contours with finer boundaries without any visible pixelation. Increasing beyond this didn't affect the nature of contours. This indicates a high-resolution, high-fidelity solution is obtained.

Hence, for the tested case, the neural network was comfortably able to capture the details for a grid size with five times more points on the sampling space.

CHAPTER 5

SUMMARY

5.1 CONCLUSION

We were able to successfully implement and demonstrate that computational fluid dynamics can be accelerated or enhanced by the use of deep learning models, for at least the steady state 1D and 2D Convection-Diffusion case and 2D fluid flow case. The approach of predictively modeling the variable of interest by not just a blind curve fitting mechanism, instead couppled with a regularization technique which has accelerated the process by forcing the neural network search a desirable subset of options rather than an open-ended prediction. This regularization technique has been refered to as physics-informed neural networks (PINN) throughout this literature. It has been showed that physics constraints to the neural networks have accelerated the process of predicting solution. It has also been shown that the neural networks are able to bridge the gap in knowledge as seen when very coarse mesh data points are used to avail a very high-resolution, high-fidelity flow fields.

5.2 APPLICATIONS

Our work finds its use in all generic flow problems. We made sure to adopt an open approach when dealing with flow problem, such as, the convection diffusion case where we proceeded without any specific property being transported. Any new governing law can be easily adopted into the model by slight tweaks in the segment involving loss functions.

We adopted a very general approach as this work can be developed upon by researchers working on any field of CFD. Also, we were determined to arive at a model that would significantly reduce the limitations of CFD which was plaguing our research in various ends. We hope, with a basic framework laid on this technique, we can research and develop further to achieve our goal of being computational less demanding.

The major and most straight-forward application of this current approach is converting low-fidelity, low-resolution data to high-fidelity, high -resolution data.

5.3 FUTURE SCOPE

Although we were able to observe positive results for the cases studied in this occasion, we must extend this approach to the other slightly different cases, such as, predicting the unsteasdy state variables by knowledge of prior field variables, extending PINN for unstructured grid. And as always, any deep learning problem has scope for improvement of the entire scheme itself. From what we can observe, the inclusion of regularization techniques made the processes much more efficient than any other variables. Hence, focus must be on developing better regularization techniques if there is an opportunity to develop a faster computing technique.

APPENDIX A

CODE ATTACHMENTS

A.1 LOSS FUNCTION

```
def PINN_GDE_Loss(train_phi,phi_pred):
  return gde_loss() + function_loss() + boundary_loss(a,b)

def function_loss(train_phi,phi_pred):
  return tf.reduce_mean(tf.square(train_phi - phi_pred))

def boundary_loss(a, b):
  phi_bc_pred = modelPINN(bc_input)
  return tf.reduce_mean(tf.square(phi_bc - phi_bc_pred))

def gde_loss():
  with tf.GradientTape() as tape2:
    tape2.watch(col_input)
    with tf.GradientTape() as tape:
      tape.watch(col_input)
      col_phi_pred= modelPINN(col_input)
      phi_x=tape.gradient(col_phi_pred, col_input)
    phi_xx=tape2.gradient(phi_x, col_input)

  residual = tf.abs(rho*(u*phi_x[0]+v*phi_x[1]) - k*(phi_xx[0]+phi_xx
      [1]))
  return tf.reduce_mean(residual)
```

REFERENCES

1. Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. “Multilayer Feedforward Networks Are Universal Approximators.” Neural networks 2.5 (1989): 359–366. Web.

2. Aurelien Geron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O’Reilly Media, Inc, 2019.

3. Baydin, Atilim Gunes et al. “Automatic Differentiation in Machine Learning: a Survey.” (2015).

4. Zienkiewicz, O. C., R. L. Taylor, and J. Z. Zhu. The Finite Element Method: Its Basis and Fundamentals. Seventh edition. Oxford, UK: Butterworth-Heinemann, 2013.

5. LeVeque, Randall J. Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems. Society for Industrial and Applied Mathematics, 2007.

6. Lin, Henry W, Max Tegmark, and David Rolnick. “Why Does Deep and Cheap Learning Work So Well?” Journal of statistical physics 168.6 (2017): 1223–1247.

7. Brunton S.L., Noack B.R., Koumoutsakos P. Machine learning for fluid mechanics // Annual Review of Fluid Mechanics 52, 477-508, 2020 https://doi.org/10.1146/annurev-fluid-010719-060214

8. Dinh N., Nourgaliev R., Bui A., Lee H. Perspectives of nuclear reactor thermal hydraulics Proceedings of the NURETH-15, Pisa, Italy, May 12-17, 2013

9. Lewis A., Smith R., Williams B., Figueroa V. An information theoretic approach to use high fidelity codes to calibrate low-fidelity codes // Journal of Computational Physics 324, 24-43, 2016 http://dx.doi.org/10.1016/j.jcp.2016.08.001

10. M. Raissi, G. E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, Journal of Computational Physics 357 (2018) 125–141.

11. M. Milano, P. Koumoutsakos, Neural network modeling for near wall turbulent flow, Journal of Computational Physics 182 (2002) 1–26.

12. T. Nguyen, B. Pham, T. T. Nguyen and B. T. Nguyen, ”A deep learning approach for solving Poisson’s equations,” 2020 12th International Conference on Knowledge and Systems Engineering (KSE), 2020, pp. 213-218, doi: 10.1109/KSE50997.2020.9287419.

13. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

14. Andreas Louskos, Physics-Informed Neural Networks and Option Pricing

15. Chengping Raoa Hao Sunb,c, Yang Liua, Physics-informed deep learning for incompressible laminar flows

16. M. Raissi, Z. Wang, M. S. Triantafyllou, G. E. Karniadakis, Deep learning of vortex-induced vibrations, Journal of Fluid Mechanics 861 (2019) 119– 137.

17. Arsen S. Iskhakov, Nam T. Dinh, Physics-integrated machine learning: embedding a neural network in the Navier-Stokes equations. Part I, https://doi.org/10.48550/arXiv.2008.10509

18. Siddharth Rout, Vikas Dwivedi, Balaji Srinivasan, Numerical Approximation in CFD Problems Using Physics Informed Machine Learning, arXiv:2111.02987

19. G. E. Karniadakis, M. Israeli, S. A. Orszag, High-order splitting methods for the incompressible navier-stokes equations, Journal of Computational Physics 97 (1991) 414–443

20. S. H. Rudy, S. L. Brunton, J. L. Proctor, J. N. Kutz, Data-driven discovery of partial differential equations, Science Advances 3 (2017).

21. R. Tripathy, I. Bilionis, Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification, arXiv preprint arXiv:1802.00850 (2018).

22. M. Milano, P. Koumoutsakos, Neural network modeling for near wall turbulent flow, Journal of Computational Physics 182 (2002) 1–26.

23. I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks 9 (1998) 987–1000

24. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707.

25. S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics 52 (2020) 477–508.