Application of the Physics-Informed Neural Network in the domain of Chemical Engineering

Project report to be submitted

in partial fulfilment

of the requirements for the award of the degree of

Bachelor of Technology in Chemical Engineering

by

Raja Babu

21JE0740

Under the supervision of

Prof. Sourav Sengupta

Department of Chemical Engineering

Indian Institute of Technology (ISM) Dhanbad

Prof. Sourav Sengupta
Assistant Professor
Department of Chemical Engineering
Indian Institute of Technology (ISM) Dhanbad, INDIA

CERTIFICATE BY THE SUPERVISOR

This is to certify that the project titled “Application of the Physics-Informed Neural Network in the domain of Chemical Engineering” is a Bonafide record of the work carried out by Raja Babu (Adm. No. 21JE0740) under my supervision and guidance in partial fulfilment of the requirements for the degree of Bachelor of Technology in Chemical Engineering during the academic session 2024-2025 in the Department of Chemical Engineering at the Indian Institute of Technology (ISM) Dhanbad.

Date:

(Prof. Sourav Sengupta)
Supervisor

Phone: +91-326-2235329 (Office) *E-mail: souravsengupta@ iitism.ac.in*

DECLARATION

I, **Raja Babu**, certify that:

- This project report represents my own original work, conducted under the guidance of my supervisor.
- The content has not been submitted to any other institution or university for any degree, diploma, or other academic recognition.
- I have strictly adhered to the ethical standards and guidelines prescribed by the institute throughout the development of this work.
- In instances where external sources of information (including data, theoretical analysis, figures, and all content) were referenced, I have appropriately credited these sources in the and listed them in the references section.

_____

Raja Babu
21JE0740

ACKNOWLEDGEMENT

I would like to extend my sincere gratitude to everyone who has supported the completion of my B.Tech final year project, titled “Application of the Physics-Informed Neural Network in the Domain of Chemical Engineering.”

First, I am profoundly grateful to my project guide, Prof. Sourav Sengupta, for their invaluable support, guidance, and mentorship. Their deep expertise, encouragement, and insightful feedback have been fundamental in shaping the direction and scope of this research. I am particularly thankful for their patience and belief in my abilities, which were essential in overcoming the challenges encountered along the way.

I would also like to express my deep appreciation to the faculty members of the Chemical Engineering Department. Their academic guidance, helpful insights, and consistent encouragement throughout my undergraduate studies have been instrumental in my academic and professional growth.

I am thankful to the Indian Institute of Technology (Indian School of Mines), Dhanbad, for providing the resources, facilities, and infrastructure necessary for conducting this project. The institute’s commitment to fostering research and innovation created a supportive environment for my work.

Finally, I am profoundly grateful to my family and friends for their unwavering support and encouragement throughout this journey. Their constant belief in my potential has been a tremendous source of motivation, enabling me to pursue my academic aspirations with confidence and determination.

Raja Babu
Student ID: 21JE0740
Department of Chemical Engineering
Indian Institute of Technology (Indian School of Mines), Dhanbad

ABSTRACT

This project investigates the application of Physics-Informed Neural Networks (PINNs) to model and solve complex partial differential equations in the field of Chemical Engineering, specifically focusing on the heat equation in a two-dimensional domain under varying initial and boundary conditions, as well as the Navier-Stokes equations for fluid flow. Utilizing the computational capabilities of MATLAB, PyTorch, and COMSOL Multiphysics, this work addresses both time-dependent and time-independent (steady-state) heat distribution scenarios, simulating temperature evolution across the domain for multiple boundary conditions.

For the heat equation, the PINNs framework enables the modelling of both transient and steady-state temperature fields by enforcing physical laws directly within the neural network architecture, which circumvents the limitations of traditional numerical methods. The solution to the heat equation is computed for different initial conditions across the boundary, providing insights into heat distribution dynamics in varying thermal environments.

Furthermore, the study extends to the steady-state Navier-Stokes equation to model incompressible, potential flow through a 2D conduit, examining flow behaviour across three distinct regimes: laminar, transitional, and turbulent. PINNs are employed here to capture the complex velocity profiles and flow characteristics inherent to each regime, delivering accurate predictions that align with theoretical expectations and experimental validations. This research also includes the application of the Burgers' equation, a fundamental equation for modelling nonlinear advection-diffusion phenomena, further validating the robustness of the PINN approach in addressing both linear and nonlinear dynamics.

Overall, this project underscores the versatility and effectiveness of PINNs as a computational tool for solving complex, physics-driven equations in engineering domains, demonstrating substantial potential for broader applications in fluid dynamics, thermal systems, and beyond.

Table of Contents

Chapter 1

Introduction and Literature Review

1.1 Introduction

Neural networks, a subset of machine learning inspired by the human brain, have recently gained significant traction in scientific computing due to their ability to approximate complex non-linear functions with high accuracy, making them powerful tools for modelling intricate phenomena across various scientific disciplines. (Raissi, Perdikaris, & Karniadakis) In Chemical Engineering, accurate modelling is critical for design, optimization, and process control. Traditional methods often struggle to capture the complexity of real-world systems, especially when dealing with multi-scale phenomena, which can be effectively addressed by neural network-based approaches. (Chen & Karniadakis, 2018)

PINNs are a novel class of neural networks that incorporate physical laws, expressed as partial differential equations (PDEs), into their loss function during training. This allows them to learn solutions to complex PDEs directly from data, eliminating the need for analytical or numerical discretization techniques in many cases. (Raissi, Perdikaris, & Karniadakis)

PINNs can leverage limited data points, including boundary conditions, to learn the solution to a PDE across the entire domain. They can handle complex geometries and non-linear PDEs without the need for mesh generation. PINNs can be significantly faster than traditional numerical methods for certain problems, especially when high-dimensional solutions are required. (Karniadakis et al., 2021)

1.2 Brief History and Evolution of Neural Networks in Engineering

Neural networks have been applied to various engineering problems including pattern recognition, classification, regression, and control systems. Early applications often focused on simple network architectures like backpropagation-based networks for tasks like fault detection and system identification. (Haykin, 1999; Bishop, 2006)

Early work on neural networks for PDEs: The concept of using neural networks to solve differential equations dates back to the 1990s with research by Lagaris et al. (1998) who proposed the use of neural networks for solving boundary value problems. Breakthrough in PINNs: The term "Physics-Informed Neural Networks" was introduced by Raissi et al. (2019) who demonstrated the effectiveness of PINNs for solving a wide range of complex PDEs in various scientific domains. Ongoing research has explored extensions of PINNs to include uncertainty quantification, multi-scale modeling, and coupling with other numerical methods. (Karniadakis et al., 2021; Chen et al., 2018)

Simple PDEs: Initial research focused on solving relatively simple PDEs like the Burgers equation, demonstrating the potential of PINNs for capturing complex non-linear dynamics with limited data. (Raissi et al., 2019) Complex PDEs: Researchers have since applied PINNs to more challenging problems in fluid dynamics, heat transfer, elasticity, and other fields, often requiring adaptations to handle complex boundary conditions and multiple variables. (Karniadakis et al., 2021)

1.3 The PINN Paradigm: Defining Physics-Informed Neural Networks

A Physics-Informed Neural Network (PINN) is a type of neural network that leverages the principles of physics to solve complex partial differential equations (PDEs) by incorporating the governing physical laws directly into the network architecture, allowing it to approximate solutions with minimal data requirements compared to traditional numerical methods. Unlike standard neural networks that solely rely on minimizing the error between predicted and target outputs, PINNs include a loss term derived from the physical governing equations, forcing the network to learn solutions that satisfy the underlying physics. While PINNs can utilize limited data for training, they primarily rely on the physics equations to guide the learning process, making them particularly useful in scenarios where experimental data is scarce or noisy.

Scope and Applications in Scientific Computation and Modelling:

Fluid Dynamics: Solving complex fluid flow problems like turbulent flow, where analytical solutions are often intractable.

Heat Transfer: Simulating heat transfer phenomena in complex geometries with intricate boundary conditions.

Solid Mechanics: Modelling stress and strain distributions in materials under various loading conditions.

Electromagnetics: Solving electromagnetic wave propagation problems in complex environments.

Inverse Problem Solving: Estimating unknown parameters in a physical system based on limited observations.

1.4 Need for PINNs in Chemical Engineering

Chemical engineering processes often involve complex nonlinear phenomena described by PDEs, where traditional numerical methods can struggle with high computational cost, particularly when dealing with complex geometries or rapidly changing conditions. PINNs offer a powerful alternative by leveraging the computational efficiency of neural networks while still adhering to the underlying physical laws.

Current Challenges in Chemical Engineering Computational Modelling is for the High Dimensional Problems like for Complex systems with many variables often require computationally expensive numerical techniques. Experimental data may be limited or expensive to acquire, hindering traditional data-driven modelling approaches. Modelling intricate reactor designs or heat exchanger configurations can be challenging with conventional methods.

PINNs can handle complex geometries and boundary conditions with relative ease, often requiring less computational effort than traditional numerical methods. Even with limited data, PINNs can leverage physical laws to make accurate predictions by learning the underlying relationships. PINNs can effectively tackle problems with a large number of variables, which can be challenging for traditional numerical methods. Optimizing reactor configurations and operating conditions by predicting concentration profiles within the reactor. Modelling heat transfer processes in complex heat exchangers, including unsteady-state behaviour. Predicting mass transfer rates in multiphase systems like distillation columns. Identifying unknown parameters in chemical reaction kinetics based on limited experimental data. (Wang, 2021) (Cai, 2021)

Chapter 2

Structure and Mathematical Framework of Physics-Informed Neural Networks (PINNs)

A PINN is a neural network architecture designed to solve complex partial differential equations (PDEs) by leveraging the power of deep learning, incorporating the physics governing the problem directly into the network through the loss function. Key components include the input layer (representing spatial coordinates or other relevant parameters), hidden layers (where non-linear transformations occur), and the output layer (providing the solution to the PDE).

2.1 Fundamentals of PINN Structure:

PINNs can handle complex geometries and non-linear PDEs, making them a powerful tool for a wide range of applications.

PDE Representation: The PDE is expressed as a residual function, where the network output is substituted into the PDE and the difference between the two is calculated. This residual is then used as part of the loss function to train the network to minimize the error between the predicted solution and the actual solution to the PDE.

Loss Function: A typical PINN loss function includes a combination of the residual loss (penalty for not satisfying the PDE) and a loss term for boundary conditions, initial conditions, and potentially collocation points within the domain.

Unlike traditional numerical methods, PINNs often require only a limited amount of data points to accurately solve the PDE. (Raissi, Perdikaris, & Karniadakis) The seminal paper introduced the concept of PINNs and demonstrated their effectiveness on various PDEs. "Deep Learning for Solving Partial Differential Equations" by Sirignano and Spiliopoulos (2018) which discusses the theoretical foundation of using neural networks to solve PDEs and provides insights into the choice of network architecture.

2.2 Loss Functions in PINNs

Physics-Based Loss Functions is the combination of various types of the losses like Residual Loss, Boundary Condition Loss and Initial Condition Loss. The choice of loss function plays a crucial role in the accuracy and stability of the PINN solution.

Residual Loss: The primary component of the loss function, calculated as the difference between the network output and the PDE residual at collocation points within the domain.

Boundary Condition Loss: Ensures the network output satisfies the specified boundary conditions at the domain boundaries.

Initial Condition Loss (if applicable): Similar to boundary condition loss, but applied to initial conditions.

2.3 Collocation Points

Randomly sampled points within the computational domain where the PDE residual is evaluated to train the network. The distribution of collocation points can significantly impact the training process.

2.4 Error Metrics and Convergence Criteria

L2 Norm: A common metric to quantify the error between the predicted solution and the exact solution.

Relative Error: Measures the error relative to the magnitude of the solution.

Convergence Criteria: Stop training the network when the loss function reaches a predefined threshold or when the error metrics stabilize.

2.5 The Universal Approximation Theorem

In the con of PINNs, the Universal Approximation Theorem implies that neural networks can approximate complex, multi-dimensional functions that satisfy the physical laws (e.g., differential equations). PINNs leverage this property by encoding physical constraints directly into the network's training, allowing the network to learn solutions to differential equations representing physical systems.

Given a continuous function $(f: R^n \to R)$ defined on a compact subset $(K \subset R^n)$ and any $(\epsilon > 0)$, there exists a neural network $(\phi(x;\theta) = \sum_{i=1}^{M} a_i \sigma(w_i^T x + b_i))$ with weights $(w_i \in R^n)$, biases $(b_i \in R)$, output weights $(a_i \in R)$, and activation function (σ), such that

$$|f(x) - \phi(x;\theta)| < \epsilon \quad \text{for all } x \in K.$$

The theorem's implications for engineering are significant because it allows neural networks to model complex, high-dimensional relationships found in physical and engineering problems. By embedding the physical laws directly into neural network loss functions, PINNs can provide approximate solutions to problems in fluid dynamics, material science, and other domains where traditional numerical methods may be computationally expensive. Thus, the Universal Approximation Theorem underpins the utility of PINNs in simulating and optimizing complex systems.

2.6 Function Space Exploration in PINNs

In Physics-Informed Neural Networks (PINNs), the process of function space exploration involves identifying solution functions that satisfy both data constraints and the governing physical laws, often represented as partial differential equations (PDEs). Formally, given a function $(u(x,t))$ that represents a solution to a PDE, PINNs seek to approximate (u) by minimizing a loss function $(\mathcal{L} = \mathcal{L}_{data} + \mathcal{L}_{physics})$, where $(\mathcal{L}_{data})$ penalizes discrepancies from known data points and $(\mathcal{L}_{physics})$ imposes the PDE constraints through differential operators.

Here's a general mathematical model to explore a function space in the con of constrained function approximation and optimization. Suppose we are given a target function $(f: R^n \to R)$, where we want to approximate (f) using a candidate function $(\phi(x;\theta))$ from a parameterized function space, subject to constraints.

a. Function Approximation Objective: Define the approximation function $(\phi: R^n \to R)$, parameterized by $(\theta \in \Theta)$ (the space of parameters):

$$\phi(x;\theta) = \sum_{i=1}^{M} a_i \sigma\left(w_i^T x + b_i\right),$$

where $(a_i \in R, w_i \in R^n, b_i \in R)$, and (σ) is a non-linear activation function

b. Loss Function Definition: The goal is to find parameters (θ) such that the error between $(f(x))$ and $(\phi(x;\theta))$ is minimized. Define a loss function $(\mathcal{L}(\theta))$:

$$\mathcal{L}(\theta) = \mathcal{L}_{data}(\theta) + \mathcal{L}_{constraint}(\theta)$$

Where, Data Loss $(\mathcal{L}_{\text{dt}}(\theta))$: Penalizes the discrepancy between $(f(x))$ and $(\phi(x;\theta))$ at a set of sample points $(\{x_i\}_{i=1}^N)$:

$$\mathcal{L}_{\text{data}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} |f(x_i) - \phi(x_i;\theta)|^2$$

And Constraint Loss $(\mathcal{L}_{constraint}(\theta))$: Encodes constraints, such as differential equations or boundary conditions, typically applied to (ϕ). For a differential operator $(\mathcal{D})$ that (ϕ) must satisfy, e.g., $(\mathcal{D}\phi(x) = 0)$, define:

$$\mathcal{L}_{constraint}(\theta) = \frac{1}{K}\sum_{j=1}^{K} |\mathcal{D}\phi(x_j)|^2,$$

where (x_j) are points in the domain.

c. Optimization Problem: Minimize $(\mathcal{L}(\theta))$ over the parameter space (Θ):

$$\theta^* = \arg\min_{\theta\in\Theta} \mathcal{L}(\theta).$$

d. Gradient-Based Optimization: Using a gradient descent method or an alternative optimization technique, iterate until convergence:

$$\theta_{k+1} = \theta_k - \eta\nabla_{\theta\mathcal{L}}(\theta_k),$$

Key challenges include the high-dimensional nature of the function spaces and the need for PINNs to satisfy physical constraints over potentially vast input domains. The non-convexity of the loss landscape, combined with sparse or noisy data, complicates the optimization process. Further, the underlying physical PDEs often involve complex operators that increase computational demands, leading to issues like gradient vanishing or exploding during training.

Chapter 3

Introduction to PDEs in Heat Transfer and Fluid Flow

Partial Differential Equations (PDEs) play a central role in modelling physical processes in heat transfer and fluid dynamics, governed by conservation laws of energy, mass, and momentum. In this chapter, we discuss the mathematical formulation and application of PDEs, particularly the heat equation and the Navier-Stokes equations, as well as the methods used to solve these equations using Physics-Informed Neural Networks (PINNs).

3.1 Overview of the Heat Equation and Its Applications

The heat equation is a PDE derived from **Fourier's Law of Heat Conduction** and the **conservation of energy**. It models the distribution of temperature in a given domain over time and space, capturing how heat diffuses through a medium.

Fourier's Law of Heat Conduction states that the heat flux ($\vec{q}$) is proportional to the negative gradient of the temperature field (T):

$$\vec{q} = -k\nabla T$$

where (k) is the thermal conductivity of the material.

Energy Conservation: For an infinitesimal volume element, the rate of heat accumulation is balanced by the heat flux in and out. Applying this principle leads to the heat equation:

$$\frac{\partial T}{\partial t} = \alpha \nabla^2 T$$

where ($\alpha = \frac{k}{\rho c_p}$) is the thermal diffusivity, with (ρ) as the density and (c_p) as the specific heat capacity of the material.

Applications:

1. **Steady-State Heat Distribution**: Used in the analysis of systems where temperature does not change with time, such as heat exchangers and steady thermal insulation.
2. **Transient Heat Transfer**: Analyzes time-dependent temperature profiles, applicable in processes like material cooling, transient heating in electronics, and dynamic thermal analysis of structures.

3.2 Navier-Stokes Equations in Fluid Dynamics

The Navier-Stokes equations govern fluid motion by combining **Newton's Second Law** with the principles of **conservation of mass** and **momentum**. These equations describe the behaviour of fluid flow in terms of velocity and pressure fields.

Conservation of Mass (Continuity Equation): For an incompressible fluid, the continuity equation ensures that mass is conserved:

$$\nabla \cdot \vec{u} = 0$$

where $(\vec{u} = (u, v, w))$ represents the velocity field of the fluid.

Conservation of Momentum: Newton's Second Law applied to fluid elements gives the Navier-Stokes equations:

$$\rho\left(\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u}\right) = -\nabla p + \mu \nabla^2 \vec{u} + \vec{f}$$

where (ρ) is the fluid density, (p) is the pressure, (μ) is the dynamic viscosity, and $(\vec{f})$ is any external body force (e.g., gravity).

Applications:

Potential Flow and Laminar Flow: Navier-Stokes in simple, low-Reynolds number flows, such as slow-moving fluids or flows with minimal turbulence.

Turbulent Flow: High-Reynolds number scenarios where chaotic flow occurs, relevant in weather systems, aerodynamics, and industrial fluid processing.

3.3 Challenges in Traditional Numerical Approaches

Solving the heat equation and Navier-Stokes equations typically requires numerical methods due to the complexity of boundary conditions and nonlinearity in the equations. Common methods include:

Finite Difference Method (FDM): Approximates derivatives using grid points, suitable for structured grids but sensitive to grid resolution and stability issues in complex domains.

Finite Element Method (FEM): Divides the domain into elements, allowing flexibility for irregular geometries. However, FEM can be computationally intensive.

Finite Volume Method (FVM): Uses control volumes to ensure conservation, commonly used in CFD but may require significant computational resources and fine meshing.

Challenges:

1. **Mesh Dependency**: Fine meshes increase accuracy but at a high computational cost.

2. **Convergence Issues**: Nonlinear terms in Navier-Stokes can lead to stability and convergence difficulties.

3. **Boundary Condition Handling**: Complex or variable boundary conditions add significant complexity.

4. **Computational Cost**: For high-dimensional or time-dependent problems, traditional methods can be prohibitively expensive.

3.4 Advantages of Using PINNs for PDE-Based Modelling

Physics-Informed Neural Networks (PINNs) offer a promising alternative by embedding physical laws into the loss function of neural networks, allowing the network to learn solutions to PDEs by minimizing deviations from these laws.

PINNs Framework:

Loss Function Design: The PINN loss function incorporates terms representing the PDE residuals, boundary conditions, and initial conditions. For example, for the heat equation, the loss function ($\mathcal{L}$) could be defined as:

$$\mathcal{L} = |\frac{\partial T}{\partial t} - \alpha \nabla^2 T|^2 + \sum (\text{Boundary and Initial Condition Terms})$$

Collocation Points: PINNs do not require a mesh but instead sample collocation points within the domain and boundaries to evaluate the loss terms, reducing computational overhead.

Advantages:

1. **Mesh-Free Computation**: PINNs reduce the reliance on grid resolution, leading to a more flexible solution that adapts to complex geometries and boundary conditions.

2. **Handling Complex Boundary Conditions**: PINNs can adapt to arbitrary boundary conditions, such as sinusoidal or exponential functions, by encoding them directly in the loss function.

3. **Automatic Differentiation**: Using frameworks like PyTorch, PINNs leverage automatic differentiation for accurate gradient calculation, which is essential for PDE constraints.

4. **Scalability and Parallelism**: Neural networks inherently support parallel computing, which is beneficial for large-scale or high-dimensional problems.

Challenges in PINNs:

- **Hyperparameter Tuning**: PINNs require careful tuning of neural network parameters and collocation points to achieve accurate solutions.

- **Computational Resources for Training**: Although PINNs are mesh-free, training deep networks on complex domains may still be resource-intensive.

- **Error Control and Convergence**: PINNs lack the traditional error control mechanisms found in classical numerical methods, requiring additional metrics to ensure solution accuracy.

In summary, PINNs present a transformative approach for solving PDEs in heat transfer and fluid dynamics, combining the flexibility of neural networks with embedded physics to handle complex, physics-driven problems in ways that traditional methods struggle with. The following sections provide a comprehensive examination of applying PINNs to solve both the heat equation and Navier-Stokes equations in two dimensions.

Chapter 4

Solving the Heat Equation with Physics-Informed Neural Networks (PINNs)

4.1 Problem Setup and Mathematical Formulation of the Heat Equation

4.1.1 Problem Statement

Define the problem of finding the steady-state temperature distribution ($T(x, y)$) on a thin rectangular plate. Describe the plate's dimensions, where:

($0 \leq x \leq L$) represents the (x)-axis.

($0 \leq y \leq W$) represents the (y)-axis.

Heat conduction occurs only within the plane of the plate, governed by Fourier's law of heat conduction.

4.1.2 Boundary Conditions for the Plate

Specify boundary conditions critical to the setup, ensuring clarity on temperature variations and uniformities:

1. **Top and Side Edges at Constant Temperature**:

Along the left ($(x = 0)$), right($(x = L)$), and top($(y = W)$) edges, the temperature is fixed at (T_0).

2. **Bottom Edge with Variable Temperature Profile**:

- Along the bottom edge ($(y = 0)$), the temperature varies with position (x):
  - **Constant Offset**: ($T(x, 0) = T_0 + C$)
  - **Sine Function**: ($T(x, 0) = T_0 + A \sin\left(\frac{2\pi n x}{L}\right)$), introducing periodic temperature variations.
  - **Exponential Variation**: ($T(x, 0) = T_0 + A \exp\left(\frac{x}{L}\right)$), representing a gradual increase or decrease along the edge.

4.2 Mathematical Derivation of the Heat Equation in Two Dimensions

4.2.1 Derivation of the Heat Equation

Present the derivation of the two-dimensional heat equation using:

Conservation of Energy: Describe the fundamental energy conservation applied to an infinitesimal control volume

Fourier's Law: Use Fourier's law ($\vec{q} = -k\nabla T$) to relate heat flux to the temperature gradient.

Final PDE Form: Derive the time-dependent heat equation:

$$\frac{\partial T}{\partial t} = \alpha\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$

where (α) is the thermal diffusivity.

4.2.2 Reduction to Steady-State Conditions

For steady-state scenarios (where ($\partial T/\partial t = 0$)), reduce the equation to the Laplace form:

$$\nabla^2 T = 0 \quad \Rightarrow \quad \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0$$

Outline assumptions, such as no internal heat sources and a homogeneous medium, that support using the steady-state condition.

4.3 Neural Network Architecture for Heat Equation Modeling

The architecture of the **HeatTransferPINN** model is designed to approximate solutions to the heat equation through a physics-informed neural network (PINN) framework. The model structure includes:

Activation Function: SiLU, which provides smooth nonlinear activation to enhance the learning process.

Layer Composition:

A sequence of linear layers initialized as follows:

```
HeatTransferPINN(
  (activation): SiLU()
  (layers): ModuleList(
    (0): Linear(in_features=18, out_features=128, bias=True)
    (1): Linear(in_features=128, out_features=256, bias=True)
    (2-3): 2 x Linear(in_features=256, out_features=256, bias=True)
    (4): Linear(in_features=256, out_features=128, bias=True)
    (5): Linear(in_features=128, out_features=1, bias=True)
  )
  (projection_layers): ModuleList(
    (0): Linear(in_features=128, out_features=256, bias=True)
    (1-2): 2 x Identity()
    (3): Linear(in_features=256, out_features=128, bias=True)
    (4): Linear(in_features=128, out_features=1, bias=True)
  )
)
```

```
Optimizer : Adam

Loss Function : MSE Loss

Additionally, a positional encoding function is implemented to improve
the model's capability to generalize across the spatial domain.
```

```python
def positional_encoding(x, num_encodings, enable_encoding=True):
    if not enable_encoding:
        return x
    encodings = [x]
    for i in range(num_encodings):
        factor = min(2  i, 1e3)  # Prevent overflow
        encodings.append(torch.sin(factor * x))
        encodings.append(torch.cos(factor * x))
    return torch.cat(encodings, dim=-1).to(device)
```

4.4 Boundary Condition Configurations for Model Training

To evaluate the performance of the HeatTransferPINN model, we explore distinct boundary condition configurations applied to the plate's bottom edge, focusing on steady-state temperature distributions.

4.4.1 Constant Boundary Condition

For the constant boundary condition along the plate's bottom edge, the temperature $(T(x,0))$ is set to a fixed value across the entire edge. Specifically, we define:

$$T(x,0) = T_0 + \Delta T$$

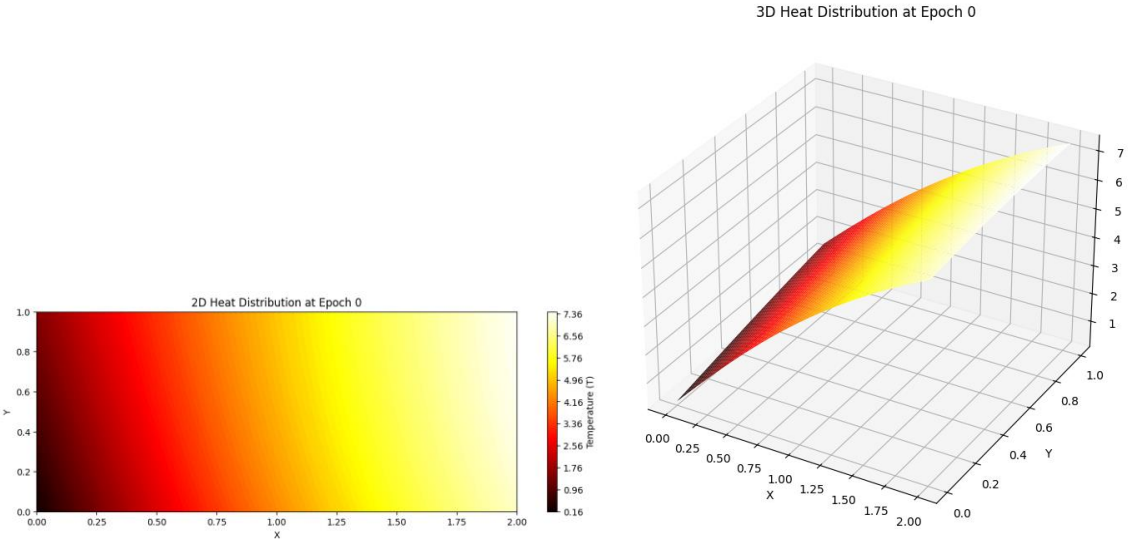
where:

- (T_0) is the reference temperature along the other edges,

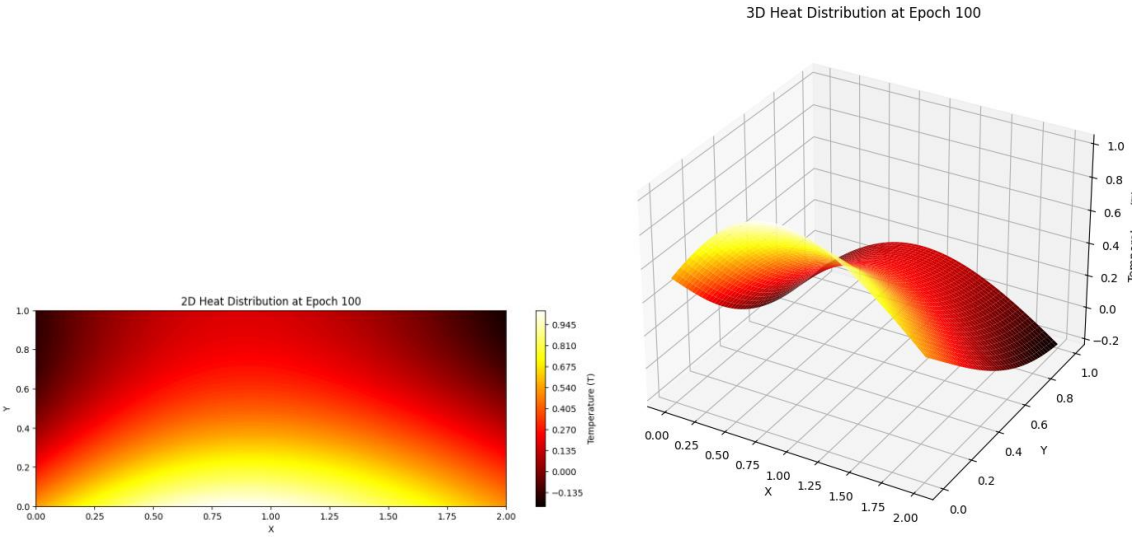
- $(\Delta T = 10 \text{ ^ } \circ \{C\})$ denotes the constant temperature difference applied solely along the bottom edge $(y = 0)$.

The model is trained to uphold this boundary condition, ensuring that temperature values along the bottom edge remain invariant with respect to (x). The resulting temperature field, $(T(x,y))$, is plotted as $(T(x,y))$ versus (x) and (y) to visualize the steady-state distribution, highlighting the uniformity of $(T(x,0))$ across the (x)-axis.

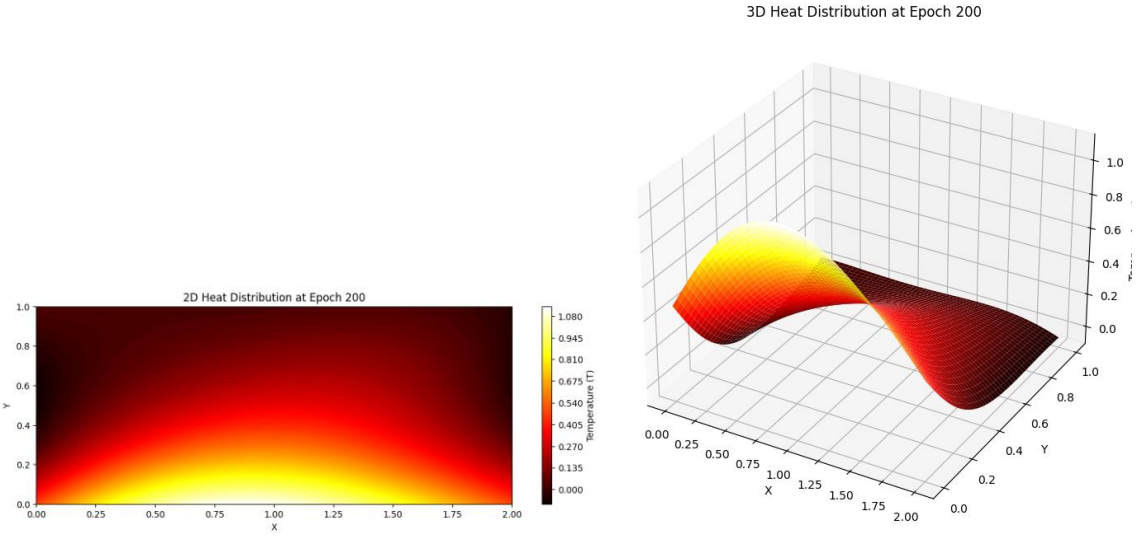
Epoch 0, Loss: 0.003833406837657094



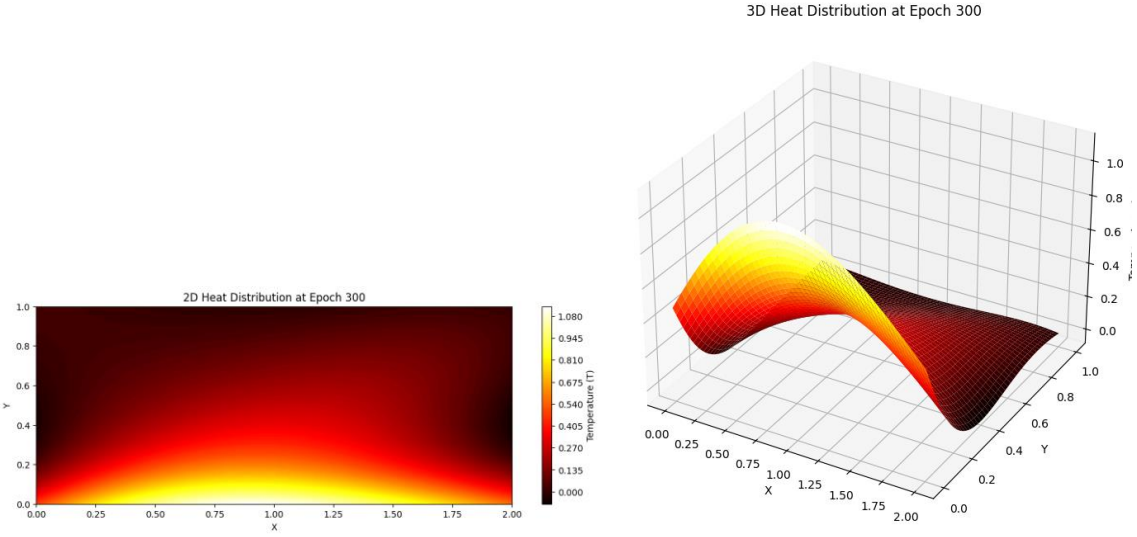
Epoch 100, Loss: 0.00040376943070441484



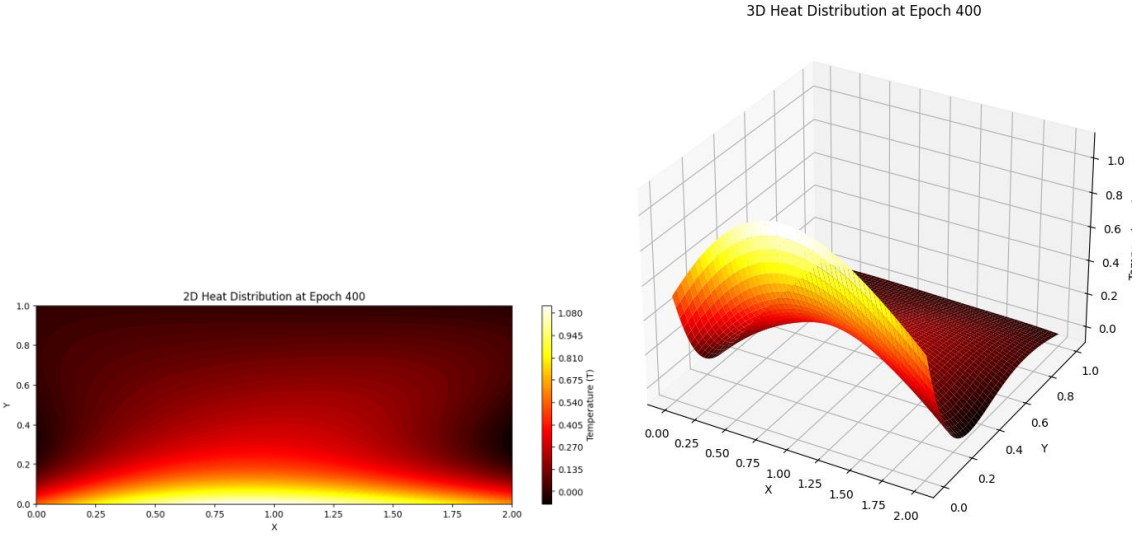
Epoch 200, Loss: 0.00024605804355815053



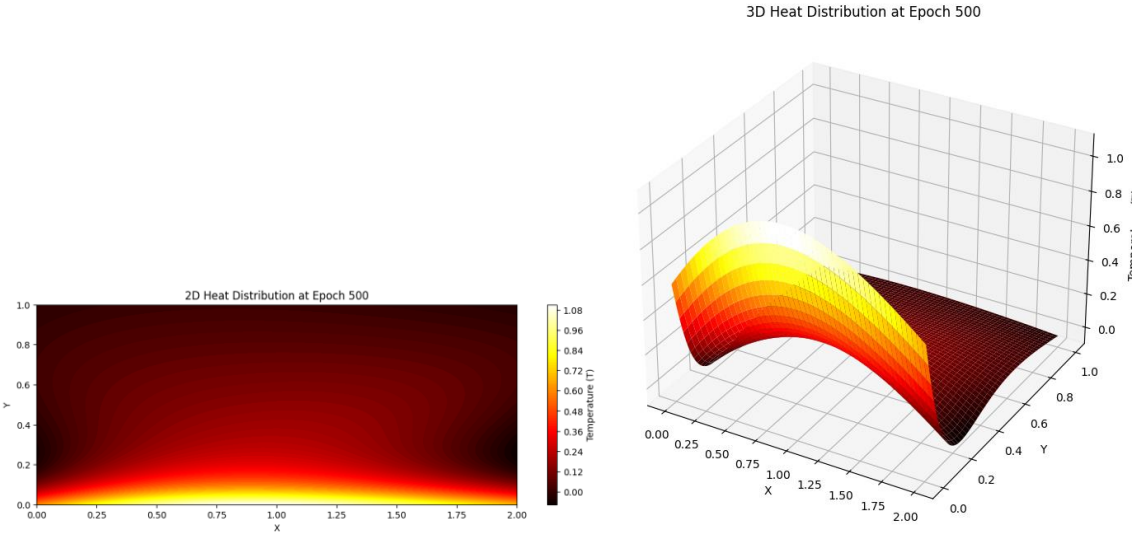
Epoch 300, Loss: 0.0001982759276870638



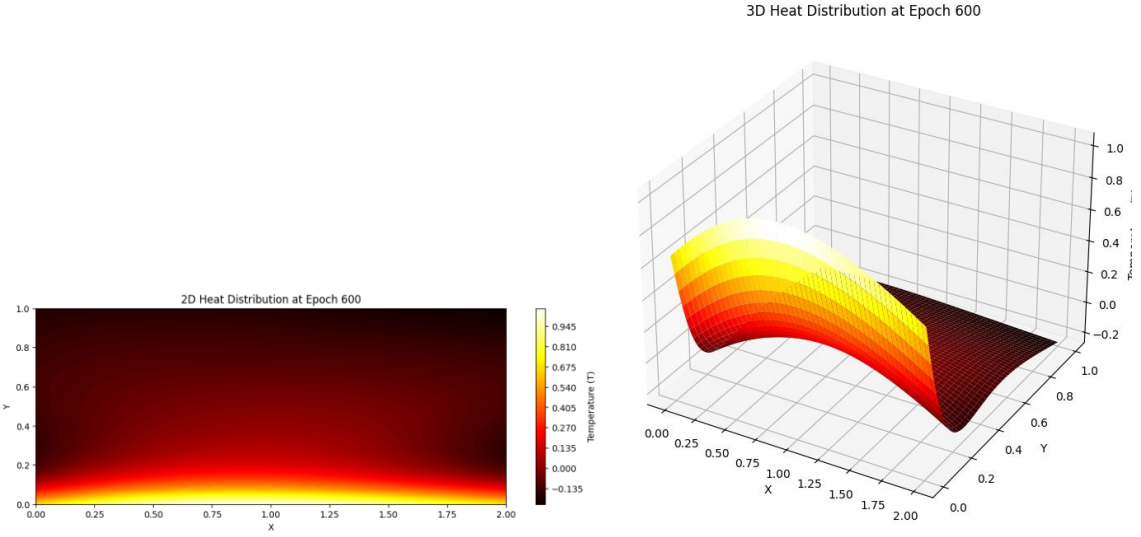
Epoch 400, Loss: 0.000167673802934587



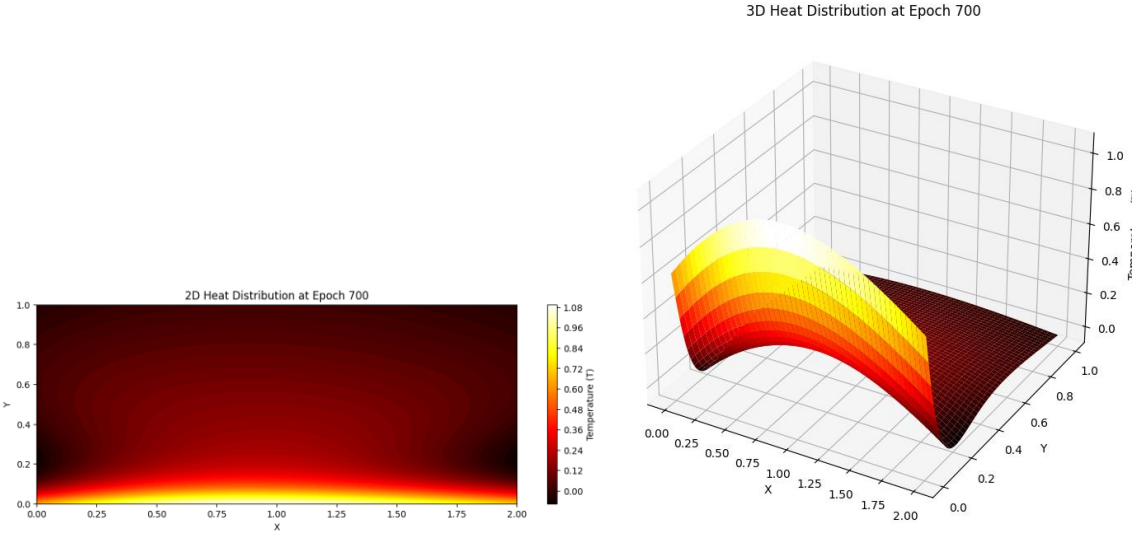
Epoch 500, Loss: 0.00014239414304029197



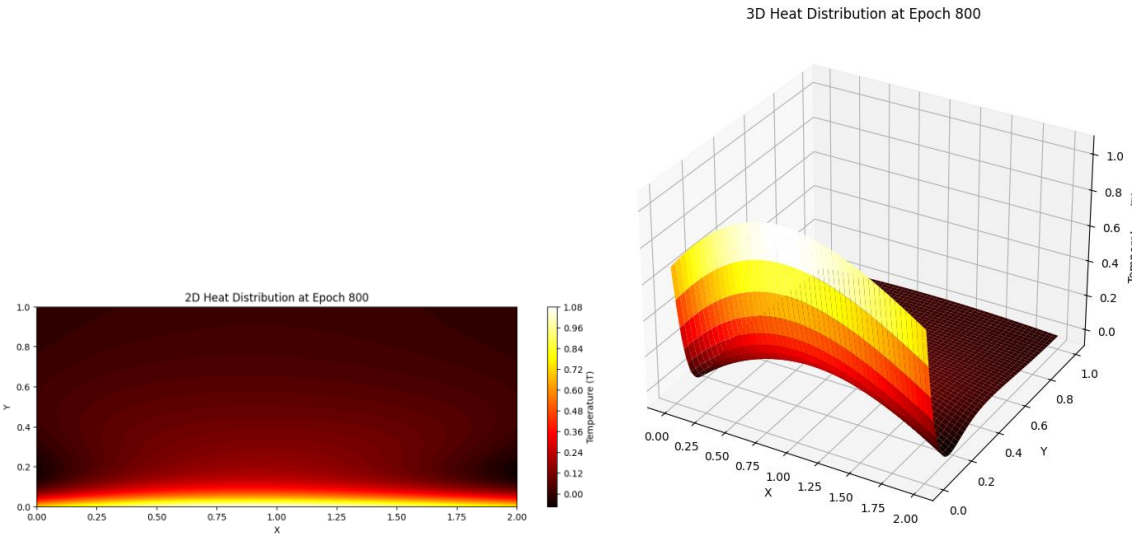
Epoch 600, Loss: 0.0001905586977954954



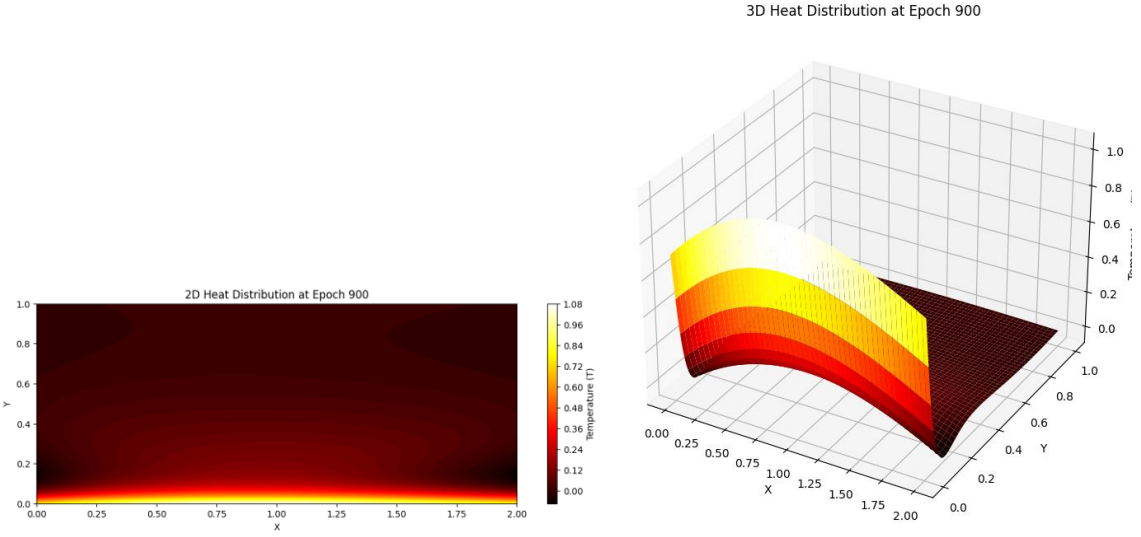
Epoch 700, Loss: 0.00012782012345269322



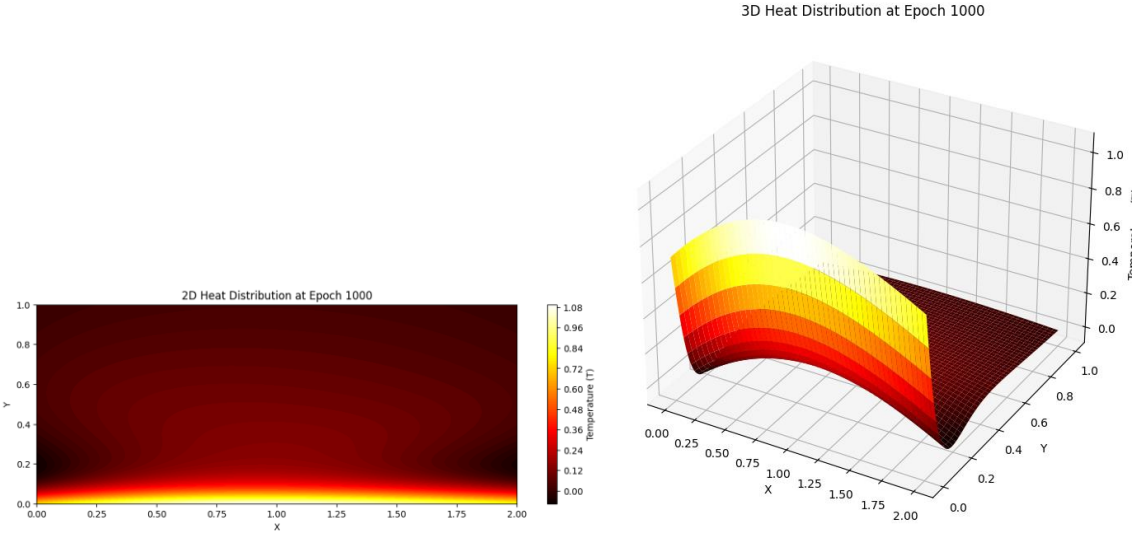
Epoch 800, Loss: 0.00010682211723178625



Epoch 900, Loss: 9.03816326172091e-05



Epoch 1000, Loss: 0.00011656971764750779



4.4.2 Sinusoidal Boundary Condition

For the sinusoidal boundary condition, we introduce a periodic temperature variation along the bottom edge of the plate, modeled as:

$$T(x,0) = T_0 + A\sin\left(\frac{2\pi nx}{L}\right)$$

where:

- (T_0) is the baseline temperature,

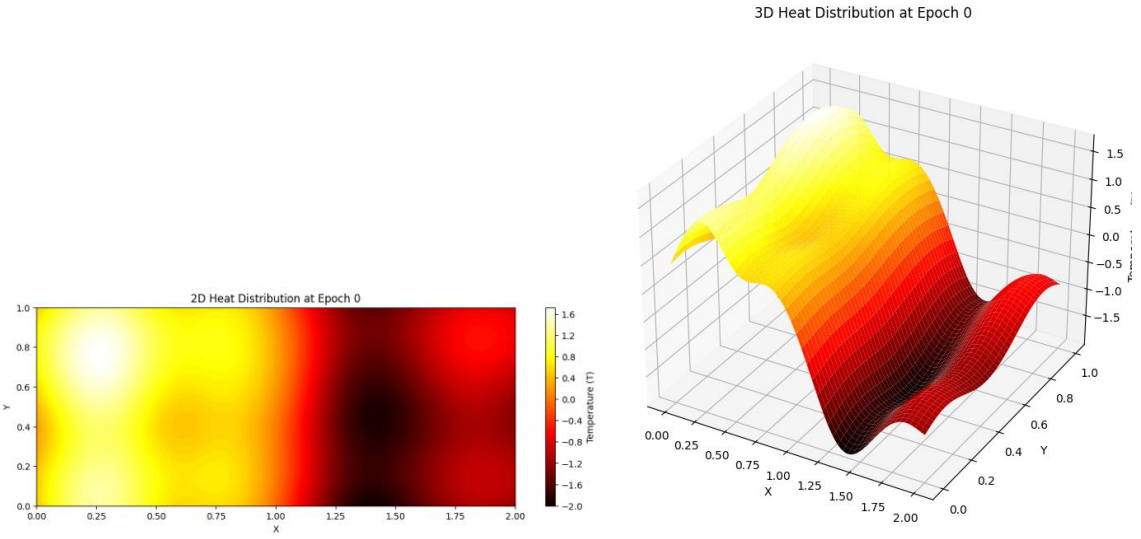
- ($A = 10$ ^ ∘ {C}) is the amplitude of the sinusoidal temperature variation,

- (n) is the frequency coefficient, representing the number of complete sinusoidal cycles along the length (L) of the plate's bottom edge,

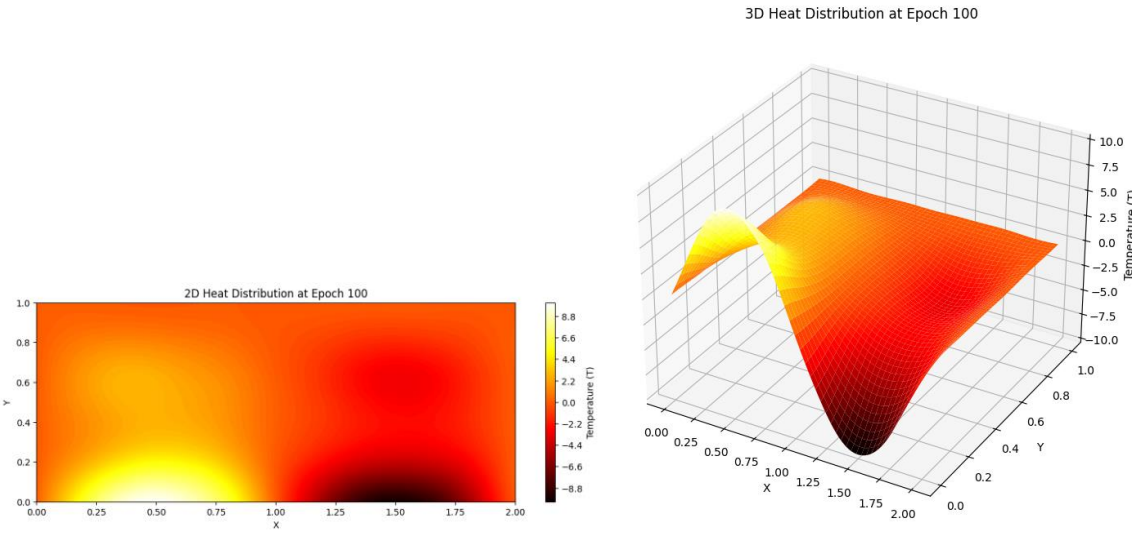
- (L) is the length of the plate along the (x) −axis.

In this setup, the temperature along the bottom edge ($y = 0$) varies sinusoidally as a function of (x), introducing periodic boundary conditions that reflect real-world scenarios where temperature fluctuations can propagate through conductive materials. The model is trained with this periodic input, and the resulting temperature field ($T(x,y)$) is visualized by plotting ($T(x,y)$) against (x) and (y). This plot illustrates the spatial variations introduced by the sinusoidal boundary condition, allowing for an examination of how periodic temperature patterns influence the overall distribution.

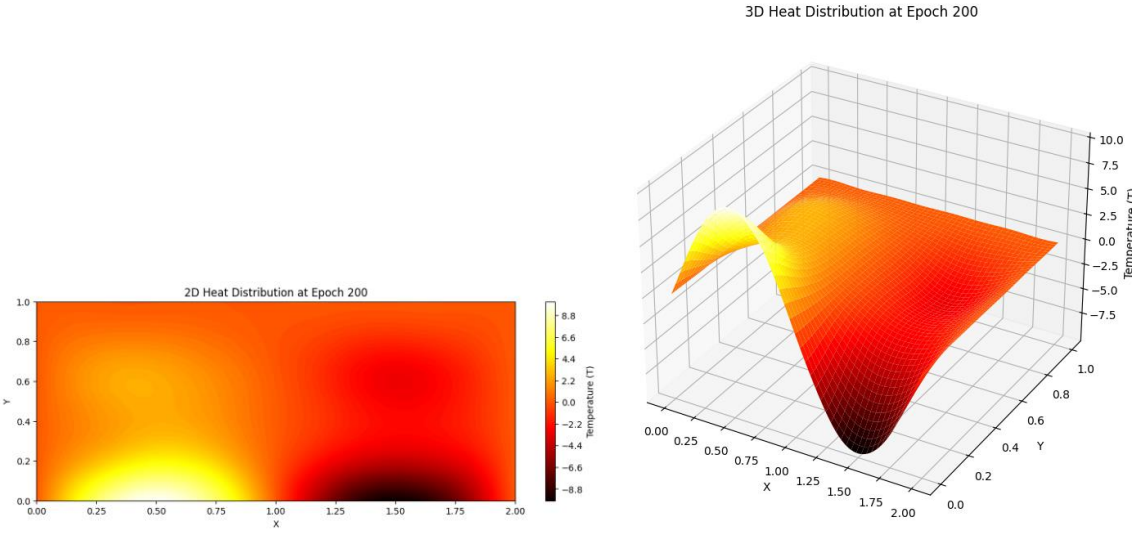
```
Epoch 0, Loss: 0.12245474010705948
```



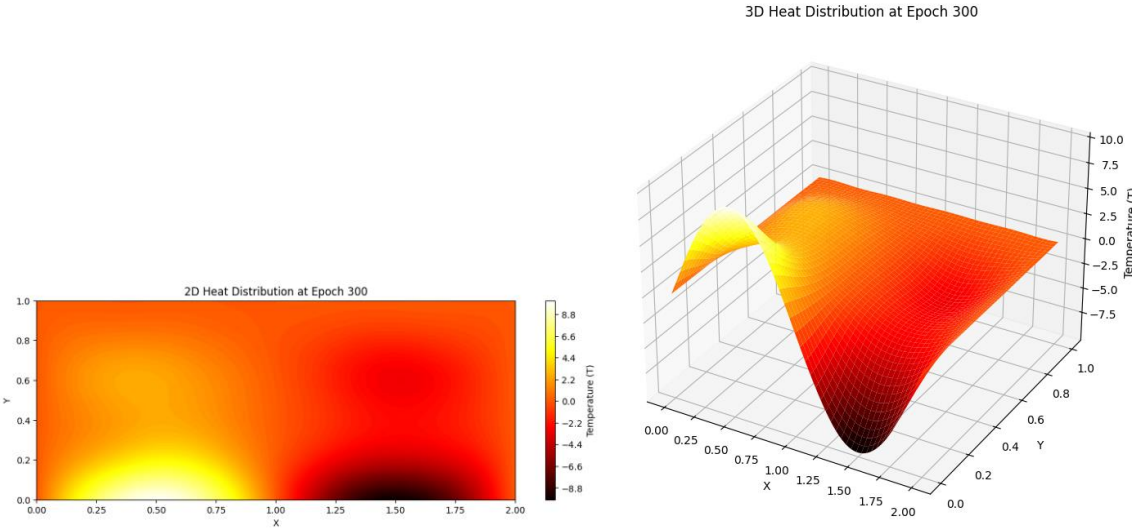
Epoch 100, Loss: 4.3964406359009445e-05



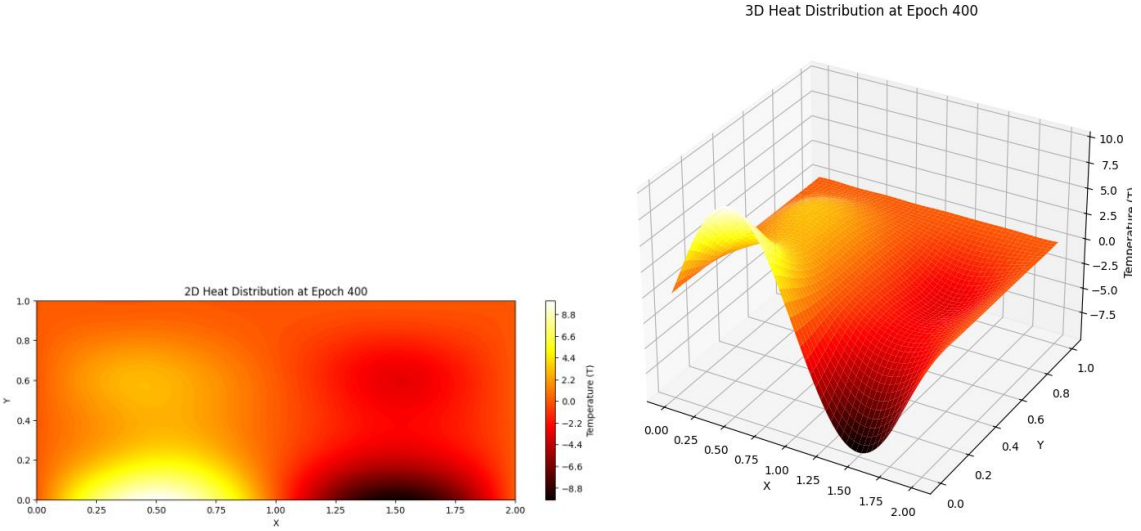
Epoch 200, Loss: 1.0129984730156139e-05



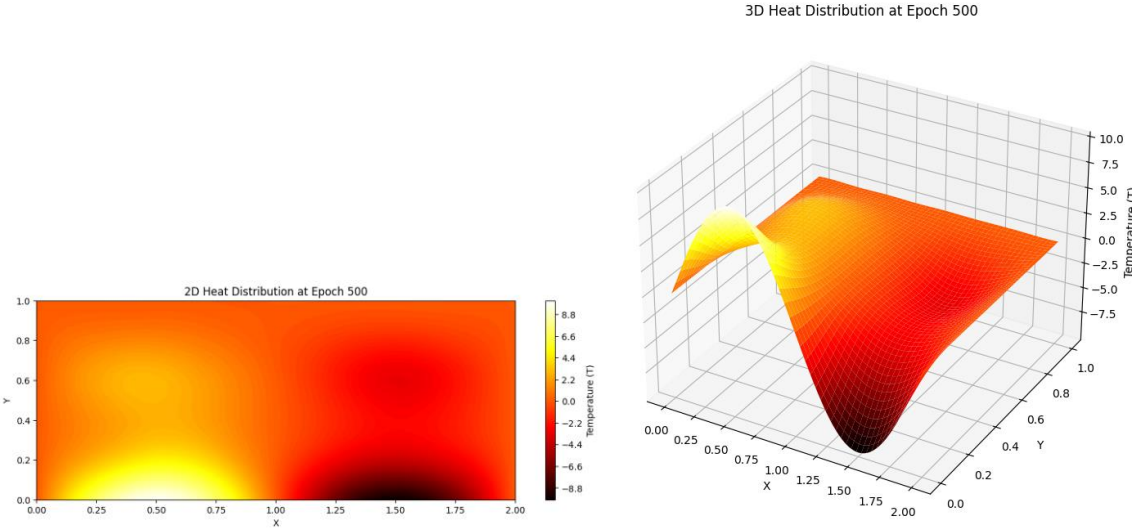
Epoch 300, Loss: 4.260995410731994e-06



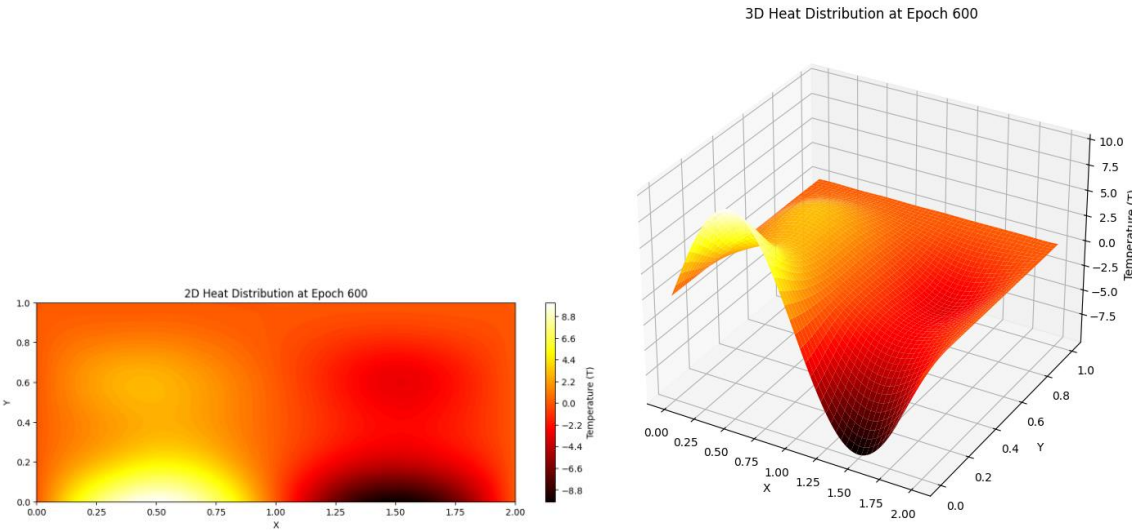
Epoch 400, Loss: 2.085906635329593e-05



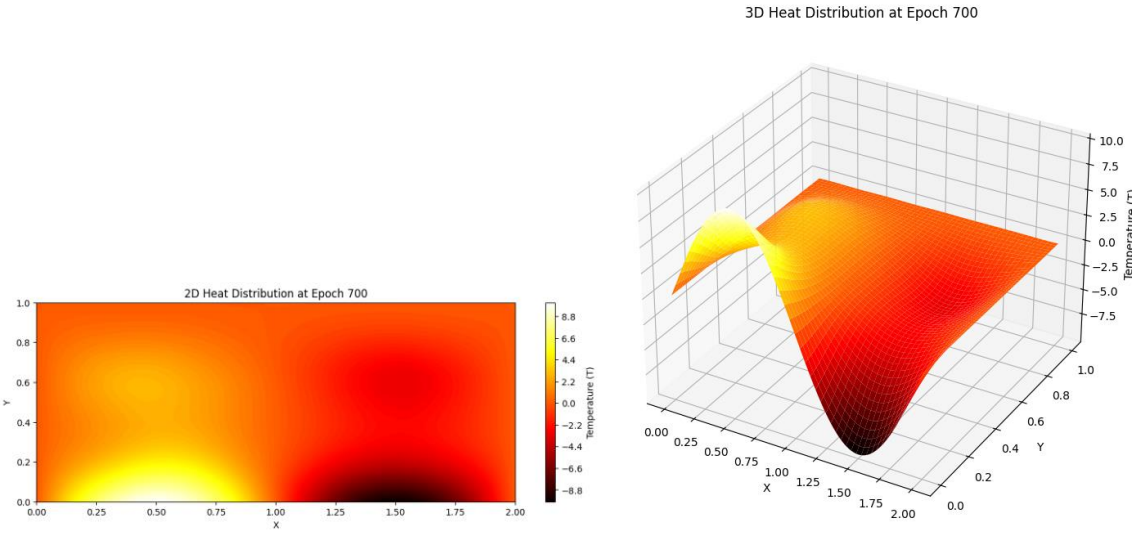
Epoch 500, Loss: 1.9897827314707683e-06



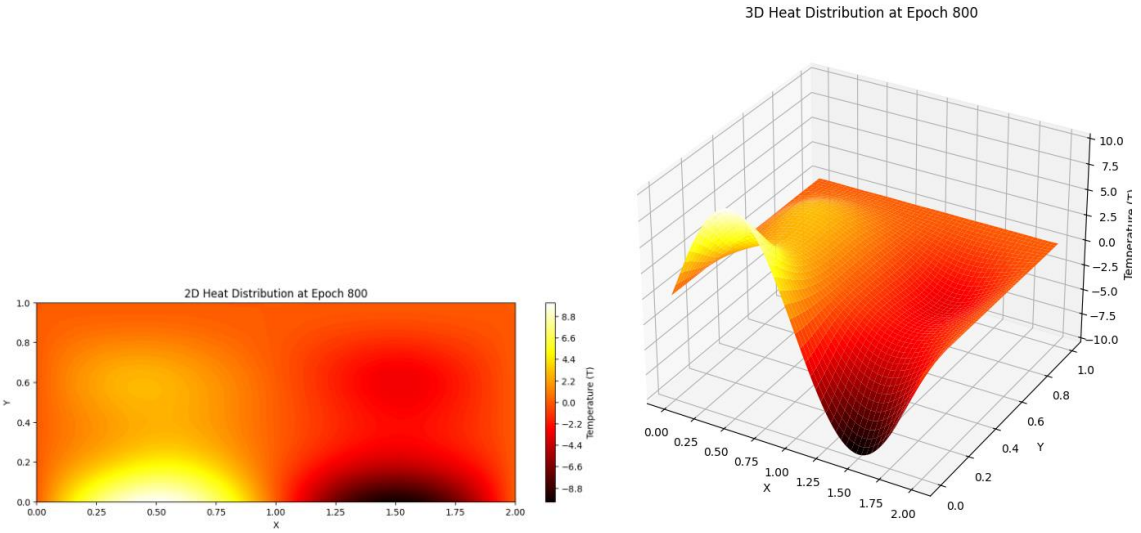
Epoch 600, Loss: 1.3831395335728303e-06



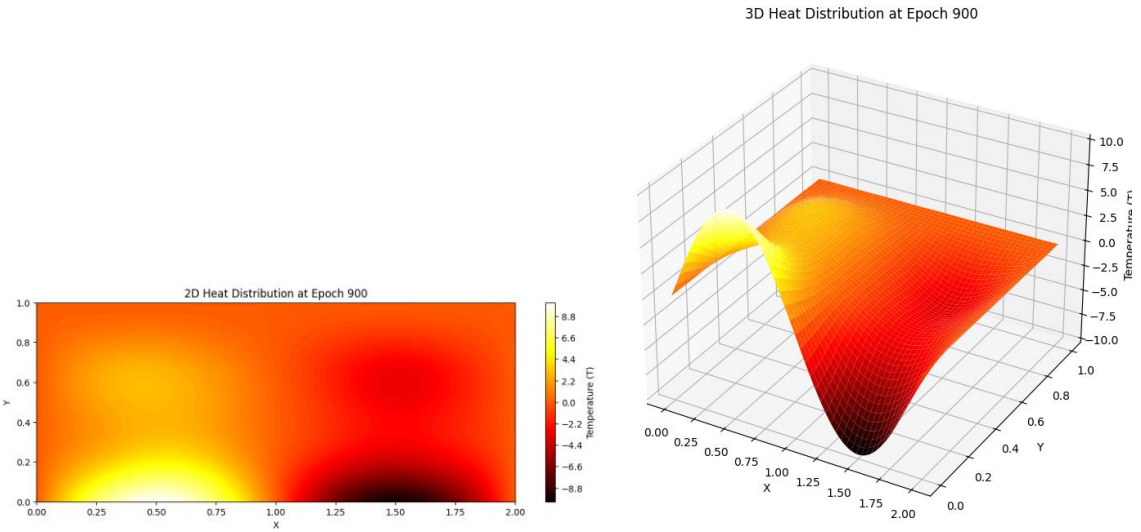
Epoch 700, Loss: 1.0433702755108243e-06



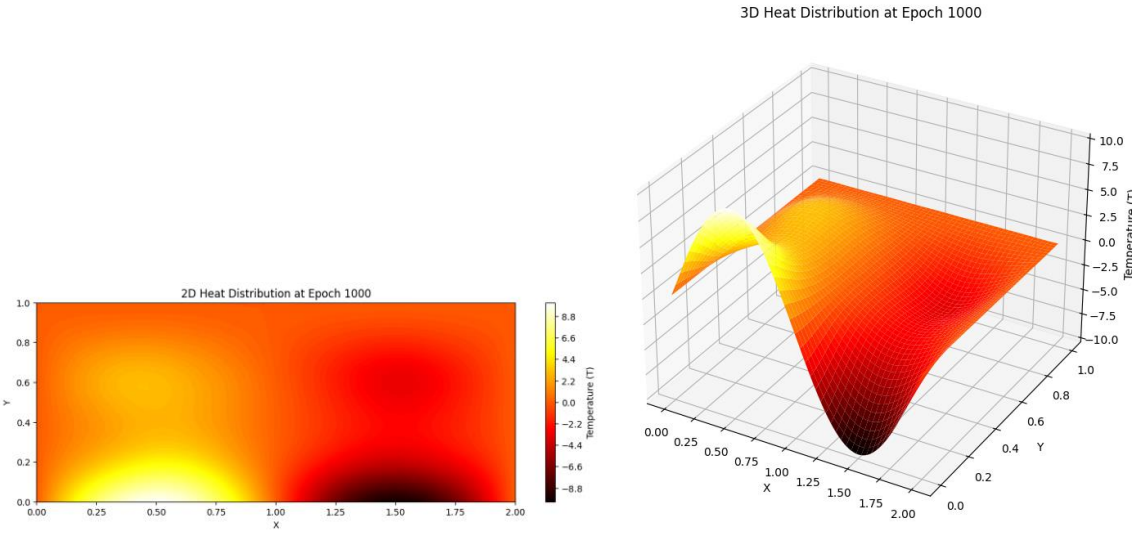
Epoch 800, Loss: 8.408109124502516e-07



Epoch 900, Loss: 1.2133270956837805e-06



Epoch 1000, Loss: 7.332453151320806e-07



Chapter 5

Solving the Navier-Stokes Equation for 2D Conduit Flow

5.1 Problem Setup and Mathematical Formulation of the Navier-Stokes Equation

5.1.1 Problem Statement

We aim to determine the velocity field ($\mathbf{u}(x, y)$) and pressure field ($p(x, y)$)for incompressible, steady-state flow through a two-dimensional rectangular conduit. The conduit is defined as follows:

($0 \leq x \leq L$) represents the (x)-axis, with (L) being the conduit length.

($0 \leq y \leq W$) represents the (y)-axis, with (W) being the conduit width.

In this setup:

Enters at the left boundary($(x = 0)$)and exits at the right boundary ($(x = L)$).

The flow is considered incompressible and fully developed, with a potential flow assumption.

Different flow regimes (Laminar, Transitional, and Turbulent) are investigated based on the Reynolds number classification.

The governing equations for this problem are the incompressible Navier-Stokes equations, which can be expressed as:

1. **Continuity Equation** for incompressibility:

$$\nabla \cdot \mathbf{u} = 0$$

2. **Momentum Equations** (in the (x)- and (y)-directions):

$$\rho(\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \mu \nabla^2 \mathbf{u}$$

5.1.2 Boundary Conditions for the Conduit

The boundary conditions are specified to reflect the fully developed, steady-state flow characteristics within the conduit:

1. **Inlet Condition at** ($\boldsymbol{x = 0}$):

- The velocity profile at the inlet is specified as a parabolic or uniform profile depending on the flow regime.

2. **Outlet Condition at** ($\boldsymbol{x = L}$):

- A zero-gradient condition for the velocity field, indicating that the flow is fully developed.

3. **Top and Bottom Walls (No-Slip Condition at** ($\boldsymbol{y = 0}$) **and** ($\boldsymbol{y = W}$)):

The velocity at the top and bottom walls is zero:

$$\mathbf{u}(x,0) = 0 \quad \text{and} \quad \mathbf{u}(x,W) = 0$$

5.1.3 Flow Classification Based on Reynolds Number

The type of flow within the conduit is classified according to the Reynolds number $((Re))$ as follows:

Laminar Flow: $(\ Re\ <\ 2000\ )$

Transitional Flow: $(\ 2000\ \leq Re\ <\ 4000\ )$

Turbulent Flow: $(\ Re\ \geq 4000\ )$

The Reynolds number is calculated as:

$$Re = \frac{\rho U D}{\mu}$$

where:

$(\ U\ )$ is the average flow velocity.

$(\ D\ )$ is the hydraulic diameter, which for a rectangular conduit is given by:

$$D\ = \frac{2 \cdot y \cdot z}{y\ +\ z}\ =\ 1$$

5.1.4 Flow Rate $(\ Q\ )$ Calculations for Different Flow Regimes

The volumetric flow rate $(\ Q\ )$ for each flow regime can be calculated by rearranging the Reynolds number equation:

$$Q = Re \cdot \nu \cdot y_{max} \cdot z_{max}/D$$

where $(\ \nu)$ is the kinematic viscosity.

5.2 Mathematical Derivation of the Navier-Stokes Equation in Two Dimensions

5.2.1 Derivation of the Navier-Stokes Equation

In this section, we derive the governing Navier-Stokes equations for incompressible, steady-state flow through a two-dimensional rectangular conduit. This derivation is structured as follows:

1. **Conservation of Mass**:

We apply the continuity equation to ensure conservation of mass within the control volume.

2. **Conservation of Momentum**:

We derive the momentum equations by applying Newton's second law to an infinitesimal control volume in the fluid, balancing forces in both $(\ x\ )$ − and $(\ y\ )$-directions.

Assumptions and Constraints

To simplify the derivation, the following assumptions are made:

- **Incompressible Flow**: The fluid density (ρ) is constant, meaning the flow is incompressible.

- **Steady-State**: All properties of the flow are time-independent, simplifying the problem to a steady-state analysis.

- **Fully Developed Flow**: Flow velocity profile does not change along the length of the conduit $((\partial u/\partial x = 0))$.

- **Potential Flow**: Flow is assumed to be irrotational, but this applies mainly in inviscid regions; however, due to the no-slip boundary condition, we relax this assumption near the walls.

- **Two-Dimensional Flow**: Flow is confined to the (x) − (y) plane, with no variation in the (z)-direction.

Conservation of Mass (Continuity Equation)

For incompressible flow, the conservation of mass requires that the net rate of flow into a control volume equals the rate of flow out. This is expressed as:

$$\nabla \cdot \mathbf{u} = 0$$

or in Cartesian coordinates:

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

where (u) and (v) are the velocity components in the (x)- and (y) −directions, respectively.

Conservation of Momentum

The momentum balance is based on Newton's second law applied to an infinitesimal fluid element, where forces due to pressure, viscosity, and external forces are considered.

1. **Momentum Equation in the** ($\boldsymbol{x}$) −**Direction**:

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

3. **Momentum Equation in the** ($\boldsymbol{y}$)**-Direction**:

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

Here:

- (μ) is the dynamic viscosity of the fluid.

- (ρ) is the fluid density.

Final PDE Form for the Navier-Stokes Equations

Combining the continuity equation with the momentum equations in Cartesian coordinates, we obtain the steady-state, incompressible Navier-Stokes equations for 2D flow:

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

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

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

5.3 Neural architecture for the Navier-Stokes modelling

The architecture of the **NS_PINN** model is designed to approximate solutions to the heat equation through a physics-informed neural network (PINN) framework. The model structure includes:

Activation Function: SiLU, which provides smooth nonlinear activation to enhance the learning process.

Layer Composition:

A sequence of linear layers initialized as follows:

```
def positional_encoding(x, num_encodings, enable_encoding=True):
    if not enable_encoding:
        return x
    encodings = [x]
    for i in range(num_encodings):
        factor = min(2  i, 1e3)  # Prevent overflow
        encodings.append(torch.sin(factor * x))
        encodings.append(torch.cos(factor * x))
    return torch.cat(encodings, dim=-1).to(device)

class NS_PINN(nn.Module):
    def __init__(self, layers, num_encodings=1, enable_encoding=True,
activation=nn.SiLU()):
        super(NS_PINN, self).__init__()
        self.num_encodings = num_encodings
        self.enable_encoding = enable_encoding
        self.activation = activation
        self.layers = nn.ModuleList()

        # Input size takes into account the positional encodings
        input_size = layers[0] * (1 + 2 * num_encodings) if
enable_encoding else layers[0]
```

```
        self.layers.append(nn.Linear(input_size, layers[1]).to(device))

        # Hidden layers
        for i in range(1, len(layers) - 1):
            self.layers.append(nn.Linear(layers[i], layers[i +
1]).to(device))

        # Projection layers for skip connections
        self.projection_layers = nn.ModuleList([
            nn.Linear(layers[i], layers[i + 1]).to(device) if layers[i]
!= layers[i + 1] else nn.Identity()
            for i in range(1, len(layers) - 1)
        ])

        # Apply Xavier initialization to all layers
        for layer in self.layers:
            if isinstance(layer, nn.Linear):
                init.xavier_uniform_(layer.weight)
                if layer.bias is not None:
                    layer.bias.data.fill_(0)

        # Apply Xavier initialization to projection layers
        for proj_layer in self.projection_layers:
            if isinstance(proj_layer, nn.Linear):
                init.xavier_uniform_(proj_layer.weight)
                if proj_layer.bias is not None:
                    proj_layer.bias.data.fill_(0)

    def forward(self, x):
        # Move input to device
        x = x.to(device)

        # Apply positional encoding
        x = positional_encoding(x, self.num_encodings,
self.enable_encoding)

        # Forward pass through layers with skip connections
        x = self.activation(self.layers[0](x))
        for i in range(1, len(self.layers) - 1):
            identity = x
            x = self.layers[i](x)
            x = self.activation(x) + self.projection_layers[i-
1](identity)

        # Final layer
        x = self.layers[-1](x)
        return x
```

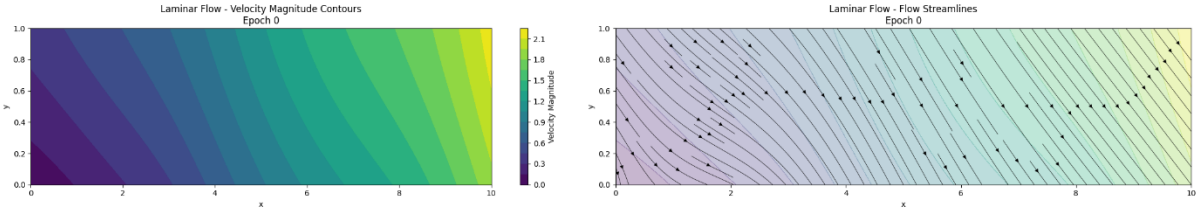
5.4 Flow through Different Regions

Each regime is trained with specified inlet and outlet flow rates (Q_{inlet}) and (Q_{outlet}) and Reynolds numbers (\(Re\)) representative of the flow type. Training is iterated over multiple epochs to minimize the total loss, which combines boundary condition enforcement, volumetric flow rate conservation, and compliance with the Navier-Stokes equations.

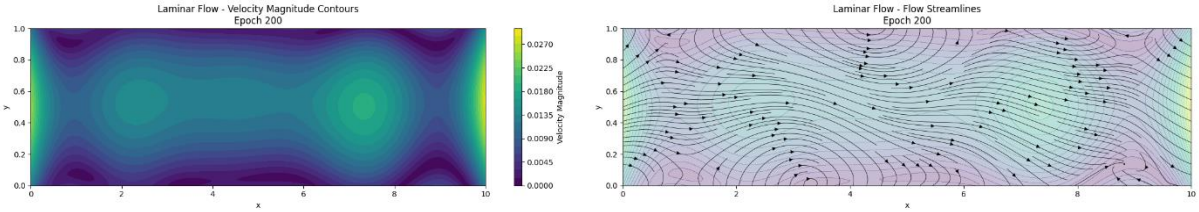
5.4.1 Laminar Region

The laminar region is characterized by smooth, orderly flow where viscous forces dominate, and flow velocity remains consistent along streamlines. The laminar model training targets an inlet and outlet flow rate ($Q_{\text{inlet}} = Q_{\text{outlet}} = 0.001004$) and a Reynolds number \($Re = 999.99$\).

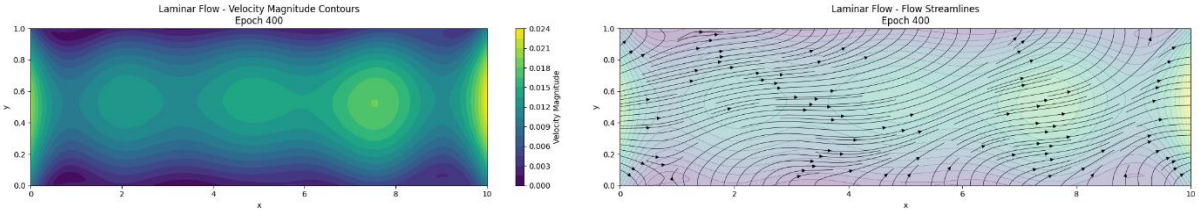
```
Training for Laminar Flow with Target Q_inlet=0.001004 and Target
Q_outlet=0.001004 and Re=999.9999999999999
Epoch 0 - Laminar Flow - Total Loss: 0.003266523549746173
```



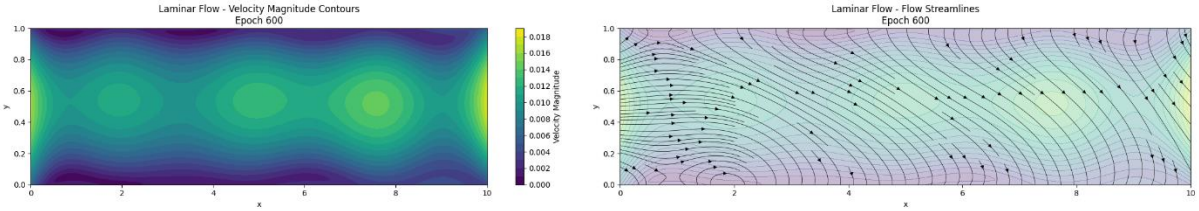

```
Epoch 200 - Laminar Flow - Total Loss: 2.2382496115977826e-06
```



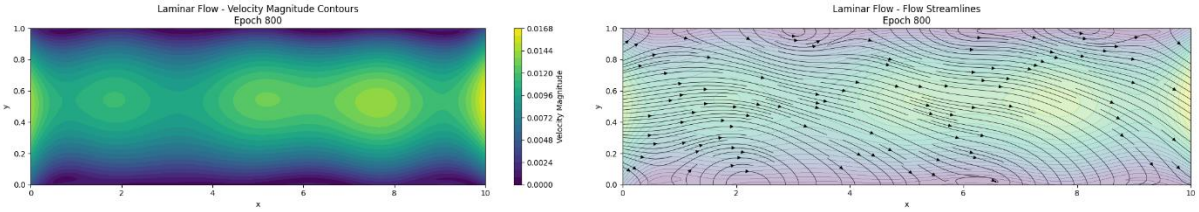

```
Epoch 400 - Laminar Flow - Total Loss: 1.0520601738319402e-06
```



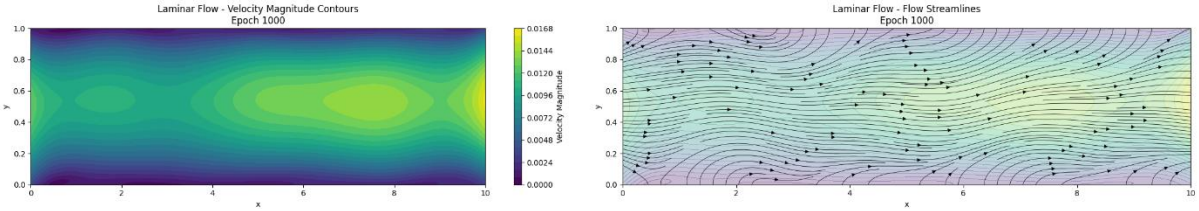

```
Epoch 600 - Laminar Flow - Total Loss: 7.769334039800487e-07
```



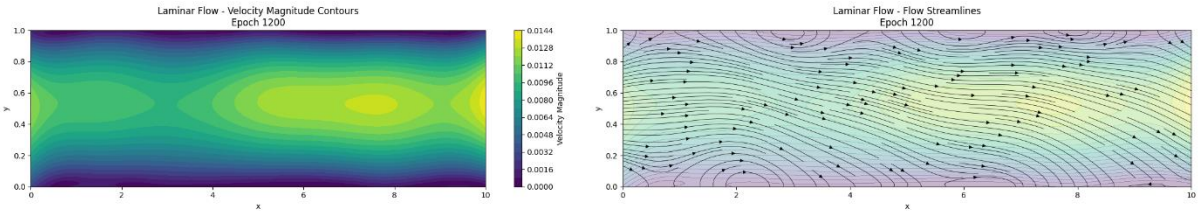

```
Epoch 800 - Laminar Flow - Total Loss: 5.400171719004196e-07
```



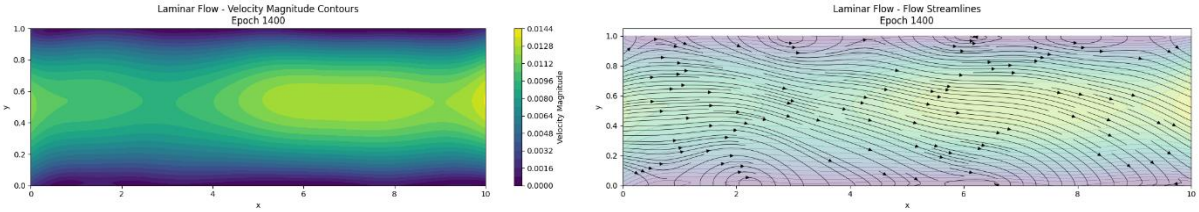

```
Epoch 1000 - Laminar Flow - Total Loss: 3.923663272996888e-07
```



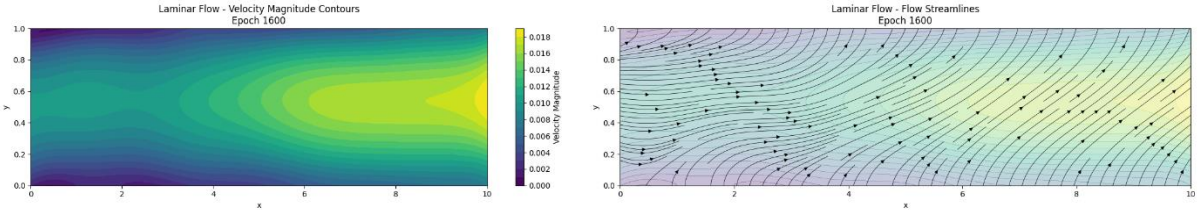

```
Epoch 1200 - Laminar Flow - Total Loss: 3.33707734769462e-07
```



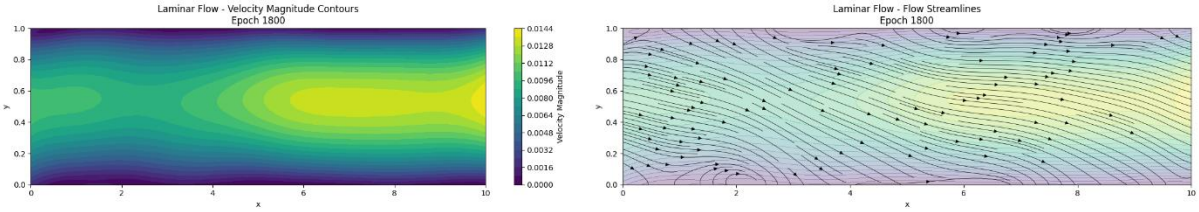

```
Epoch 1400 - Laminar Flow - Total Loss: 3.1791673117576694e-07
```



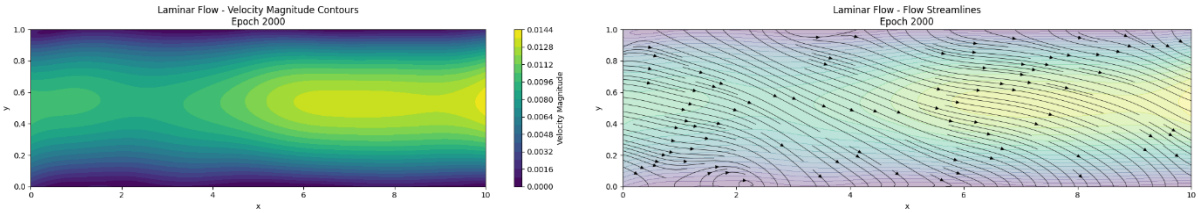

```
Epoch 1600 - Laminar Flow - Total Loss: 2.2111419874311236e-07
```




```
Epoch 1800 - Laminar Flow - Total Loss: 2.6560647500845956e-07
```




```
Epoch 2000 - Laminar Flow - Total Loss: 2.5844559905328167e-07
```

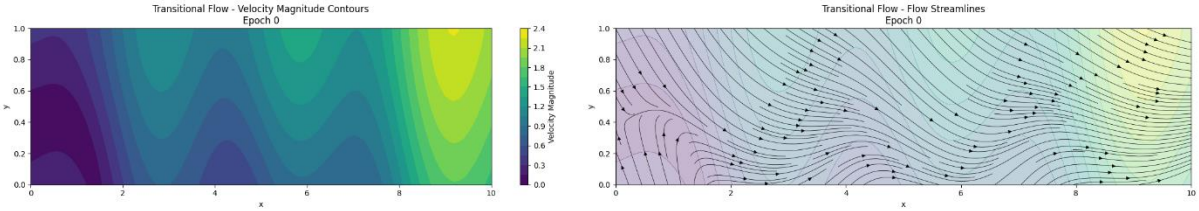



```
Model for Laminar Flow saved.
```

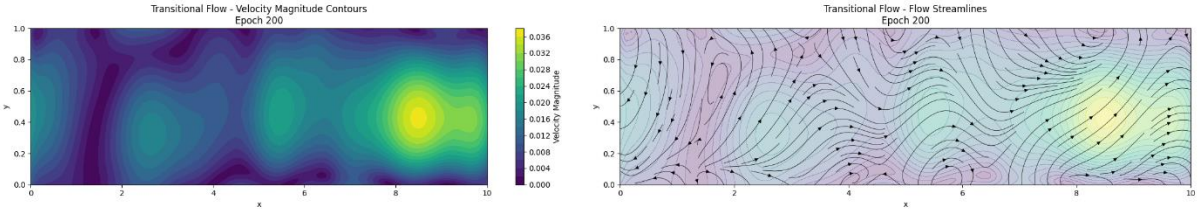
5.4.2 Transitional Region

In the transitional region, the flow begins to exhibit instabilities, marking the shift from laminar to turbulent flow. This region is defined with ($Q_{\text{inlet}} = Q_{\text{outlet}} = 0.002$) and a Reynolds number ($Re \approx 1992.03$), which is within the threshold for transitional behaviour.

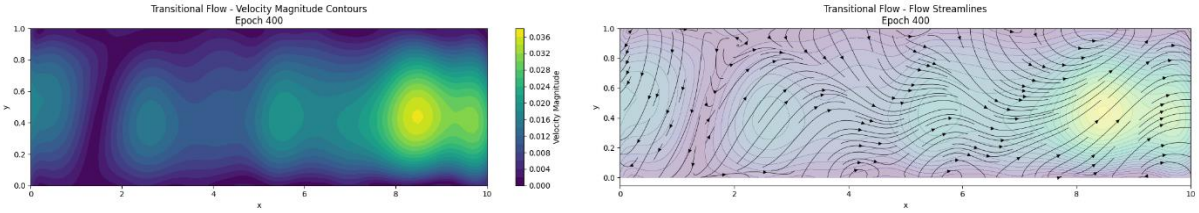
```
Training for Transitional Flow with Target Q_inlet=0.002 and Target
Q_outlet=0.002 and Re=1992.03187250996
Epoch 0 - Transitional Flow - Total Loss: 0.0028180211792190497
```



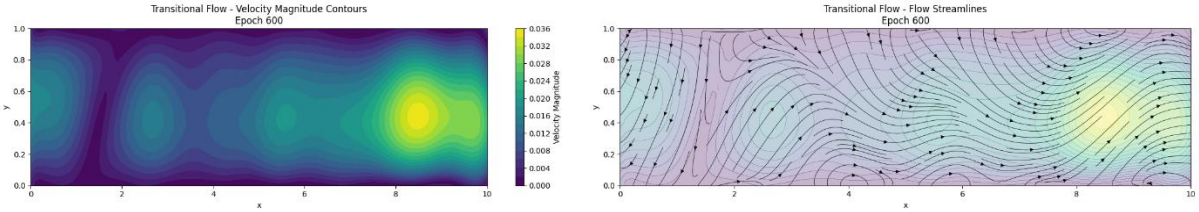
Epoch 200 - Transitional Flow - Total Loss: 3.491022346682472e-06



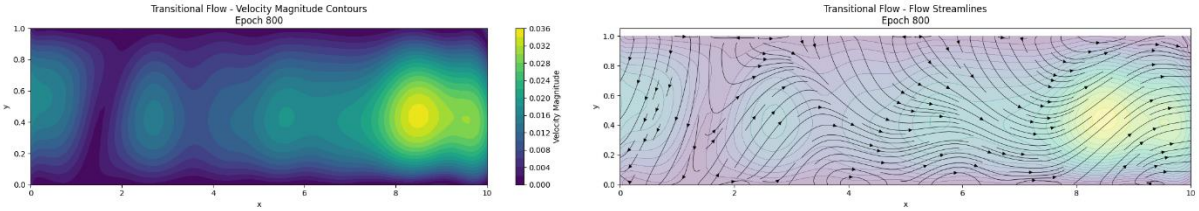
Epoch 400 - Transitional Flow - Total Loss: 2.819239413521027e-06



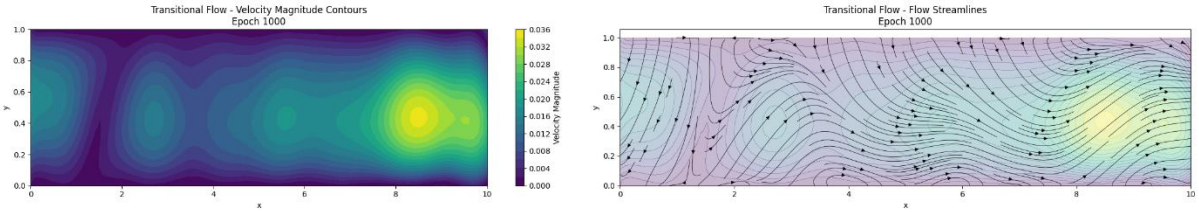
Epoch 600 - Transitional Flow - Total Loss: 2.457185817920826e-06



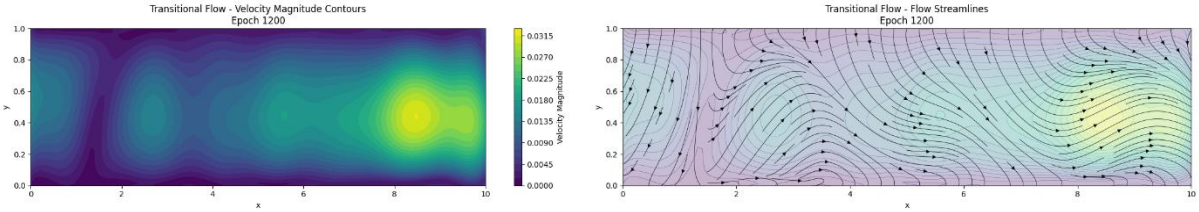
Epoch 800 - Transitional Flow - Total Loss: 2.2666284326447753e-06



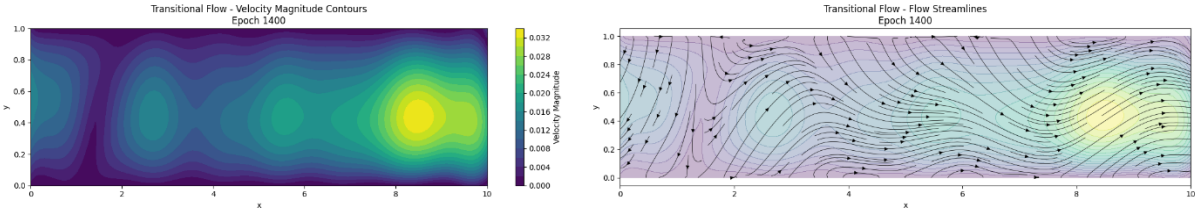
Epoch 1000 - Transitional Flow - Total Loss: 2.1842107355961734e-06



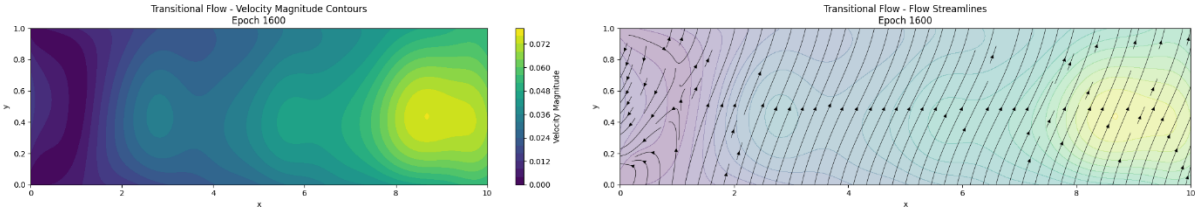
Epoch 1200 - Transitional Flow - Total Loss: 1.8042527352055315e-06



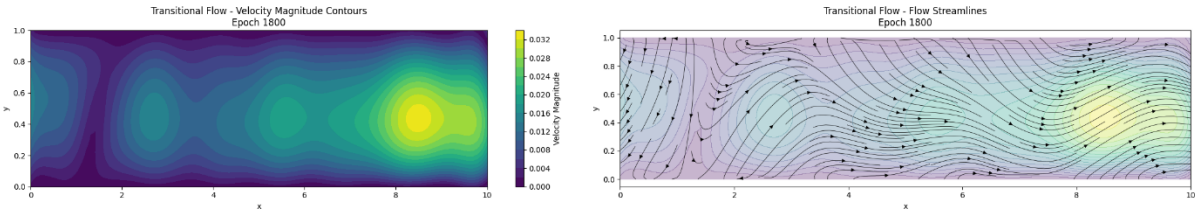
```
Epoch 1400 - Transitional Flow - Total Loss: 1.8410914730341826e-06
```



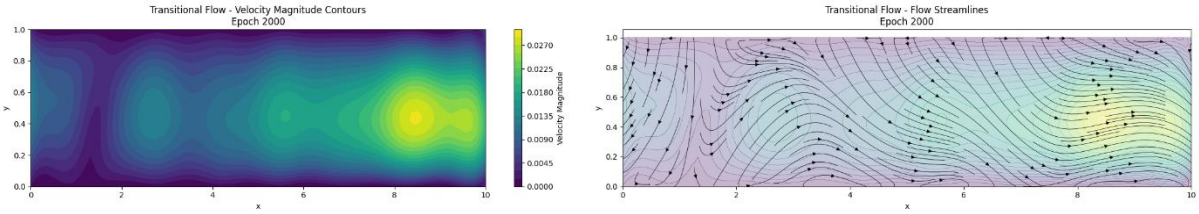

```
Epoch 1600 - Transitional Flow - Total Loss: 4.010929128220217e-06
```




```
Epoch 1800 - Transitional Flow - Total Loss: 1.7115863557389244e-06
```




```
Epoch 2000 - Transitional Flow - Total Loss: 1.287202417515584e-06
```

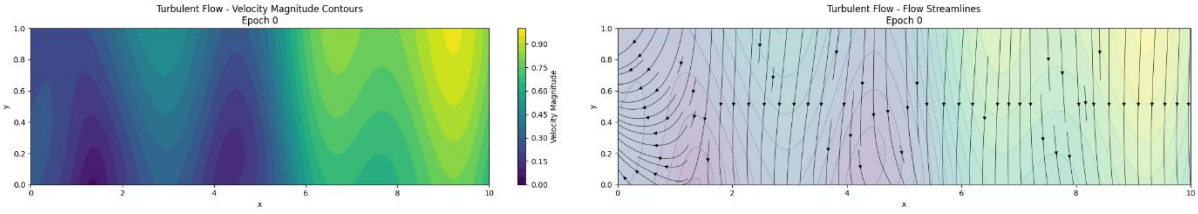



```
Model for Transitional Flow saved.
```

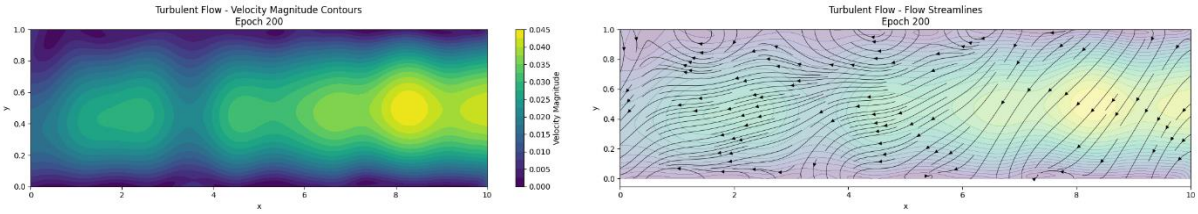
5.4.3 Turbulent Region

Turbulent flow is highly chaotic, with vortices and irregular fluctuations. The model for this regime uses ($Q_{\text{inlet}} = Q_{\text{outlet}} = 0.008$) and a Reynolds number ($Re = 7968.13$), aligning with typical turbulent conditions.

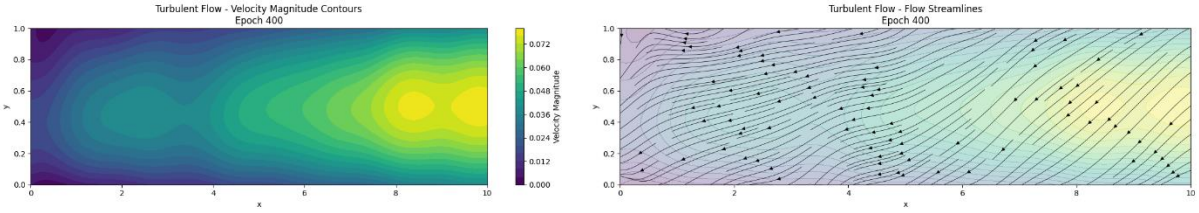
```
Training for Turbulent Flow with Target Q_inlet=0.008 and Target
Q_outlet=0.008 and Re=7968.12749003984
Epoch 0 - Turbulent Flow - Total Loss: 0.002003642711937461
```



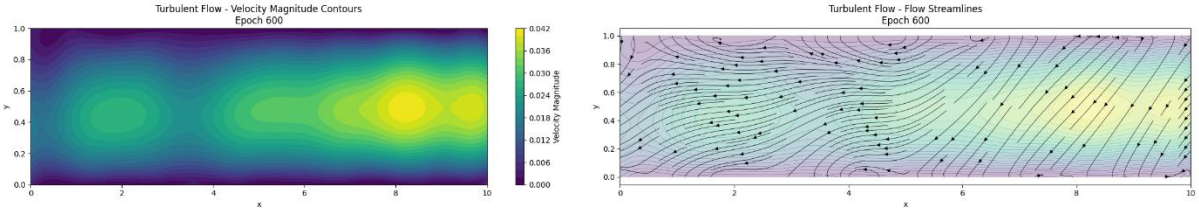

```
Epoch 200 - Turbulent Flow - Total Loss: 4.784299469899042e-06
```



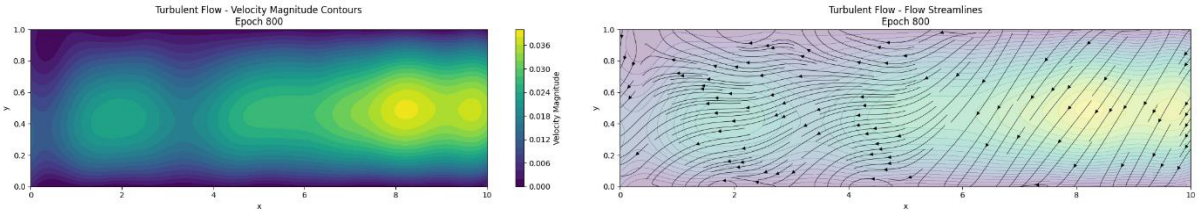
Epoch 400 - Turbulent Flow - Total Loss: 3.7472122897417505e-06



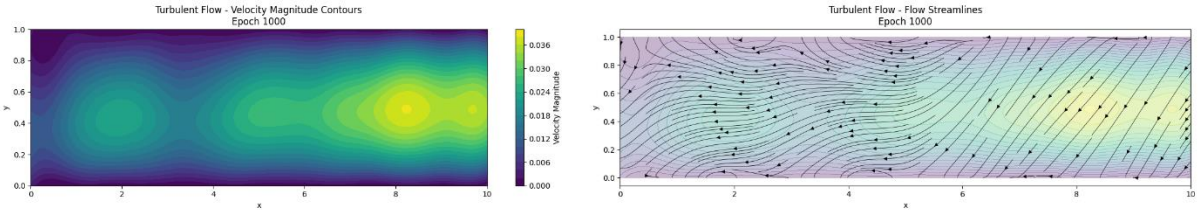
Epoch 600 - Turbulent Flow - Total Loss: 3.7652256572767347e-06



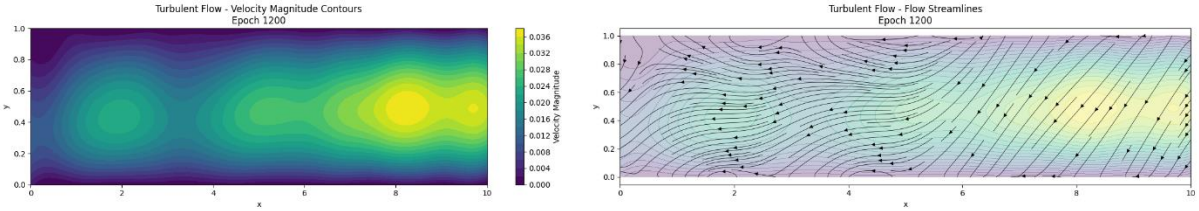
Epoch 800 - Turbulent Flow - Total Loss: 2.6960070461465575e-06



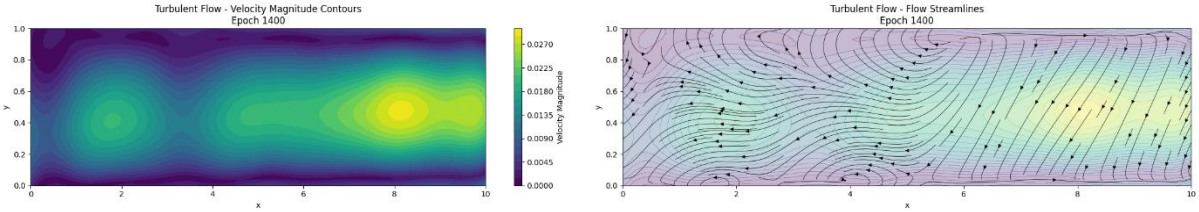
Epoch 1000 - Turbulent Flow - Total Loss: 2.801837983479708e-06



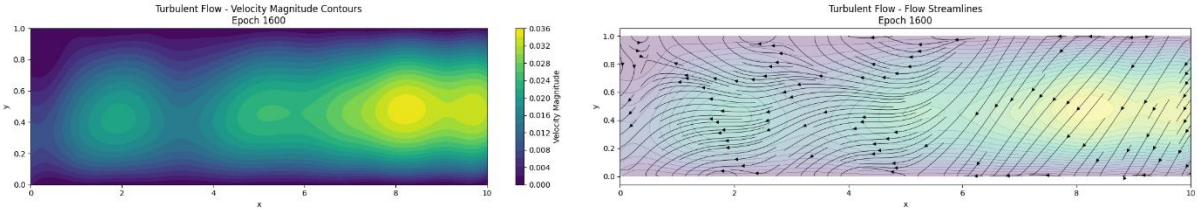
Epoch 1200 - Turbulent Flow - Total Loss: 2.714095385129079e-06



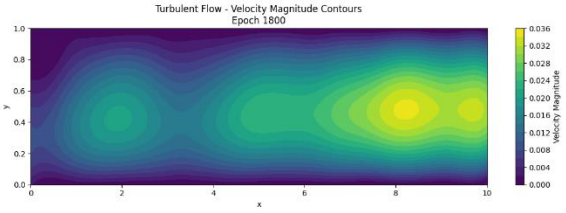
Epoch 1400 - Turbulent Flow - Total Loss: 3.6282509097565433e-06

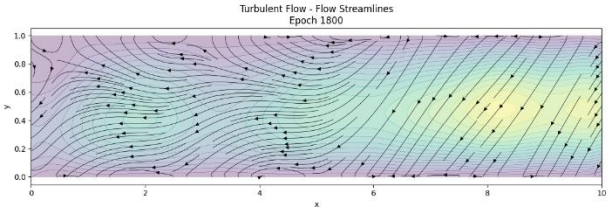


Epoch 1600 - Turbulent Flow - Total Loss: 2.5099234430395343e-06

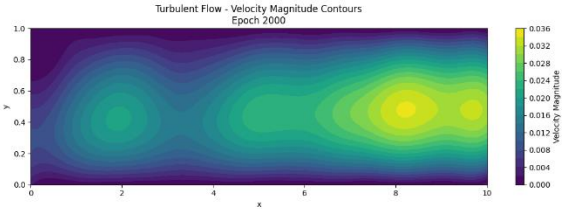


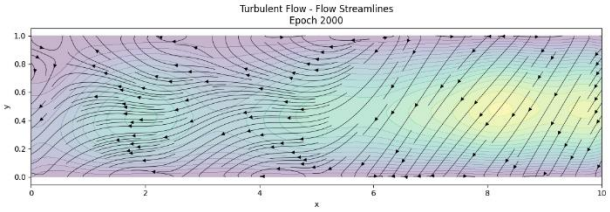
```
Epoch 1800 - Turbulent Flow - Total Loss: 2.6722012926895505e-06
```






```
Epoch 2000 - Turbulent Flow - Total Loss: 2.695307028724448e-06
```






```
Model for Turbulent Flow saved.
```

Each trained model for laminar, transitional, and turbulent flow was saved, preserving the learned characteristics for each flow type. These models can now be used for further analysis and validation of flow behaviour across the different regions within the duct.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this work, we explored the application of Physics-Informed Neural Networks (PINNs) to solve partial differential equations (PDEs) in fluid dynamics and heat transfer, focusing specifically on the Navier-Stokes and heat equations. PINNs offer a promising alternative to traditional numerical methods, as they combine data-driven learning with physics-based modelling, leading to more accurate and efficient solutions for complex engineering problems.

The study progressed by first reviewing the evolution and structure of PINNs, including their fundamental architecture, loss functions, and convergence criteria. This background set the stage for examining how PINNs solve the heat equation and Navier-Stokes equations, showing their adaptability and efficacy in handling problems with complex boundary conditions and nonlinear flow behaviours.

1. **Heat Equation Modelling**: The PINN successfully solved the heat equation under constant and sinusoidal boundary conditions. The model demonstrated rapid convergence with low error margins, illustrating the method's ability to handle varying boundary conditions in thermal systems.

2. **Navier-Stokes Equation Modeling**: For fluid flow in a 2D conduit, the PINN approach was applied to simulate laminar, transitional, and turbulent flow regimes. The training process validated the model's predictions for each flow type, achieving consistent results across various Reynolds numbers and flow rates, with a gradual reduction in total loss across epochs.

Overall, the results demonstrate the versatility of PINNs for solving PDEs across different engineering applications. The model efficiently captured the dynamics of both heat transfer and fluid flow, even under changing boundary conditions and flow regimes, with reasonable accuracy.

6.2 Future Work

Although this research has successfully implemented PINNs for specific cases of the heat and Navier-Stokes equations, there remain several avenues for future work:

1. **Expanding to Three-Dimensional Problems**: Extending the current 2D model to 3D flow fields would enable a more comprehensive study of complex geometries, particularly relevant for real-world applications in chemical and mechanical engineering.

2. **Improving Training Efficiency**: Training PINNs for high-fidelity solutions requires significant computational resources. Future work could investigate more efficient training strategies, such as adaptive learning rates, curriculum learning, or more advanced sampling methods for collocation points, to reduce computation time and enhance convergence.

3. **Integration with Experimental Data**: Incorporating experimental data into the training of PINNs would increase the model's accuracy in cases where the governing equations do not fully capture system dynamics. Hybrid PINNs, which combine measured data with the physics model, could address model inaccuracies due to unknown parameters or simplifying assumptions.

4. **Exploring Alternative Neural Network Architectures**: The performance of PINNs could be improved by investigating other architectures, such as transformer-based models or graph neural networks, which may better capture the spatial dependencies in fluid flow and heat transfer problems.

5. **Generalization to Other Engineering Domains**: The success of PINNs in solving the heat and Navier-Stokes equations suggests that this approach could be extended to other domains in engineering, such as elasticity, electromagnetism, and chemical reaction kinetics. Future research can explore these applications to further validate and broaden the utility of PINNs.

6. **Multi-Physics and Multi-Scale Problems**: Real-world engineering problems often involve multiple interacting physical phenomena across different spatial and temporal scales. Developing multi-physics PINNs that can solve coupled PDEs—such as fluid-structure interactions or reactive flow—would be a valuable extension of this work.

6.3 Closing Remarks

This study underscores the potential of Physics-Informed Neural Networks to revolutionize the way we approach complex engineering simulations. While there remain challenges in scaling and efficiency, the advantages of PINNs for integrating physical laws into machine learning models are clear. As computational techniques and hardware continue to advance, PINNs are likely to become a cornerstone of engineering analysis, offering new possibilities for modelling, optimization, and control in a range of scientific and industrial applications.

References

Cai. (2021). (Wang, 2021)A Review of Physics-Informed Neural Networks for Computational Fluid Dynamics. *Computers & Fluids*.

Chen, R., & Karniadakis, G. (2018). A deep learning framework for solving partial differential equations: Applications in fluid dynamics. *Journal of Computational Physics*(375), 1137-1157.

Raissi, M., Perdikaris, P., & Karniadakis, G. (n.d.). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics, 378*(0021-9991), 686-707. doi:https://doi.org/10.1016/j.jcp.2018.10.045

Toscano, J. D. (2021). From PINNs to PIKANs: Recent Advances in Physics-Informed Machine Learning. *Arxiv*.

Wang, . (2021). Physics-Informed Neural Networks for Chemical Engineering Applications. *AIChE Journal*.