Undergraduate Project Report

Physics Informed Machine Learning for solving Direct and Inverse problems in Heat Transfer

Jatin Bansal

Bachelor of Technology
Department of Mechanical Engineering
Indian Institute of Technology Kanpur
Kanpur, India

Supervisor: Dr. Malay Kumar Das

Table of Contents

List of Figures

List of Tables

Chapter 1

Physics Informed Neural Network

A conventional artificial neural network operates based on the principle of optimizing a loss function, such as the mean squared error derived from the training data and predictions generated by the neural network. The training data is typically acquired from experimental observations confined within a specified bounded interval. These models exhibit suboptimal performance when confronted with test data that deviates from this prescribed interval. The efficiency of the artificial neural network can be substantially enhanced by incorporating an algebraic equation that governs the data within the network. Such a model is known as Physics Informed Neural Network.

In an attempt to solve Partial Differential Equation using an Artificial Neural Network, Raissi et al. [2019], suggested optimizing partial differential equation residual loss, boundary condition loss, and initial condition loss along with the mean squared error obtained from training data and neural network predictions. Let us assume that a PDE is defined by

$$\mathscr{N}(u(x,t)) = 0, \quad x \in \Omega, \quad t \in [0,T] \tag{1.1}$$

$$\mathscr{B}(u(x,t)) = f(t), \quad x \text{ on } \partial\Omega \tag{1.2}$$

$$\mathscr{I}(u(x,0)) = g(x) \tag{1.3}$$

where $\mathscr{N}$ denotes a differential operator acting on u which is a function of x and t, x denotes a d-dimension variable with $\Omega \subset \mathbb{R}^d$ and $\mathscr{B}$ and $\mathscr{I}$ denote boundary condition and initial condition respectively. We model a PINN that takes x and t as input and gives $\hat{u}$ as the output. Then the total loss function is given by

$$\mathscr{L} = \mathscr{L}_{\text{MSE}} + \mathscr{L}_{\text{Physics}} \tag{1.4}$$

where,

$$\mathscr{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{n}(\hat{u}_i - u_i(x,t))^2 \tag{1.5}$$

$$\mathscr{L}_{\text{Physics}} = \frac{1}{N_m}\sum_{i=1}^{N_m}(\mathscr{N}(u(x,t))^2 + \frac{1}{N_b}\sum_{i=1}^{N_b}(\hat{\mathscr{B}}(u(x,t)) - f(t))^2 + \frac{1}{N_i}\sum_{i=1}^{N_i}(\hat{\mathscr{I}}(u(x,t) - g(x))^2 \tag{1.6}$$

where N, N_m, N_b and N_i denote the number of training samples, total points on the meshgrid, boundary points and initial points respectively. This loss function is then optimized using suitable optimizers such as Adams or BFGS. In some cases, we want to prioritize the minimization of a particular component of the total loss while compromising the other component. This can be achieved by using appropriate weights of the corresponding loss components. A loss component which has a higher weight will be minimized to a smaller value compared to the other loss components.

On the basis of calculation of gradients, PINNs can be differentiated into two categories - a-PINN which works on the principle that output of a PINN is a function of its inputs, thus calculating the gradients analytically, and n-PINN, which uses numerical approximation for the differential operators.

The true potential of PINNs can be realized in the cases where the number of training samples is very less or is non-existent. PINNs are so powerful that it can be used to solve PDEs without having much training samples.

Chapter 2

Using PINN for solving PDE

Most of the engineering disciplines, especially processes related to heat transfer, are governed by mathematical equations that can be solved by PINNs with minimal error. In this section, I have presented the solutions of some standard equations using PINNs.

2.0.1 Steady State One Dimensional Heat Diffusion Equation with constant thermal diffusivity

Let us assume that there exists a rod of length L = 1 unit made of a material having thermal diffusivity α. In this case, the heat Diffusion Equation in 1 dimension is given by:

$$\frac{\partial^2 T}{\partial x^2} = 0, x \in [0,1] \tag{2.1}$$

$$T(0) = 0 \tag{2.2}$$

$$T(1) = 10 \tag{2.3}$$

where T denotes the temperature which is a function of independent variable x.

Solution of the Forward Problem

The forward problem can be solved using a PINN which takes x as input and predicts the Temperature $\hat{T}$. As we do not have any training samples, the total loss is just the physics loss and is given by the following expression:

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{physics}} = \frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{\partial^2 \hat{T}}{\partial x^2})^2 + (\hat{T}(0)-0)^2 + (\hat{T}(1)-10)^2 \tag{2.4}$$

where N_m represents the number of points on which we are solving the PDE, which in our case is 100. The PINN has 10 hidden layers with each layer having 40 nodes with $tanh$ activation function. The loss function is minimized using Adam optimizer having learning rate 10^{-4}. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 49.1 to 4.6 x 10^{-5} after training for 10^5 epochs. The plot of total loss vs epoch and temperature distribution v/s the independent variable x are shown in Figure A.1 and Figure A.2 respectively.

Solution of the Inverse Problem

Solving the inverse problem aims to determine the boundary conditions or thermal properties of materials by using some training samples obtained either by experiments or by taking some of the data points from the solution of the forward problem. The inverse problem can be modeled as:

$$\frac{\partial^2 T}{\partial x^2} = 0, x \in [0,1] \tag{2.5}$$

$$T(0) = 0 \tag{2.6}$$

$$T(1) = T_0 \tag{2.7}$$

where T_0 is an unknown which represents the temperature at $x = 1$. We are also given the values of Temperature at 11 points uniformly spaced in the interval $x \in [0.1, 0.9]$. The total loss function is given by,

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{MSE}} + \mathscr{L}_{\text{Physics}} \tag{2.8}$$

where,

$$\mathscr{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{n}(\hat{T}_i - T_i(x))^2 \tag{2.9}$$

$$\mathscr{L}_{\text{physics}} = \frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{\partial^2 \hat{T}}{\partial x^2})^2 + (\hat{T}(0) - 0)^2 + (\hat{T}(1) - T_0)^2 \tag{2.10}$$

where N and N_m represent the number of training data samples and the number of points on which we are solving the PDE. The Inverse PINN has 10 hidden layers with each layer having 40 nodes with *tanh* activation function. The Inverse PINN takes x as input and outputs the Temperature $\hat{T}$ while updating T_0 simultaneously as the weights of the neural network are updated, but with a different learning rate. The learning rate for the weights of artificial neural network and T_0 is 10^{-4} and 2.5 x 10^{-2} respectively. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 933.45 to 0.011 after training for 10^5 epochs. T_0 was intitialised to be 30 units, but after training it came out to be 9.96078 units against the expected value of 10 units.

Python code for the forward and the inverse solution of Steady State One Dimensional Heat Diffusion Equation with constant thermal diffusivity can be found at this link.

2.0.2 Steady State One Dimensional Heat Diffusion Equation with variable thermal diffusivity

Let us assume that there exists a rod of length $L = 1$ unit made of a material having thermal diffusivity α. In this case, the heat Diffusion Equation in 1 dimension is given by:

$$\frac{\partial}{\partial x}(\alpha\frac{\partial T}{\partial x}) = 0, x \in [0,1] \tag{2.11}$$

$$\alpha = \alpha_0 + \alpha_1 T \tag{2.12}$$

$$T(0) = 0 \tag{2.13}$$

$$T(1) = 10 \tag{2.14}$$

where T denotes the temperature which is a function of independent variable x and α_0 and α_1 are two known parameters.

Solution of the Forward Problem

The forward problem can be solved using a PINN which takes x as input and predicts the Temperature $\hat{T}$. As we do not have any training samples, total loss is just the physics loss and is given by the following expression:

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{physics}} = \frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{\partial^2\hat{T}}{\partial x^2})^2 + (\hat{T}(0) - 0)^2 + (\hat{T}(1) - 10)^2 \tag{2.15}$$

where N_m represents the number of points on which we are solving the PDE, which in our case is 100. In setting up this problem, $\alpha_0 = 1$ and $\alpha_1 = 0.1$. The PINN has 10 hidden layers with each layer having 40 nodes with *tanh* activation function. The loss function is minimized using Adam optimizer having learning rate 2x10^{-4}. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 49.1 to 2x10^{-5} after training for 10^5 epochs. The plot of total loss vs epoch and temperature distribution v/s the independent variable x are shown in Figure A.3 and Figure A.4 respectively.

Solution of the Inverse Problem

Solving the inverse problem aims to determine the boundary conditions or thermal properties of materials by using some training samples obtained either by experiments or by taking some of the data points from the solution of the forward problem. The inverse problem can be modeled as:

$$\frac{\partial}{\partial x}(\alpha\frac{\partial T}{\partial x}) = 0, x \in [0,1] \tag{2.16}$$

$$\alpha = \alpha_0 + \alpha_1 T \tag{2.17}$$

$$T(0) = T_0 \tag{2.18}$$

$$T(1) = T_1 \tag{2.19}$$

where T_0 and T_1 are unknown quantities which represent the temperature at $x = 0$ and $x = 1$ respectively. We are also given the values of Temperature at 11 points uniformly spaced in the interval $x \in [0.1, 0.9]$. The total loss function is given by,

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{MSE}} + \mathscr{L}_{\text{Physics}} \tag{2.20}$$

where,

$$\mathscr{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{n}(\hat{T}_i - T_i(x))^2 \tag{2.21}$$

$$\mathscr{L}_{\text{physics}} = \frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{\partial^2 \hat{T}}{\partial x^2})^2 + (\hat{T}(0) - T_0)^2 + (\hat{T}(1) - T_1)^2 \tag{2.22}$$

where N and N_m represent the number of training data samples and the number of points on which we are solving the PDE. In setting up this problem, $\alpha_0 = 1$ and $\alpha_1 = 0.1$. The Inverse PINN has 10 hidden layers with each layer having 40 nodes with $tanh$ activation function. The Inverse PINN takes x as input and outputs the Temperature $\hat{T}$ while updating T_0 and T_1 simultaneously as the weights of the neural network are updated, but with a different learning rate. The learning rate for the weights of artificial neural network and boundary Temperatures is 10^{-4} and 5x10^{-2} respectively. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 1166.41 to 0.0104 after training for 10^5 epochs. T_0 and T_1 was initialised to be 15 and 30 units respectively, but after training it came out to be 0.01925 and 10.04667 units against the expected value of 0 and 10 units respectively.

Python code for the forward and the inverse solution of Steady State One Dimensional Heat Diffusion Equation with variable thermal diffusivity can be found at this link.

2.0.3 Unsteady One Dimensional Heat Diffusion Equation

Let us assume that there exists a rod of length $L = 1$ unit made of a material having constant thermal diffusivity α. In this case, the heat Diffusion Equation in 1 dimension is given by:

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = 0, x \in [0,1], t \in [0,5] \tag{2.23}$$

$$T(0,t) = 0 \tag{2.24}$$

$$T(1,t) = 0 \tag{2.25}$$

$$T(x,0) = \sin(\pi x) \tag{2.26}$$

where T denotes the temperature which is a function of spatial variable x and time t.

Solution of the Forward Problem

The forward problem can be solved using a PINN which takes x and t as input and predicts the Temperature $\hat{T}$. We will solve the problem on a mesh grid x and t of size 100x10. As we do not have any training samples, the total loss is just the physics loss and is given by the following expression:

$$\begin{aligned} \mathscr{L}_{\text{physics}} = & \frac{1}{N_m} \sum_{i=1}^{N_m} \left(\frac{\partial \hat{T}}{\partial t} - \alpha \frac{\partial^2 \hat{T}}{\partial x^2} \right)^2 + \frac{1}{N_{b1}} \sum_{i=1}^{N_{b1}} (\hat{T}(0,t_i) - 0)^2 \\ & + \frac{1}{N_{b2}} \sum_{i=1}^{N_{b2}} (\hat{T}(1,t_i) - 0)^2 + \frac{1}{N_0} \sum_{i=1}^{N_0} (\hat{T}(x_i,0) - \sin(\pi x))^2 \end{aligned} \tag{2.27}$$

where N_m, N_{b1}, N_{b2}, N_0 represents the number of points on the mesh grid, number of points on the boundary $x = 0$, number of points on the boundary $x = 1$ and number of points where $t = 0$ respectively. In this case, N_m, N_{b1}, N_{b2}, N_0 is $1000, 10, 10, 1$ respectively. The PINN has 10 hidden layers with each layer having 40 nodes with *tanh* activation function. The loss function is minimized using Adam optimizer having learning rate 10^{-4}. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 0.62 to 1.7x10^{-6} after training for 10^5 epochs. The plot of total loss vs epoch, contour plot of temperature v/s the spatial variable x and time t and plot of Temperature T v/s spatial variable x for different values of time t are shown in Figure A.5, Figure A.6 and Figure A.7 respectively.

Solution of the Inverse Problem

Solving the inverse problem aims to determine the boundary conditions or thermal properties of materials by using some training samples obtained either by experiments or by taking some of the data points from the solution of the forward problem. The inverse problem can be modeled as:

$$\frac{\partial T}{\partial t} - \alpha \frac{\partial^2 T}{\partial x^2} = 0, x \in [0,1], t \in [0,5] \tag{2.28}$$

$$T(0,t) = T_0 \tag{2.29}$$

$$T(1,t) = T_1 \tag{2.30}$$

$$T(x,0) = \sin(\pi x) \tag{2.31}$$

where T_0 and T_1 are unknown quantities which represent the temperature at $x = 0$ and $x = 1$ respectively. We are also given the values of Temperature at 12 points lying on 4x3 mesh grid formed by $x = [0.2, 0.4, 0.6, 0.8]$ and $t = [2.0, 3.0, 4.0]$. The training samples have zero mean Gaussian noise with standard deviation $\sigma = 0.1$. The total loss function is given by,

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{MSE}} + \mathscr{L}_{\text{Physics}} \tag{2.32}$$

where,

$$\mathscr{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{n} (\hat{T}_i - T_i(x))^2 \tag{2.33}$$

$$\begin{aligned} \mathscr{L}_{\text{physics}} = & \frac{1}{N_m} \sum_{i=1}^{N_m} \left(\frac{\partial \hat{T}}{\partial t} - \alpha \frac{\partial^2 \hat{T}}{\partial x^2} \right)^2 + \frac{1}{N_{b1}} \sum_{i=1}^{N_{b1}} (\hat{T}(0,t_i) - T_1)^2 \\ & + \frac{1}{N_{b2}} \sum_{i=1}^{N_{b2}} (\hat{T}(1,t_i) - T_0)^2 + \frac{1}{N_0} \sum_{i=1}^{N_0} (\hat{T}(x_i,0) - \sin(\pi x))^2 \end{aligned} \tag{2.34}$$

where N represents the number of training data samples. In setting up this problem, $\alpha = 0.1$. The Inverse PINN has 10 hidden layers with each layer having 40 nodes with *tanh* activation function. The Inverse PINN takes x and t as input and outputs the Temperature $\hat{T}$ while updating T_0 and T_1 simultaneously as the weights of the neural network are updated, but with a different learning rate. The learning rate for the weights of artificial neural network and boundary Temperatures is 2x10^{-4} and 5x10^{-2} respectively. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 1014.53 to 0.005 after training for5x10^{4} epochs. T_0 and T_1 were initialised to be 10 and 30 units respectively, but after training it came out to be 0.082424 and 0.05334 units against the expected value of 0 and 0 units respectively.

Python code for the forward and the inverse solution of unsteady One Dimensional Heat Diffusion Equation can be found at this link.

The inverse problem can also be modeled in another way in which we know the boundary temperatures but we have to find the value of thermal properties of material, say α. In setting up this problem, $T_0 = T_1 = 0$. The Inverse PINN has 10 hidden layers with each layer having

40 nodes with $tanh$ activation function. The Inverse PINN takes x and t as input and outputs the Temperature $\hat{T}$ while updating α simultaneously as the weights of the neural network are updated, but with a different learning rate. The learning rate for the weights of artificial neural network and thermal diffusivity α is 2x10^{-4} and 10^{-2} respectively. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 0.84 to 0.0011 after training for 10^5 epochs. α was initialised to be 1 unit but after training it came out to be 0.1101 units against the expected value of 0.1 units.

2.0.4 Steady Two Dimensional Heat Diffusion Equation with constant thermal diffusivity

Let us assume that there exists a square plate of side a = 1 unit made of a material having constant thermal diffusivity α. In this case, the steady heat Diffusion Equation in 2 dimensions is given by:

$$\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}=0, x\in[0,1], y\in[0,1] \tag{2.35}$$

$$T(0,y)=10y \tag{2.36}$$

$$T(1,y)=10\sin(\frac{\pi y}{2}) \tag{2.37}$$

$$T(x,0)=0 \tag{2.38}$$

$$T(x,1)=10 \tag{2.39}$$

where T denotes the temperature which is a function of spatial variable x and y.

Solution of the Forward Problem

The forward problem can be solved using a PINN which takes x and y as inputs and predicts the Temperature $\hat{T}$. We will solve the problem on a mesh grid x and y of size 20x20. As we do not have any training samples, the total loss is just the physics loss and is given by the following expression:

$$\begin{aligned}\mathscr{L}_{\text{total}} &= \frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{\partial^2 \hat{T}}{\partial x^2}+\frac{\partial^2 \hat{T}}{\partial y^2})^2+\frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{T}(0,y_i)-10y)^2+\frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\hat{T}(1,y_i)-10\sin(\frac{\pi y}{2}))^2\\ &+\frac{1}{N_0}\sum_{i=1}^{N_{b3}}(\hat{T}(x_i,0)-0)^2+\frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\hat{T}(x_i,1)-10)^2\end{aligned} \tag{2.40}$$

where N_m, N_{b1}, N_{b2}, N_{b3} and N_{b4} represents the number of points on the mesh grid, number of points on the boundary $x=0$, number of points on the boundary $x=1$, number of points on the boundary $y=0$ and number of points on the boundary $y=1$ respectively. In this case, N_m, N_{b1}, N_{b2}, N_{b3} and N_{b4} is $100, 20, 20, 20, 20$ respectively. The PINN has 10 hidden layers with each layer having 40 nodes with *tanh* activation function. The loss function is minimized using Adam optimizer having learning rate 10^{-4}. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 187.97 to 6x10^{-3} after training for 10^5 epochs. The plot of total loss vs epoch and the contour plot of temperature v/s the spatial variables x and y are shown in Figure A.8 and Figure A.9 respectively.

Python code for the forward solution of Steady Two Dimensional Heat Diffusion Equation can be found at this link.

2.0.5 Blasius Equation

Blasius Equation is given by:

$$\frac{1}{2}f\frac{d^2f}{d\eta^2}+\frac{d^3f}{d\eta^3}=0, \eta \in [0,5] \tag{2.41}$$

$$f(0)=0 \tag{2.42}$$

$$\frac{df(0)}{d\eta}=0 \tag{2.43}$$

$$\frac{df(5)}{d\eta}=1 \tag{2.44}$$

where f is a function of spatial variable η.

Solution of the Forward Problem

The forward problem can be solved using a PINN which takes η as input and predicts $\hat{f}$. We will solve the problem on a for 200 equally spaced values of $\eta \in [0,5]$. As we do not have any training samples, the total loss is just the physics loss and is given by the following expression:

$$\mathscr{L}_{\text{total}}=\frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{1}{2}\hat{f}\frac{d^2\hat{f}}{d\eta^2}+\frac{d^3\hat{f}}{d\eta^3})^2+(\hat{f}(0))^2+(\frac{d\hat{f}(0)}{d\eta})^2+(\frac{d\hat{f}(5)}{d\eta}-1)^2 \tag{2.45}$$

where N_m represents the number of points on the mesh grid. In this case, N_m is 201. The PINN has 10 hidden layers with each layer having 40 nodes with *tanh* activation function. The loss function is minimized using Adam optimizer having learning rate 10^{-4}. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 1.008 to 7x10^{-7} after training for 15000 epochs. The plot of total loss vs epoch and f, $\frac{d\hat{f}}{d\eta}$ and $\frac{d\hat{f}^2}{d\eta^2}$ v/s the spatial variable η are shown in Figure A.10 and Figure A.11 respectively.

This model is also validated using experimental data which can be found at this link. The validation plot is shown in Figure A.12.

Solution of Blasius equation can be used to find the dimensionless temperature θ by using the following equation:

$$\frac{1}{2}Prf\frac{d\theta}{d\eta}+\frac{d^2\theta}{d\eta^2}=0, \eta \in [0,5] \tag{2.46}$$

$$\theta(0)=0 \tag{2.47}$$

$$\theta(5)=1 \tag{2.48}$$

where Pr is a constant and θ is a function of spatial variable η.

This problem is solved by using a PINN which takes $\hat{f}$ and η as inputs and predicts $\hat{\theta}$ for 200 equally spaced values of $\eta \in [0,5]$. As we do not have any training samples, the total loss is just the physics loss and is given by the following expression:

$$\mathscr{L}_{\text{total}} = \frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{1}{2}Prf\frac{d\hat{\theta}}{d\eta} + \frac{d^2\hat{\theta}}{d\eta^2})^2 + (\hat{\theta}(0))^2 + (\hat{\theta}(5)-1)^2 \tag{2.49}$$

The PINN has 10 hidden layers with each layer having 40 nodes with *tanh* activation function. While solving the problem, Prandtl Number is taken to be 0.7. The loss function is minimized using Adam optimizer having learning rate 10^{-4}. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 0.83 to 3x10^{-4} after training for 1000 epochs. The plot of total loss vs epoch and θ and $\frac{d\hat{\theta}}{d\eta}$ v/s the spatial variable η are shown in Figure A.13 and Figure A.14 respectively.
This model is also validated using experimental data which can be found at this link. The validation plot is shown in Figure A.15.

Python code for the forward solution of Blasius equation can be found at this link.

Solution of the Inverse Problem

The inverse problem can be modeled as:

$$\frac{1}{2}Prf\frac{d\theta}{d\eta} + \frac{d^2\theta}{d\eta^2} = 0, \eta \in [0,5] \tag{2.50}$$

$$\theta(0) = 0 \tag{2.51}$$

$$\theta(5) = 1 \tag{2.52}$$

where *Pr* is an unknown constant. Unlike all the inverse problems that have been solved in this text, this inverse problem is solved by using experimental data present at this link and not the predictions of the forward PINN. We have taken the value of η, f, θ and $\frac{d\theta}{d\eta}$ from the file. The total loss function with regularization term is given by,

$$\begin{aligned}\mathscr{L}_{\text{total}} = &\frac{1}{N}\sum_{i=1}^{n}(\hat{\theta}_i - \theta_i(x))^2 + \frac{1}{N}\sum_{i=1}^{n}(\frac{d\hat{\theta}_i}{d\eta} - \frac{d\theta_i}{d\eta})^2 + \frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{1}{2}Prf\frac{d\hat{\theta}}{d\eta} + \frac{d^2\hat{\theta}}{d\eta^2})^2 + \\ &+ (\hat{\theta}(0))^2 + (\hat{\theta}(5)-1)^2 - N(3\sigma)^2\end{aligned} \tag{2.53}$$

where N represents the number of training data samples and σ represent zero mean Gaussian noise that has been added to the data. The Inverse PINN has 10 hidden layers with each layer having 40 nodes with *tanh* activation function. The Inverse PINN takes η and t as input and outputs the Temperature $\hat{\theta}$ while updating *Pr* simultaneously as the weights of the neural network are updated, but with a different learning rate. The learning rate for the weights of artificial neural network and Pr is 1x10^{-4} and 2.5x10^{-2} respectively. The predicted values of Prandtl Number Pr for different number of training samples N and values of standard deviation σ are given in Table B.1.

Python code for the inverse solution of Blasius equation can be found at this link.

2.0.6 Flow over flat plate problem

Let us assume that there is a flat plate having infinite length and width of 0.2 units. The fluid has viscosity $\nu = 0.001$. In this case, the equations governing the flow without energy consideration are given by:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0, x\in[0,1], y\in[0,0.2] \tag{2.54}$$

$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=\nu\frac{\partial^2 u}{\partial y^2} \tag{2.55}$$

$$u(0,y)=1, v(0,y)=0 \tag{2.56}$$

$$u(x,0)=0, v(x,0)=0, \tag{2.57}$$

$$u(x,0.2)=1, \left.\frac{\partial v}{\partial y}\right|_{y=0.2}=0 \tag{2.58}$$

where u and v denote the velocity in the x and y direction respectively.

Solution of the Forward Problem

The forward problem can be solved using a PINN which takes x and y as inputs and predicts the velocities $\hat{u}$ and $\hat{v}$. We will solve the problem on a mesh grid x and y of size 40x40. As we do not have any training samples, the total loss is just the physics loss and is given by the following expression:

$$\begin{aligned}\mathscr{L}_{\text{total}} &= \frac{1}{N_m}\sum_{i=1}^{N_m}\left(\frac{\partial \hat{u}}{\partial x}+\frac{\partial \hat{v}}{\partial y}\right)^2+\frac{1}{N_m}\sum_{i=1}^{N_m}\left(\hat{u}\frac{\partial \hat{u}}{\partial x}+\hat{v}\frac{\partial \hat{u}}{\partial y}-\nu\frac{\partial^2 \hat{u}}{\partial y^2}\right)^2+\frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{u}(0,y_i)-1)^2\\ &+\frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{v}(0,y_i)-0)^2+\frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\hat{u}(x_i,0)-0)^2+\frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\hat{v}(x_i,0)-0)^2\\ &+\frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{u}(x_i,0.2)-1)^2+\frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}\left(\left.\frac{\partial \hat{v}}{\partial y}\right|_{y=0.2}-0\right)^2\end{aligned} \tag{2.59}$$

where N_m, N_{b1}, N_{b2} and N_{b3} represent the number of points on the mesh grid, number of points on the boundary $x = 0$, number of points on the boundary $y = 0$ and number of points on the boundary $y = 1$ respectively. In this case, N_m, N_{b1}, N_{b2} and N_{b3} is 1600, 40, 40 and 40 respectively. The PINN has 10 hidden layers with each layer having 40 nodes with $tanh$ activation function. The loss function is minimized using Adam optimizer having learning rate 10^{-4}. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 2.07 to 1.4x10^{-2} after training for 25x10^3 epochs. The plot of total loss vs epoch, the contour plot of velocity in x direction v/s the spatial variables x and y and the contour plot of velocity in y direction v/s the spatial variables x and $y$are shown in Figure A.16, Figure A.17 and Figure A.18 respectively.

Plot of velocity profiles at various values of x are given in Figure A.19 and Figure A.20. Plot of Skin Friction vs x is given in Figure A.21

The model has been trained on a meshgrid of x and y. Corresponding values of η can be calculated by using the relation given below:

$$\eta = y(\frac{U}{\nu x})^{\frac{1}{2}} \tag{2.60}$$

This model is also validated using experimental data which can be found at this link. The validation plot is shown in Figure A.22.

Solution of the Inverse Problem

The inverse problem can be modeled as:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, x \in [0,1], y \in [0,0.2] \tag{2.61}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\frac{\partial^2 u}{\partial y^2} \tag{2.62}$$

$$u(0,y) = 1, v(0,y) = 0 \tag{2.63}$$

$$u(x,0) = 0, v(x,0) = 0, \tag{2.64}$$

$$u(x,0.2) = 1, \left.\frac{\partial v}{\partial y}\right|_{y=0.2} = 0 \tag{2.65}$$

where ν is an unknown variable. We are also given the values of u and v at 30 points lying on 5x6 mesh grid formed by $x = [0.1, 0.2, 0.4, 0.6, 0.8]$ and $t = [0.02, 0.06, 0.08, 0.12, 0.16, 0.2]$. The training samples have zero mean Gaussian noise with standard deviation $\sigma = 0.01$. The total loss function is given by,

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{MSE}} + \mathscr{L}_{\text{Physics}} \tag{2.66}$$

where,

$$\mathscr{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{n}(\hat{u}_i - u_i)^2 + \frac{1}{N}\sum_{i=1}^{n}(\hat{v}_i - v_i)^2 \tag{2.67}$$

$$\begin{aligned}
\mathscr{L}_{\text{physics}} = &\frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{\partial \hat{u}}{\partial x} + \frac{\partial \hat{v}}{\partial y})^2 + \frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{u}}{\partial x} + \hat{v}\frac{\partial \hat{u}}{\partial y} - \nu\frac{\partial^2 \hat{u}}{\partial y^2})^2 + \frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{u}(0,y_i) - 1)^2 \\
&+ \frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{v}(0,y_i) - 0)^2 + \frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\hat{u}(x_i,0) - 0)^2 + \frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\hat{v}(x_i,0) - 0)^2 \\
&+ \frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{u}(x_i,0.2) - 1)^2 + \frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\frac{\partial \hat{v}}{\partial y}\bigg|_{y=0.2} - 0)^2
\end{aligned} \tag{2.68}$$

where N represents the number of training data samples. The Inverse PINN has 10 hidden layers with each layer having 40 nodes with *tanh* activation function. The Inverse PINN takes x and y as input and outputs the velocities $\hat{u}$ and $\hat{v}$ while updating ν simultaneously as the weights of the neural network are updated, but with a different learning rate. The learning rate for the weights of artificial neural network and ν is 2x10^{-4} and 1x10^{-5} respectively. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 2.98 to 0.014 after training for2.5x10^{4} epochs. ν was initialised to be 0.0005 units, but after training it came out to be 0.001097 units against the expected value of 0.001 units.

Python code for the solution of Flow over flat plate Problem can be found at this link.

2.0.7 Convective flow problem

Let us assume that there is a flat plate having infinite length and width of 1 unit. The fluid has thermal diffusivity α, viscosity $\nu = 0.001$ and Prandtl Number $Pr = 0.7$. Relation between α, ν and Pr is given below:

$$Pr = \frac{\nu}{\alpha} \tag{2.69}$$

In this case, the equations governing the convective flow are given by:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, x \in [0,5], y \in [0,1] \tag{2.70}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\frac{\partial^2 u}{\partial y^2} \tag{2.71}$$

$$u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \alpha\frac{\partial^2 T}{\partial y^2} \tag{2.72}$$

$$u(0,y) = 1, v(0,y) = 0, T(0,y) = 0 \tag{2.73}$$

$$u(x,0) = 0, v(x,0) = 0, T(x,0) = 10 \tag{2.74}$$

$$u(x,1) = 1, \left.\frac{\partial v}{\partial y}\right|_{y=1} = 0, T(x,1) = 0 \tag{2.75}$$

where u and v denote the velocity in the x and y direction respectively and T denotes the Temperature.

Solution of the Forward Problem

The forward problem can be solved using a PINN which takes x and y as inputs and predicts the velocities $\hat{u}$ and $\hat{v}$ and the temperature $\hat{T}$. We will solve the problem on a mesh grid x and y of size 50x50. As we do not have any training samples, the total loss is just the physics loss and is given by the following expression:

$$\begin{aligned}
\mathscr{L}_{\text{total}} = & \frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{\partial \hat{u}}{\partial x} + \frac{\partial \hat{v}}{\partial y})^2 + \frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{u}}{\partial x} + \hat{v}\frac{\partial \hat{u}}{\partial y} - \nu\frac{\partial^2 \hat{u}}{\partial y^2})^2 + \frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{T}}{\partial x} + \hat{v}\frac{\partial \hat{T}}{\partial y} - \alpha\frac{\partial^2 \hat{T}}{\partial y^2})^2 \\
& + \frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{u}(0,y_i) - 1)^2 + \frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{v}(0,y_i) - 0)^2 + \frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{T}(0,y_i) - 0)^2 \\
& + \frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\hat{u}(x_i,0) - 0)^2 + \frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\hat{v}(x_i,0) - 0)^2 + \frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\hat{T}(x_i,0) - 10)^2 \\
& + \frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{u}(x_i,1) - 1)^2 + \frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\left.\frac{\partial \hat{v}}{\partial y}\right|_{y=1} - 0)^2 + \frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{T}(x_i,1) - 0)^2
\end{aligned} \tag{2.76}$$

where N_m, N_{b1}, N_{b2} and N_{b3} represent the number of points on the mesh grid, number of points on the boundary $x = 0$, number of points on the boundary $y = 0$ and number of points on the boundary $y = 1$ respectively. In this case, N_m, N_{b1}, N_{b2} and N_{b3} is 1600, 40, 40 and 40 respectively. The PINN has 10 hidden layers with each layer having 40 nodes with $tanh$ activation function. The loss function is minimized using Adam optimizer having learning rate 5x10^{-4}. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 103.11 to 1.23 after training for 2x10^4 epochs. The plot of total loss vs epoch, the contour plot of velocity in x direction v/s the spatial variables x and y, the contour plot of velocity in y direction v/s the spatial variables x and y and the contour plot of Temperature T v/s the spatial variables x and y are shown in Figure A.23, Figure A.24, Figure A.25 and Figure A.26 respectively.

Plot of velocity and Temperature profiles at various values of x are given in Figure A.27, Figure A.28 and Figure A.29.

The model has been trained on a meshgrid of x and y. Corresponding values of η can be calculated by using the relation given below:

$$\eta = y(\frac{U}{\nu x})^{\frac{1}{2}} \tag{2.77}$$

Python code for the forward solution of Flow over flat plate Problem can be found at this link.

2.0.8 Full Navier Stokes Equation with Energy consideration

Let us assume that there is a flat plate having infinite length and width of 1 unit. The fluid has density $\rho = 1.293$, thermal diffusivity α, viscosity $\nu = 0.001$ and Prandtl Number $Pr = 0.7$. Relation between α, ν and Pr is given below:

$$Pr = \frac{\nu}{\alpha} \tag{2.78}$$

In this case, the equations governing the convective flow are given by:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, x \in [0,5], y \in [0,1] \tag{2.79}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial x} + \nu(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{2.80}$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial P}{\partial y} + \nu(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) \tag{2.81}$$

$$u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = \alpha(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}) \tag{2.82}$$

$$u(0,y) = 1, v(0,y) = 0, T(0,y) = 0, P(0,y) = 1 \tag{2.83}$$

$$\left.\frac{\partial u}{\partial x}\right|_{x=5} = 0, \left.\frac{\partial v}{\partial x}\right|_{x=5} = 0, \left.\frac{\partial T}{\partial x}\right|_{x=5} = 0, \left.\frac{\partial P}{\partial x}\right|_{x=5} = 0 \tag{2.84}$$

$$u(x,0) = 0, v(x,0) = 0, T(x,0) = 10, \left.\frac{\partial P}{\partial y}\right|_{y=0} = 0 \tag{2.85}$$

$$u(x,1) = 1, \left.\frac{\partial v}{\partial y}\right|_{y=1} = 0, T(x,1) = 0, \left.\frac{\partial P}{\partial y}\right|_{y=1} = 0 \tag{2.86}$$

where u and v denote the velocity in the x and y direction respectively, T denotes the Temperature and P denotes the pressure.

Solution of the Forward Problem

The forward problem can be solved using a PINN which takes x and y as inputs and predicts the velocities $\hat{u}$ and $\hat{v}$, temperature $\hat{T}$ and Pressure $\hat{P}$. We will solve the problem on a mesh grid x and y of size 50x50. As we do not have any training samples, the total loss is just the physics loss and is given by the following expression:

$$
\begin{aligned}
\mathscr{L}_{\text{total}} = &\frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{\partial \hat{u}}{\partial x}+\frac{\partial \hat{v}}{\partial y})^2 + \frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{u}}{\partial x}+\hat{v}\frac{\partial \hat{u}}{\partial y}+\frac{1}{\rho}\frac{\partial P}{\partial x}-\nu((\frac{\partial^2 \hat{u}}{\partial x^2})^2+(\frac{\partial^2 \hat{u}}{\partial y^2})^2))\\
&+\frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{v}}{\partial x}+\hat{v}\frac{\partial \hat{v}}{\partial y}+\frac{1}{\rho}\frac{\partial P}{\partial y}-\nu((\frac{\partial^2 \hat{v}}{\partial x^2})^2+(\frac{\partial^2 \hat{v}}{\partial y^2})^2))+\frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{T}}{\partial x}+\hat{v}\frac{\partial \hat{T}}{\partial y}-\alpha((\frac{\partial^2 \hat{T}}{\partial x^2})^2+(\frac{\partial^2 \hat{T}}{\partial y^2})^2))\\
&+\frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{u}(0,y_i)-1)^2+\frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{v}(0,y_i)-0)^2+\frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{T}(0,y_i)-0)^2+\frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{P}(0,y_i)-1)^2\\
&+\frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\frac{\partial \hat{u}}{\partial x}\bigg|_{x=5}-0)^2+\frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\frac{\partial \hat{v}}{\partial x}\bigg|_{x=5}-0)^2+\frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\frac{\partial \hat{T}}{\partial x}\bigg|_{x=5}-0)^2+\frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\frac{\partial \hat{P}}{\partial x}\bigg|_{x=5}-0)^2\\
&+\frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{u}(x_i,0)-0)^2+\frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{v}(x_i,0)-0)^2+\frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{T}(x_i,0)-10)^2+\frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\frac{\partial \hat{P}}{\partial y}\bigg|_{y=0}-0)^2\\
&+\frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\hat{u}(x_i,1)-1)^2+\frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\frac{\partial \hat{v}}{\partial y}\bigg|_{y=1}-0)^2+\frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\hat{T}(x_i,1)-0)^2+\frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\frac{\partial \hat{P}}{\partial y}\bigg|_{y=1}-0)^2
\end{aligned}
\tag{2.87}
$$

where N_m, N_{b1}, N_{b2}, N_{b3} and N_{b4} represent the number of points on the mesh grid, number of points on the boundary $x=0$, number of points on the boundary $x=5$, number of points on the boundary $y=0$ and number of points on the boundary $y=1$ respectively. In this case, N_m, N_{b1}, N_{b2}, , N_{b3} and N_{b4} is 2500, 50, 50, 50 and 50 respectively. The PINN has 10 hidden layers with each layer having 40 nodes with $tanh$ activation function. The loss function is minimized using Adam optimizer having learning rate 2x10^{-4}. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 203.36 to 0.99 after training for 2x10^4 epochs. The plot of total loss vs epoch and the contour plot of velocity in u, v, T and P v/s the spatial variables x and y are shown in Figure A.38, Figure A.30, Figure A.31, Figure A.32 and Figure A.33 respectively.

Velocity, Temperature and pressure profiles at various values of x are given in Figure A.34, Figure A.35, Figure A.36 and Figure A.37.

Solution of the Inverse Problem

The inverse problem can be modeled as:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0, x \in [0,5], y \in [0,1] \tag{2.88}$$

$$u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{1}{\rho}\frac{\partial P}{\partial x}+\nu(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}) \tag{2.89}$$

$$u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial P}{\partial y}+\nu(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}) \tag{2.90}$$

$$u\frac{\partial T}{\partial x}+v\frac{\partial T}{\partial y}=\alpha(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}) \tag{2.91}$$

$$u(0,y)=1, v(0,y)=0, T(0,y)=0, P(0,y)=1 \tag{2.92}$$

$$\left.\frac{\partial u}{\partial x}\right|_{x=5}=0, \left.\frac{\partial v}{\partial x}\right|_{x=5}=0, \left.\frac{\partial T}{\partial x}\right|_{x=5}=0, \left.\frac{\partial P}{\partial x}\right|_{x=5}=0 \tag{2.93}$$

$$u(x,0)=0, v(x,0)=0, T(x,0)=10, \left.\frac{\partial P}{\partial y}\right|_{y=0}=0 \tag{2.94}$$

$$u(x,1)=1, \left.\frac{\partial v}{\partial y}\right|_{y=1}=0, T(x,1)=0, \left.\frac{\partial P}{\partial y}\right|_{y=1}=0 \tag{2.95}$$

where Prandtl Number Pr is an unknown variable. We are also given the values of u and v at 35 points lying on 7x5 mesh grid formed by $x=[1.0,2.0,2.5,3.0,3.5,4.0,5.0]$ and $y=[0.1,0.2,0.4,0.6,0.8]$. The total loss function is given by,

$$\mathscr{L}_{\text{total}}=\mathscr{L}_{\text{MSE}}+\mathscr{L}_{\text{Physics}} \tag{2.96}$$

where,

$$\mathscr{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{n}(\hat{u}_i - u_i)^2 + \frac{1}{N}\sum_{i=1}^{n}(\hat{v}_i - v_i)^2 + \frac{1}{N}\sum_{i=1}^{n}(\hat{T}_i - T_i)^2 + \frac{1}{N}\sum_{i=1}^{n}(\hat{P}_i - P_i)^2 \tag{2.97}$$

$$\begin{aligned}
\mathscr{L}_{\text{physics}} = {} & \frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{\partial \hat{u}}{\partial x} + \frac{\partial \hat{v}}{\partial y})^2 + \frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{u}}{\partial x} + \hat{v}\frac{\partial \hat{u}}{\partial y} + \frac{1}{\rho}\frac{\partial P}{\partial x} - \nu((\frac{\partial^2 \hat{u}}{\partial x^2})^2 + (\frac{\partial^2 \hat{u}}{\partial y^2})^2)) \\
& + \frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{v}}{\partial x} + \hat{v}\frac{\partial \hat{v}}{\partial y} + \frac{1}{\rho}\frac{\partial P}{\partial y} - \nu((\frac{\partial^2 \hat{v}}{\partial x^2})^2 + (\frac{\partial^2 \hat{v}}{\partial y^2})^2)) + \frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{T}}{\partial x} + \hat{v}\frac{\partial \hat{T}}{\partial y} - \alpha((\frac{\partial^2 \hat{T}}{\partial x^2})^2 + (\frac{\partial^2 \hat{T}}{\partial y^2})^2) \\
& + \frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{u}(0,y_i) - 1)^2 + \frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{v}(0,y_i) - 0)^2 + \frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{T}(0,y_i) - 0)^2 + \frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{P}(0,y_i) - 1)^2 \\
& + \frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\frac{\partial \hat{u}}{\partial x}\bigg|_{x=5} - 0)^2 + \frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\frac{\partial \hat{v}}{\partial x}\bigg|_{x=5} - 0)^2 + \frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\frac{\partial \hat{T}}{\partial x}\bigg|_{x=5} - 0)^2 + \frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\frac{\partial \hat{P}}{\partial x}\bigg|_{x=5} - 0)^2 \\
& + \frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{u}(x_i,0) - 0)^2 + \frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{v}(x_i,0) - 0)^2 + \frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{T}(x_i,0) - 10)^2 + \frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\frac{\partial \hat{P}}{\partial y}\bigg|_{y=0} - 0)^2 \\
& + \frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\hat{u}(x_i,1) - 1)^2 + \frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\frac{\partial \hat{v}}{\partial y}\bigg|_{y=1} - 0)^2 + \frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\hat{T}(x_i,1) - 0)^2 + \frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\frac{\partial \hat{P}}{\partial y}\bigg|_{y=1} - 0)^2
\end{aligned} \tag{2.98}$$

where N represents the number of training data samples. The Inverse PINN has 10 hidden layers with each layer having 40 nodes with $tanh$ activation function. The Inverse PINN takes x and y as input and outputs the velocities $\hat{u}$ and $\hat{v}$ while updating Pr simultaneously as the weights of the neural network are updated, but with a different learning rate. The learning rate for the weights of artificial neural network and ν is 2x10^{-4} and 2x10^{-3} respectively. The total loss $\mathscr{L}_{\text{total}}$ gets reduced from 117.28 to 0.99 after training for5x10^{4} epochs. Pr was initialised to be 0.5 units, but after training it came out to be 0.06970 units against the expected value of 0.7 units.

Python code for the solution of Flow over flat plate Problem can be found at this link.

2.0.9 2-D Lid Driven Cavity Problem

Let us consider a square cavity with side length $L = 1$ unit. The top lid of the cavity is moving with a constant velocity, creating a lid-driven flow inside the cavity. The properties of the fluid are given by Reynolds Number $Re = 1000$. The boundary conditions for the lid-driven cavity problem are: The top lid: No-slip condition, i.e., the velocity of the lid is U and all velocities on the lid are set to U. The other three walls: No-slip and no-penetration conditions, i.e., the velocity is zero and there is no fluid penetration through the walls. The fluid inside the cavity: Incompressible and governed by the Navier-Stokes equations along with the continuity equation. This problem aims to study the flow patterns and the velocity profiles inside the lid-driven cavity. In this case, the equations governing the Lid Driven Cavity flow are given by:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, x \in [0,1], y \in [0,1] \tag{2.99}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial P}{\partial x} + \frac{1}{Re}(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) \tag{2.100}$$

$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial P}{\partial y} + \frac{1}{Re}(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) \tag{2.101}$$

$$u(0,y) = v(0,y) = 0, u(1,y) = v(1,y) = 0 \tag{2.102}$$

$$u(x,0) = v(x,0) = 0, u(x,1) = 1, v(x,1) = 0 \tag{2.103}$$

where u and v denote the velocity in the x and y direction respectively, and P denotes the pressure.

Solution of the Forward Problem

The forward problem can be solved using a PINN which takes x and y as inputs and predicts the velocities $\hat{u}$ and $\hat{v}$, and Pressure $\hat{P}$. We will solve the problem on a mesh grid x and y of size 31x31. Here we are considering few data points from CFD as training samples. So, the total loss is CFD data loss and the physics loss which is given by the following expression:

$$\begin{aligned}\mathscr{L}_{\text{total}} = {} & \frac{1}{N_m}\sum_{i=1}^{N_m}(\frac{\partial \hat{u}}{\partial x}+\frac{\partial \hat{v}}{\partial y})^2+\frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{u}}{\partial x}+\hat{v}\frac{\partial \hat{u}}{\partial y}+\frac{\partial P}{\partial x}-\frac{1}{Re}((\frac{\partial^2 \hat{u}}{\partial x^2})^2+(\frac{\partial^2 \hat{u}}{\partial y^2})^2))+\frac{1}{N_{cfd_x}}\sum_{i=1}^{N_{cfd_x}}(\hat{u}(0.5,y_i)-0) \\ & +\frac{1}{N_m}\sum_{i=1}^{N_m}(\hat{u}\frac{\partial \hat{v}}{\partial x}+\hat{v}\frac{\partial \hat{v}}{\partial y}+\frac{\partial P}{\partial y}-\frac{1}{Re}((\frac{\partial^2 \hat{v}}{\partial x^2})^2+(\frac{\partial^2 \hat{v}}{\partial y^2})^2))+\frac{1}{N_{cfd_y}}\sum_{i=1}^{N_{cfd_y}}(\hat{v}(x_i,0.5)-0)^2 \\ & +\frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{u}(0,y_i)-0)^2+\frac{1}{N_{b1}}\sum_{i=1}^{N_{b1}}(\hat{v}(0,y_i)-0)^2+\frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\hat{u}(1,y_i)-0)^2+\frac{1}{N_{b2}}\sum_{i=1}^{N_{b2}}(\hat{v}(1,y_i)-0)^2 \\ & +\frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{u}(x_i,0)-0)^2+\frac{1}{N_{b3}}\sum_{i=1}^{N_{b3}}(\hat{v}(x_i,0)-0)^2+\frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\hat{u}(x_i,1)-1)^2+\frac{1}{N_{b4}}\sum_{i=1}^{N_{b4}}(\hat{v}(x_i,1)-0)^2\end{aligned} \tag{2.104}$$

where $N_m, N_{cfd_x}, N_{cfd_y}, N_{b1}, N_{b2}, N_{b3}$ and N_{b4} represent the number of points on the mesh grid, number of data points from cfd data for x velocity, number of data points from cfd data for y velocity, number of points on the boundary $x = 0$, number of points on the boundary $x = 1$, number of points on the boundary $y = 0$ and number of points on the boundary $y = 1$ respectively. In this case, $N_m, N_{cfd_x}, N_{cfd_y}, N_{b1}, N_{b2}, N_{b3}$ and N_{b4} is 961, 17, 17, 101, 101, 101 and 101 respectively. The PINN has 4 hidden layers with each layer having 30 nodes with *radialbasisfunction* as activation function. The loss function is minimized using Adam optimizer having learning rate 2x10^{-4}. The plot of total loss vs epoch, Stream line plot of the cavity and the contour plot of velocity in $u$and v v/s the spatial variables x and y are shown in Figure A.38, Figure A.39, Figure A.40, and Figure A.41 respectively.

Chapter 3

Conclusion

In this project, we developed Physics Informed Machine Learning Models for solving Forward and Inverse Problems in Heat and Mass Transfer and successfully achieved results faster than any of the existing methodologies with negligible error.

3.0.1 Limitations of current work

In our attempt to solve the lid-driven cavity problem using Physics-Informed Neural Networks (PINN), we've encountered some challenges. While the bulk of the stream plots align reasonably well with the benchmark solutions, there are discrepancies, particularly at the boundaries. Notably, the top boundary deviates significantly from the expected benchmark solutions. Additionally, the total loss has not decreased to our desired value, indicating that our current model might not be capturing the underlying physics or features of the problem adequately.

These discrepancies suggest that there are aspects of our PINN structure that require further refinement and optimization. We recognize the importance of accurately capturing boundary conditions and overall loss reduction for a reliable and accurate solution.

3.0.2 Future Prospects

Given the observed discrepancies and challenges, our next steps will focus on two main areas:
(i) We will revisit and refine the architecture of our PINN to better capture the physics of the lid-driven cavity problem. This might involve adjusting the number of layers, nodes, or incorporating additional features to enhance the model's performance, especially at the boundaries.
(ii) We will explore and implement advanced optimization algorithms to improve the training process and reduce the total loss.
(ii) We will explore other neural network architectures such as Convolutional Neural Networks for solving Inverse Problems in Heat Transfer.

Appendices

Appendix A

Figures

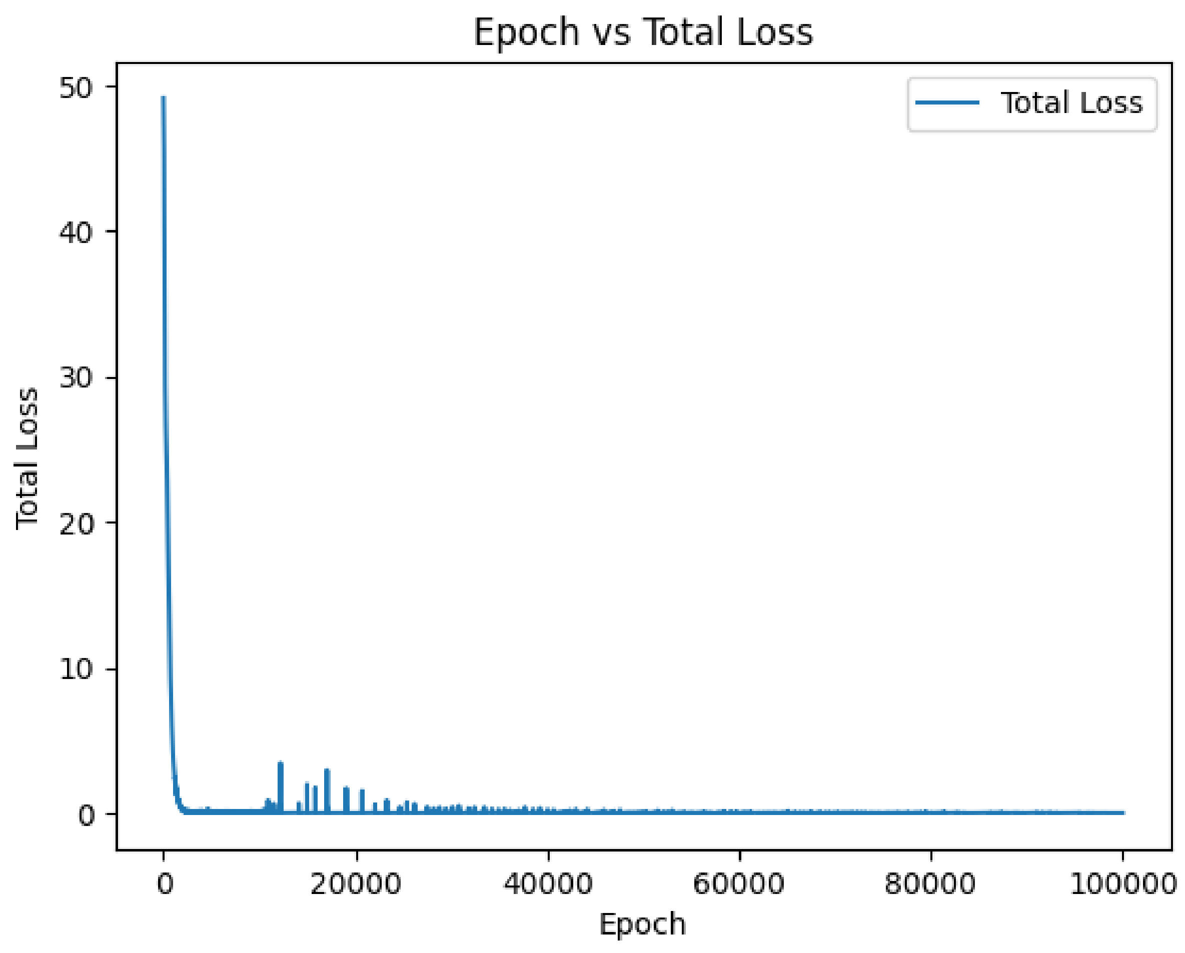

Figure A.1: Plot of Total loss vs Epoch for solving forward Steady State One Dimensional Heat Diffusion Equation with constant thermal diffusivity

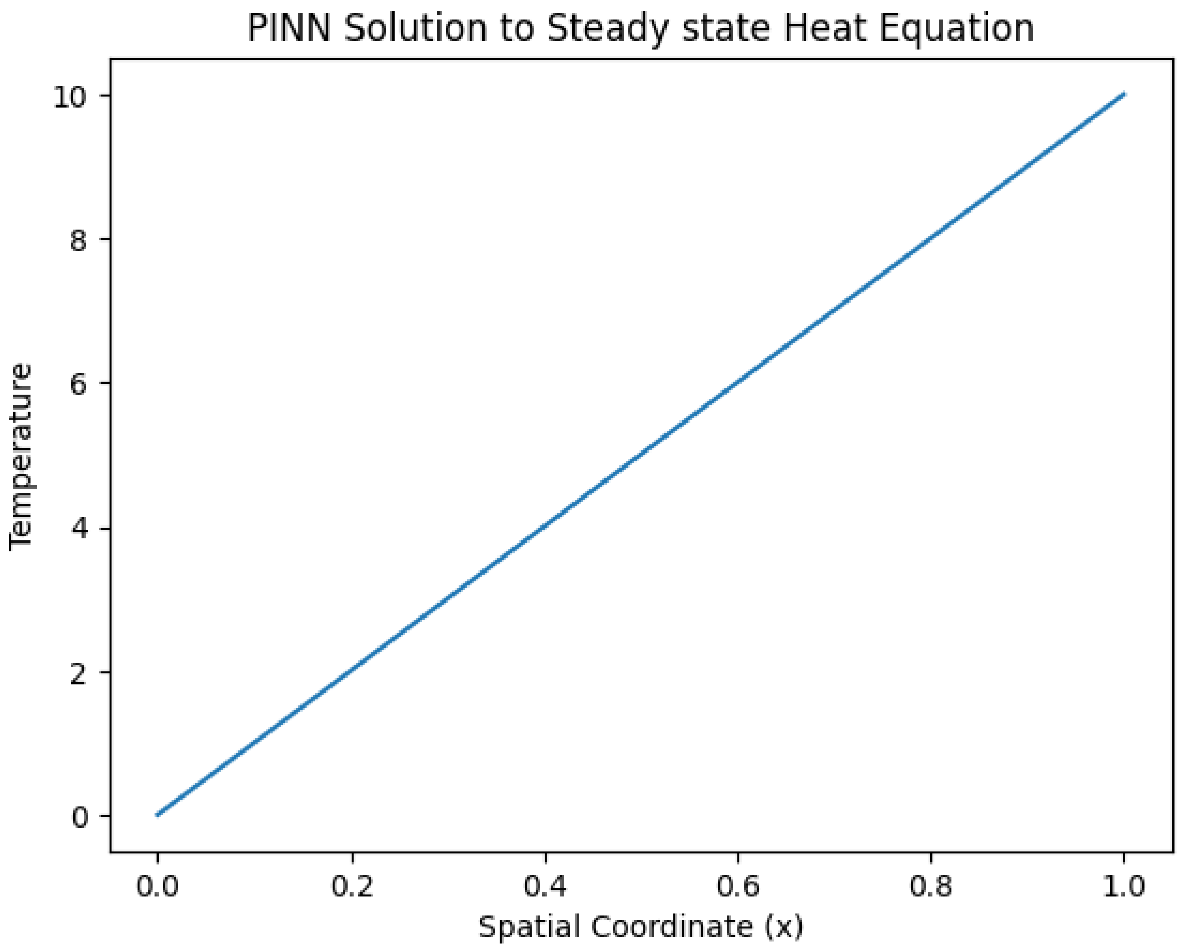

Figure A.2: Plot of Temperature v/s independent variable for solving forward Steady State One Dimensional Heat Diffusion Equation with constant thermal diffusivity

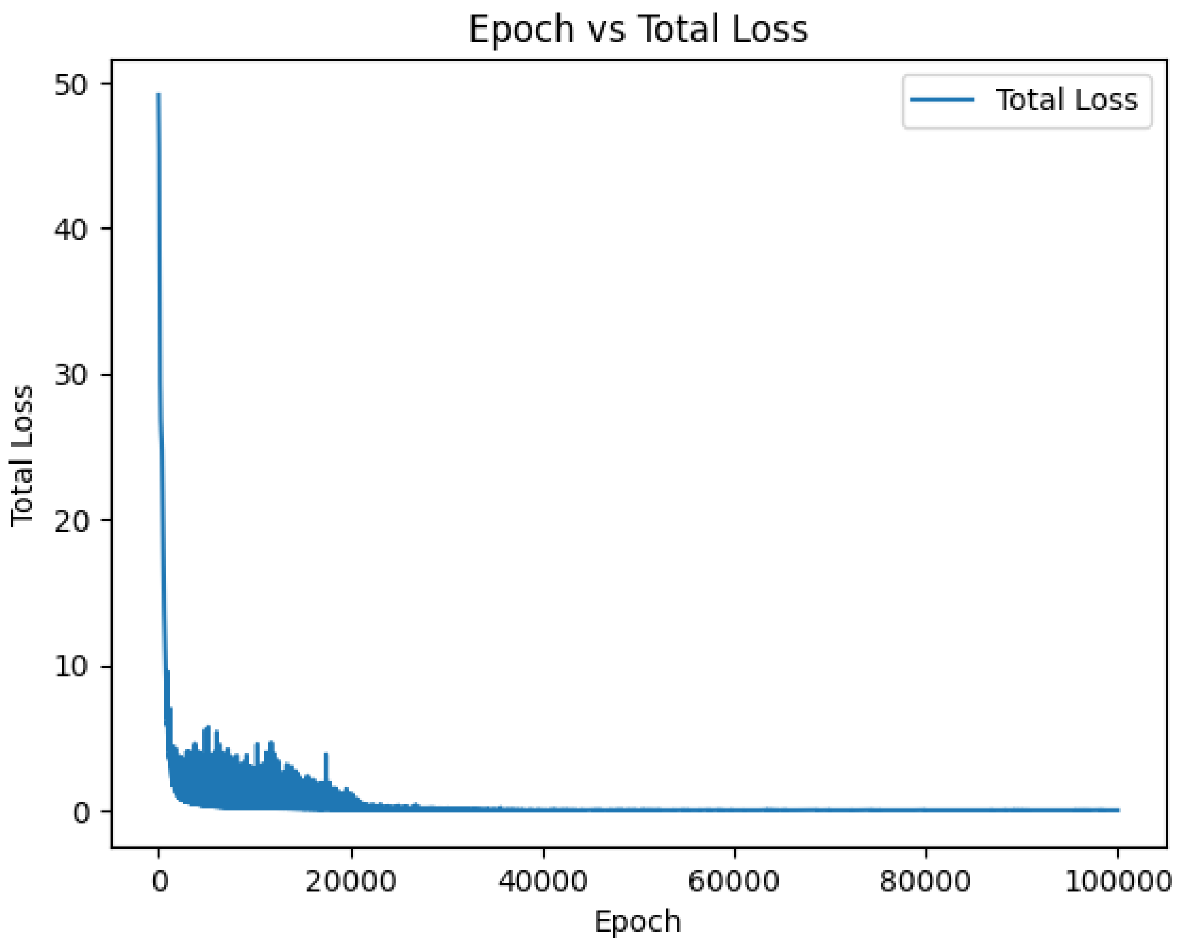

Figure A.3: Plot of Total loss vs Epoch for solving forward Steady State One Dimensional Heat Diffusion Equation with variable thermal diffusivity

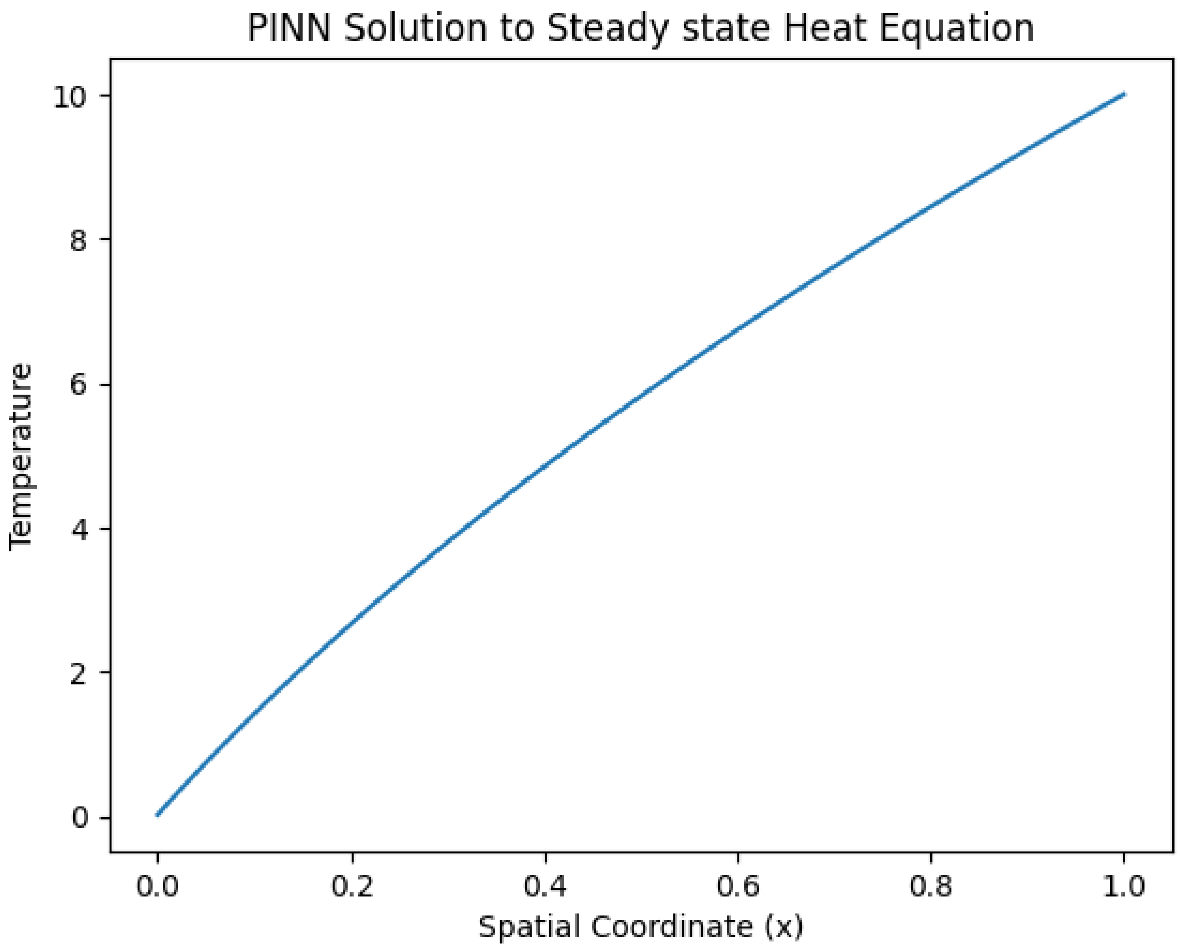

Figure A.4: Plot of Temperature v/s independent variable for solving forward Steady State One Dimensional Heat Diffusion Equation with variable thermal diffusivity

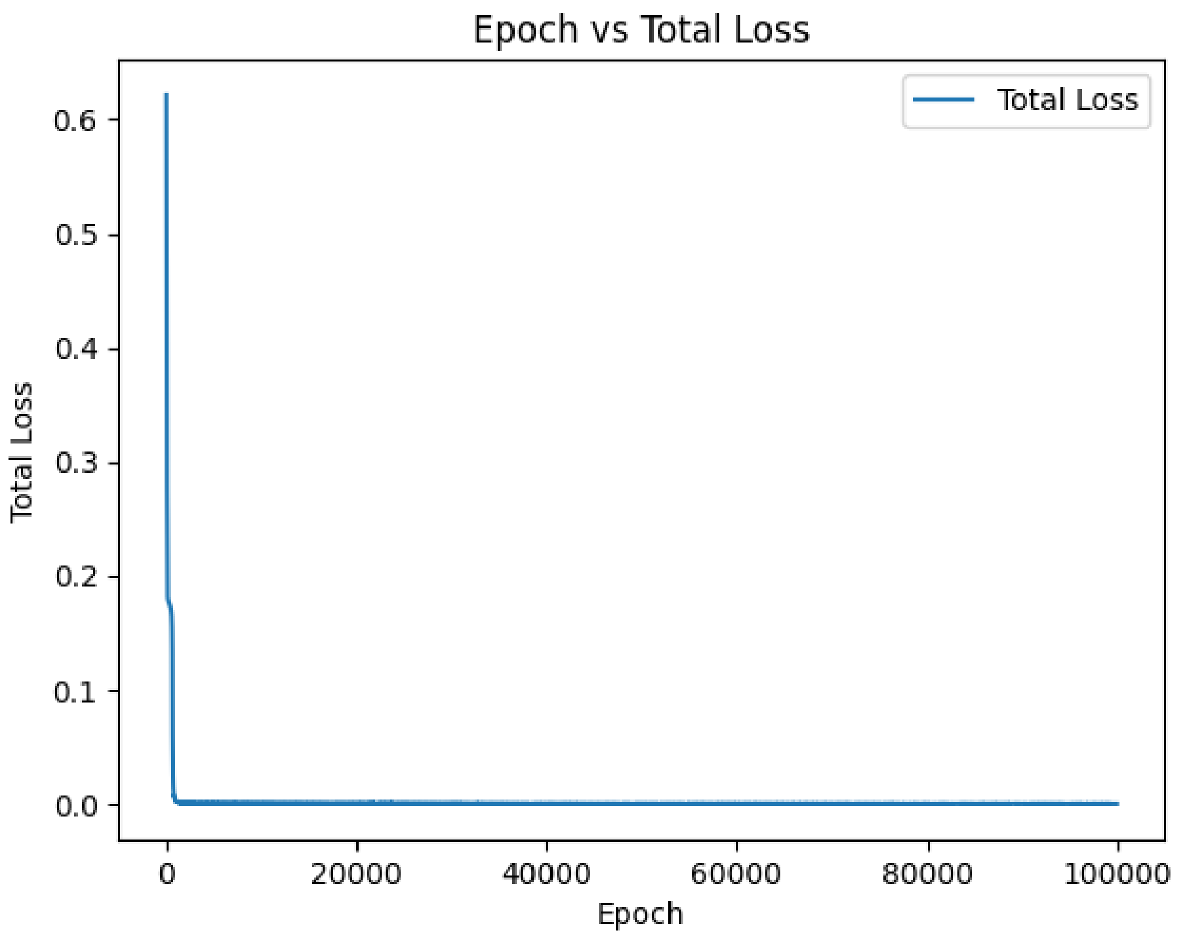

Figure A.5: Plot of Total loss vs Epoch for solving forward unsteady One Dimensional Heat Diffusion Equation

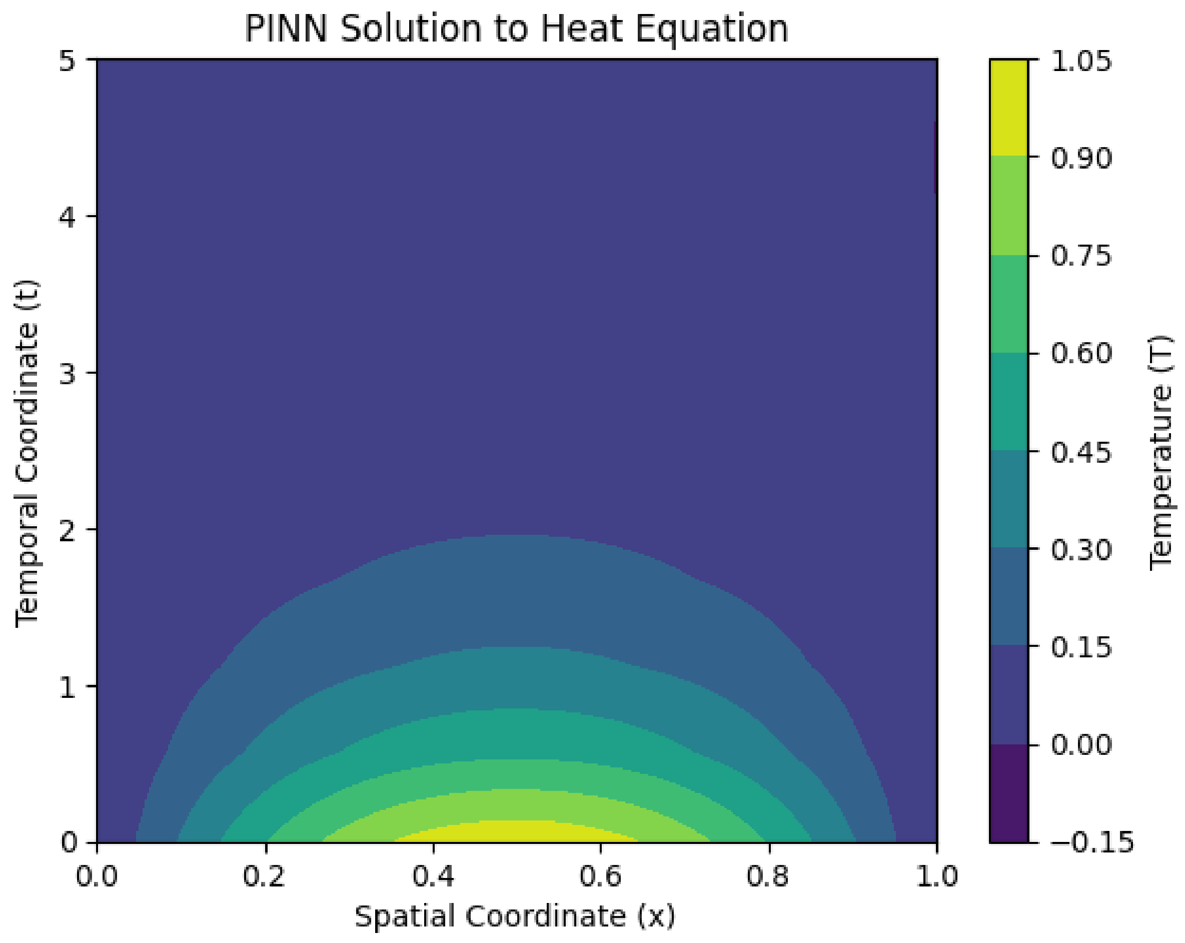

Figure A.6: Contour Plot of Temperature v/s Spatial variable x and time t for solving forward unsteady One Dimensional Heat Diffusion Equation

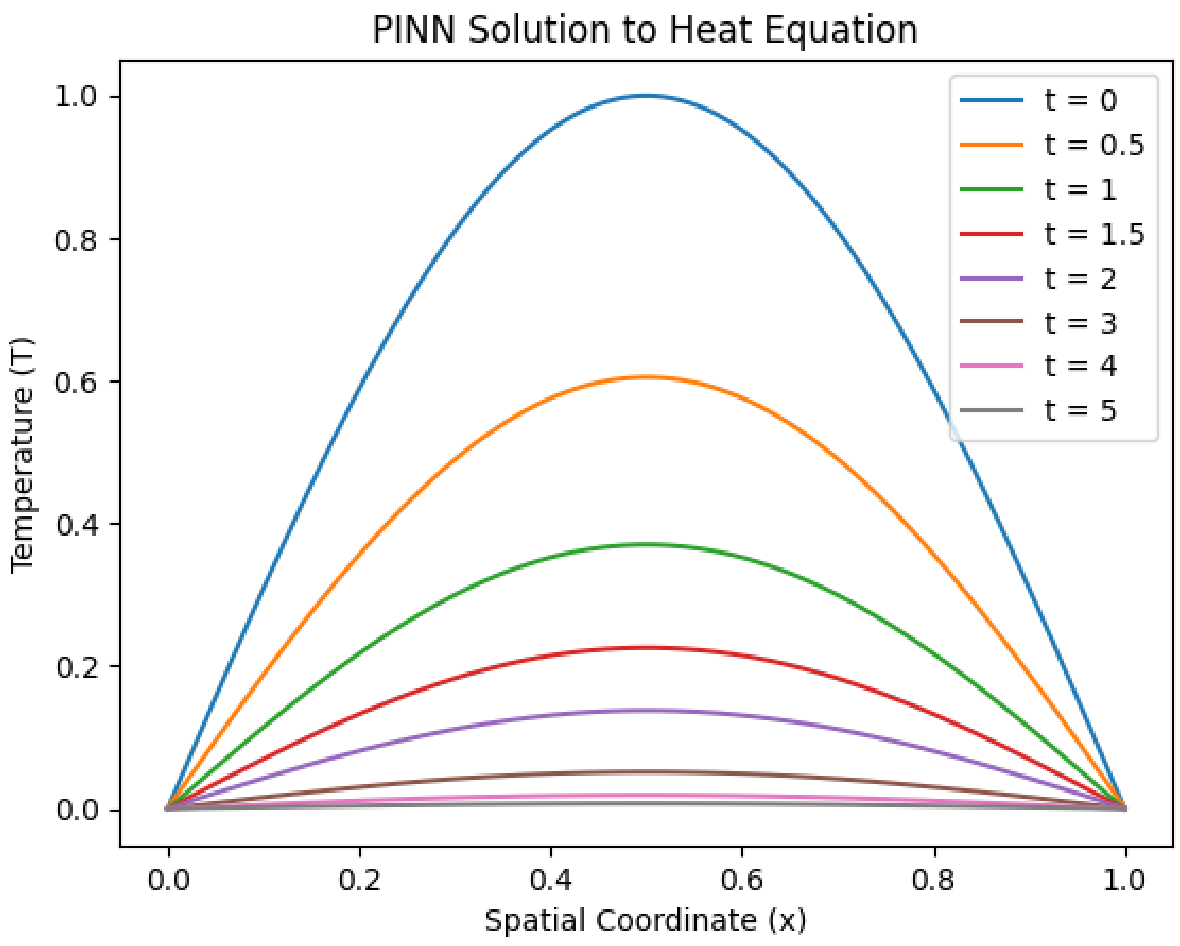

Figure A.7: Plot of Temperature v/s spatial variable x for discrete values of time t for solving forward Steady State One Dimensional Heat Diffusion Equation

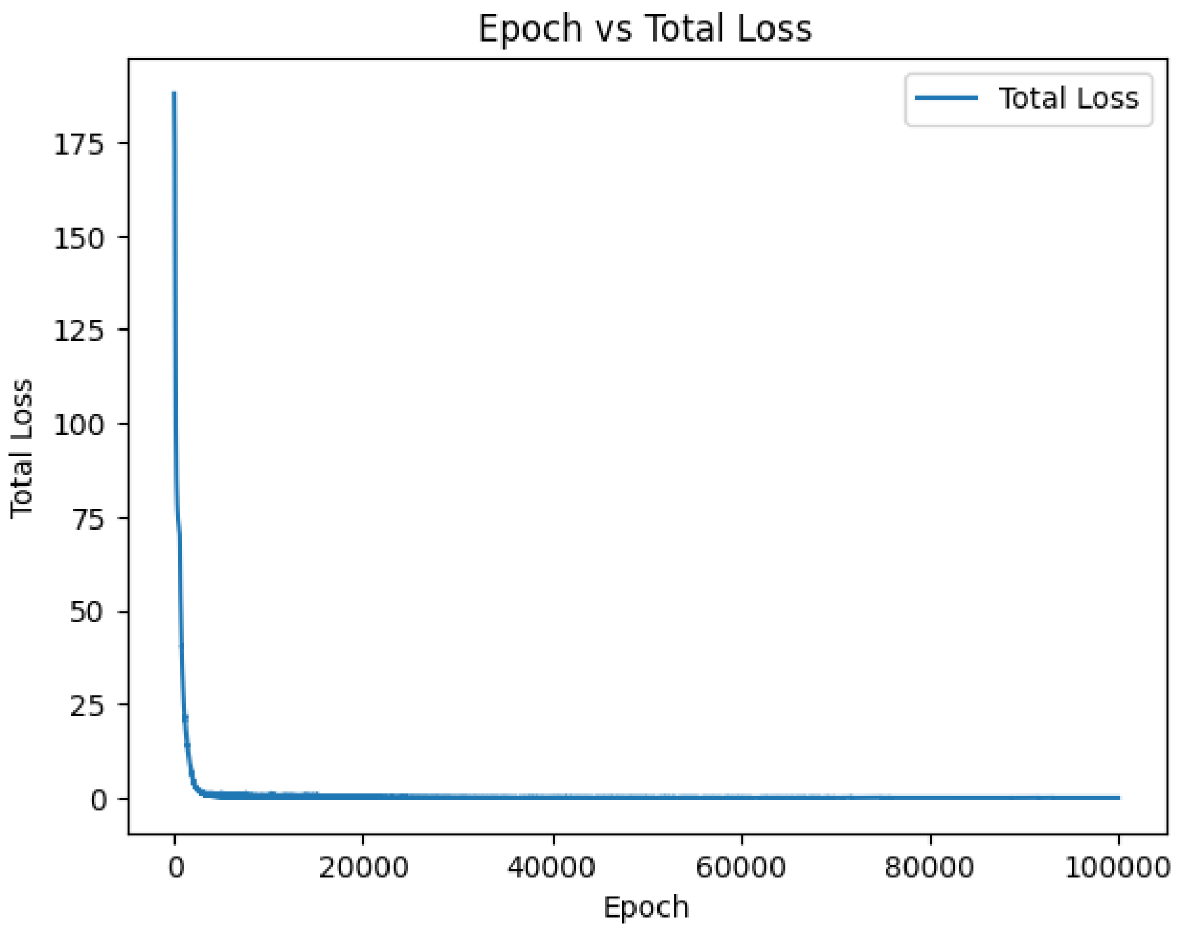

Figure A.8: Plot of Total loss vs Epoch for solving forward Steady Two Dimensional Heat Diffusion Equation

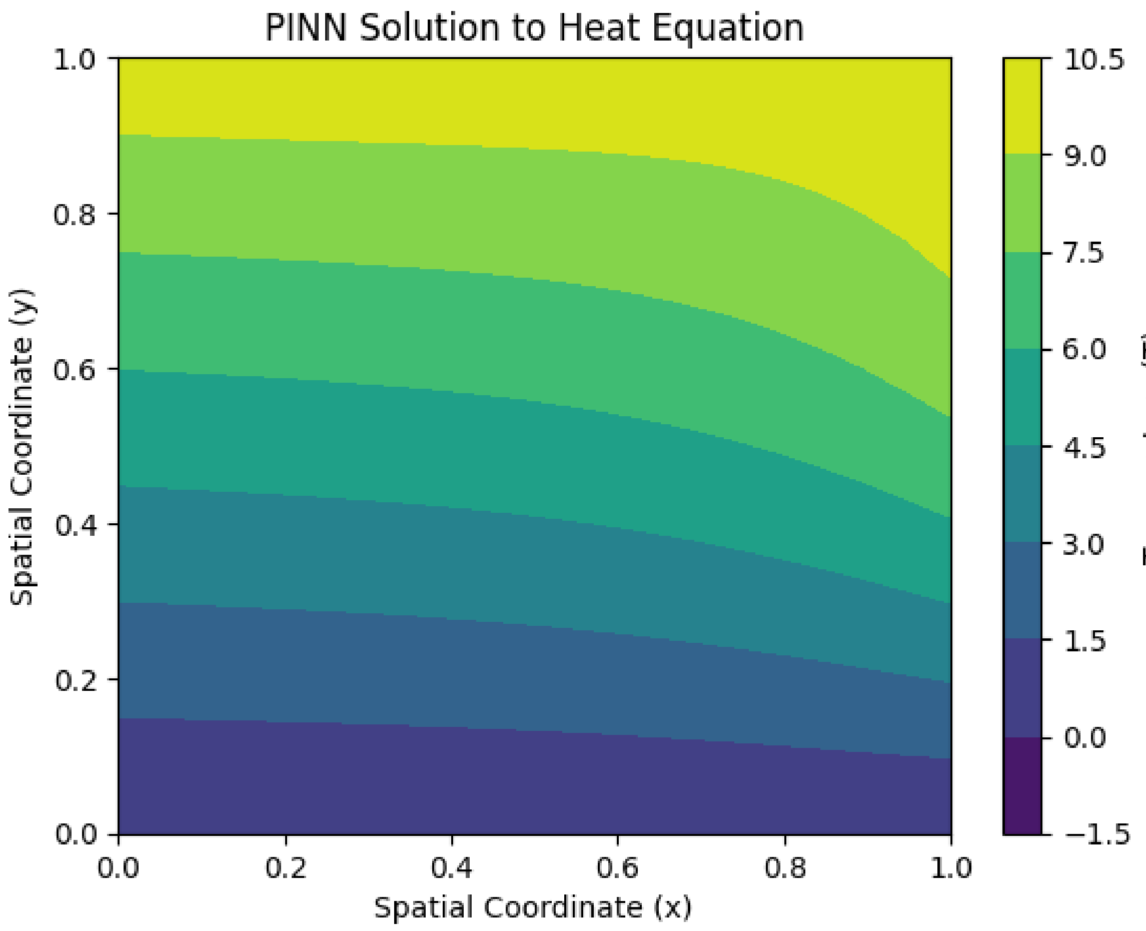

Figure A.9: Contour Plot of Temperature v/s Spatial variables x and y for solving forward Steady Two Dimensional Heat Diffusion Equation

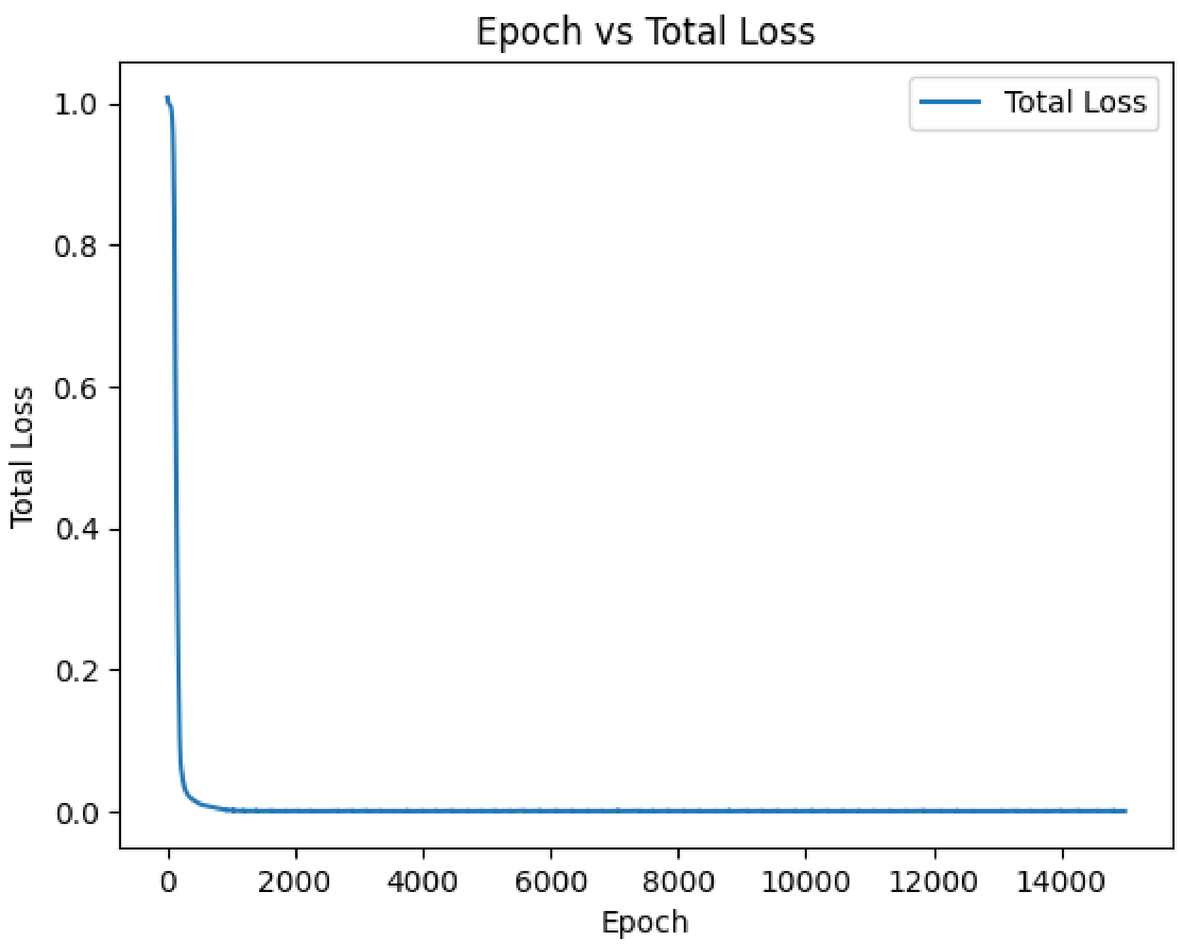

Figure A.10: Plot of Total loss vs Epoch for solving Blasius Equation

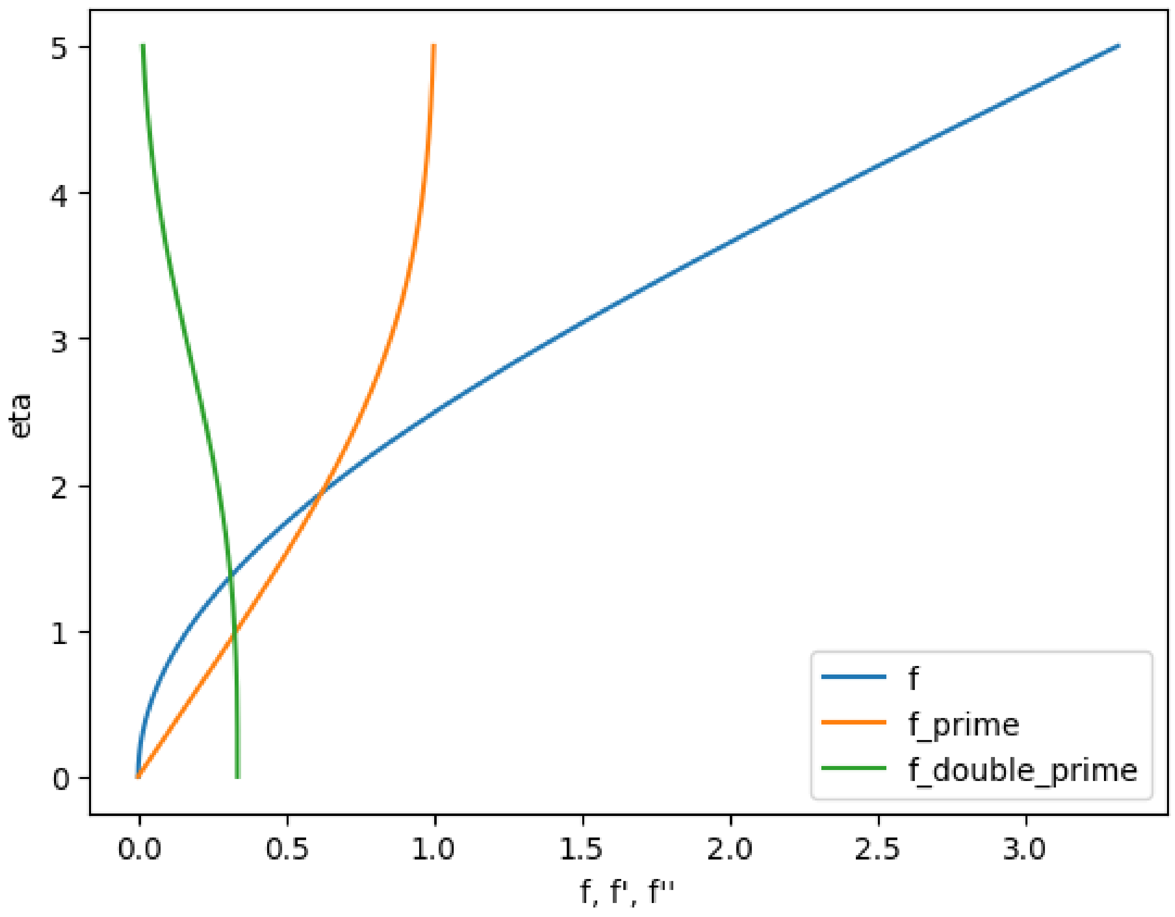

Figure A.11: Plot of f, $\frac{d\hat{f}}{d\eta}$ and $\frac{d\hat{f}^2}{d\eta^2}$ v/s the spatial variable η for Blasius Equation

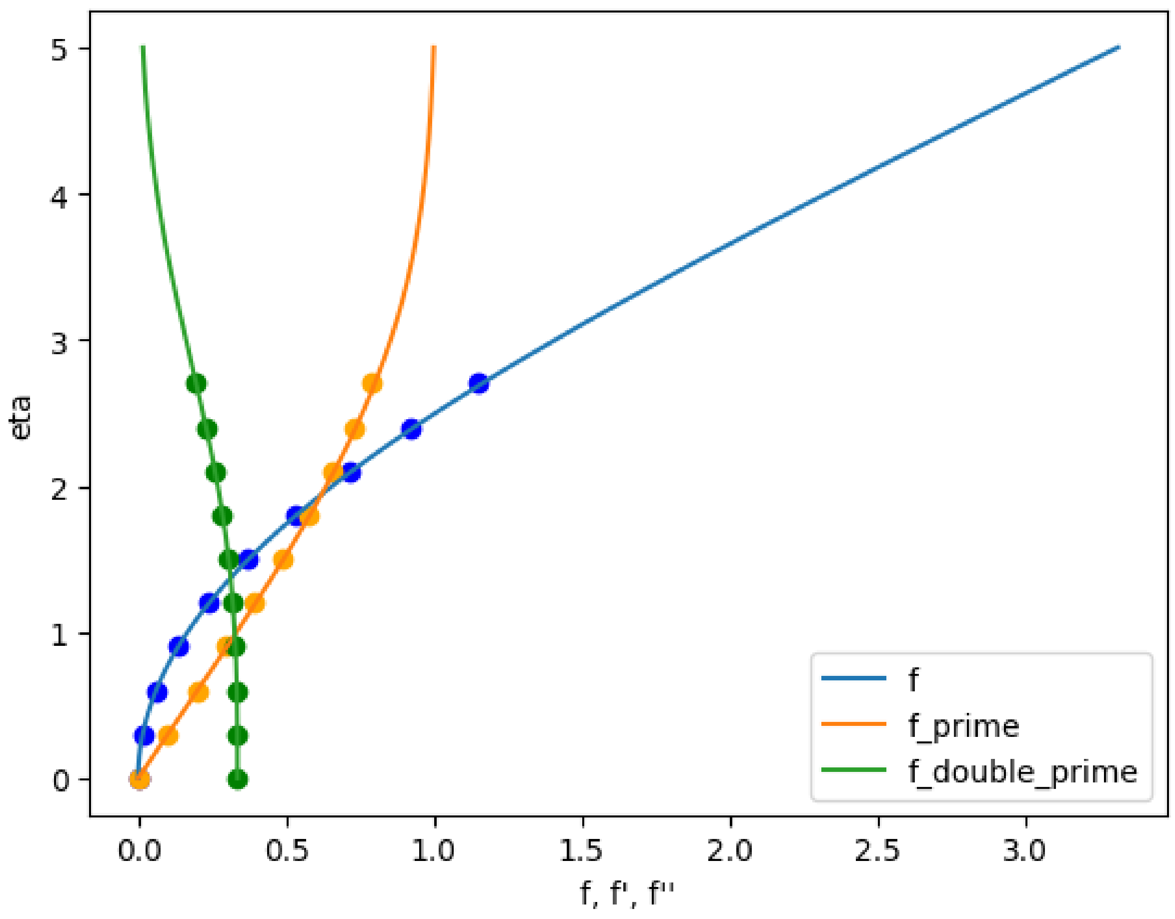

Figure A.12: Plot of $f, \frac{d\hat{f}}{d\eta}$ and $\frac{d\hat{f}^2}{d\eta^2}$ v/s the spatial variable η for Blasius Equation. The dots represent experimental values of the quantity whereas the lines represent model predictions.

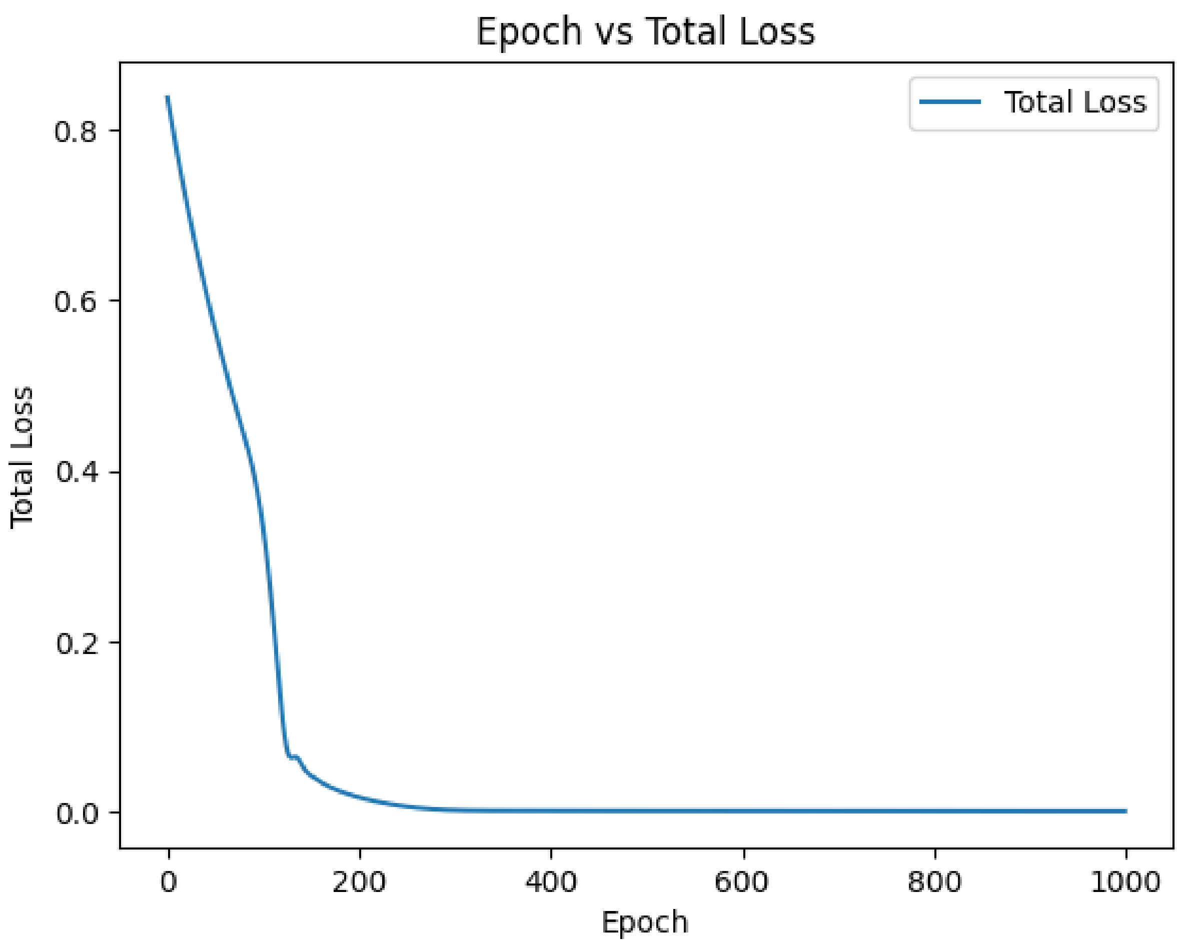

Figure A.13: Plot of Total loss vs Epoch for solving Blasius Equation

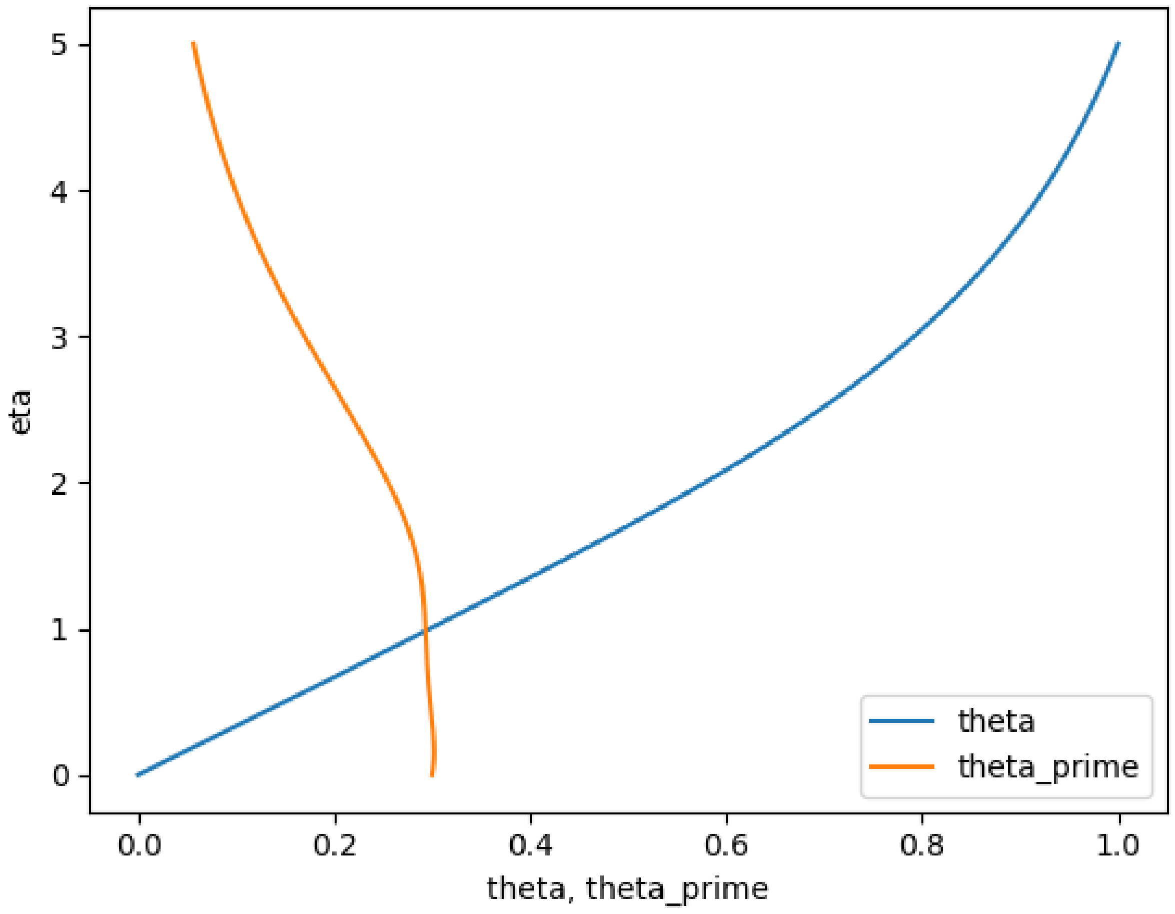

Figure A.14: Plot of θ and $\frac{d\hat{\theta}}{d\eta}$ v/s the spatial variable η for Blasius Equation.

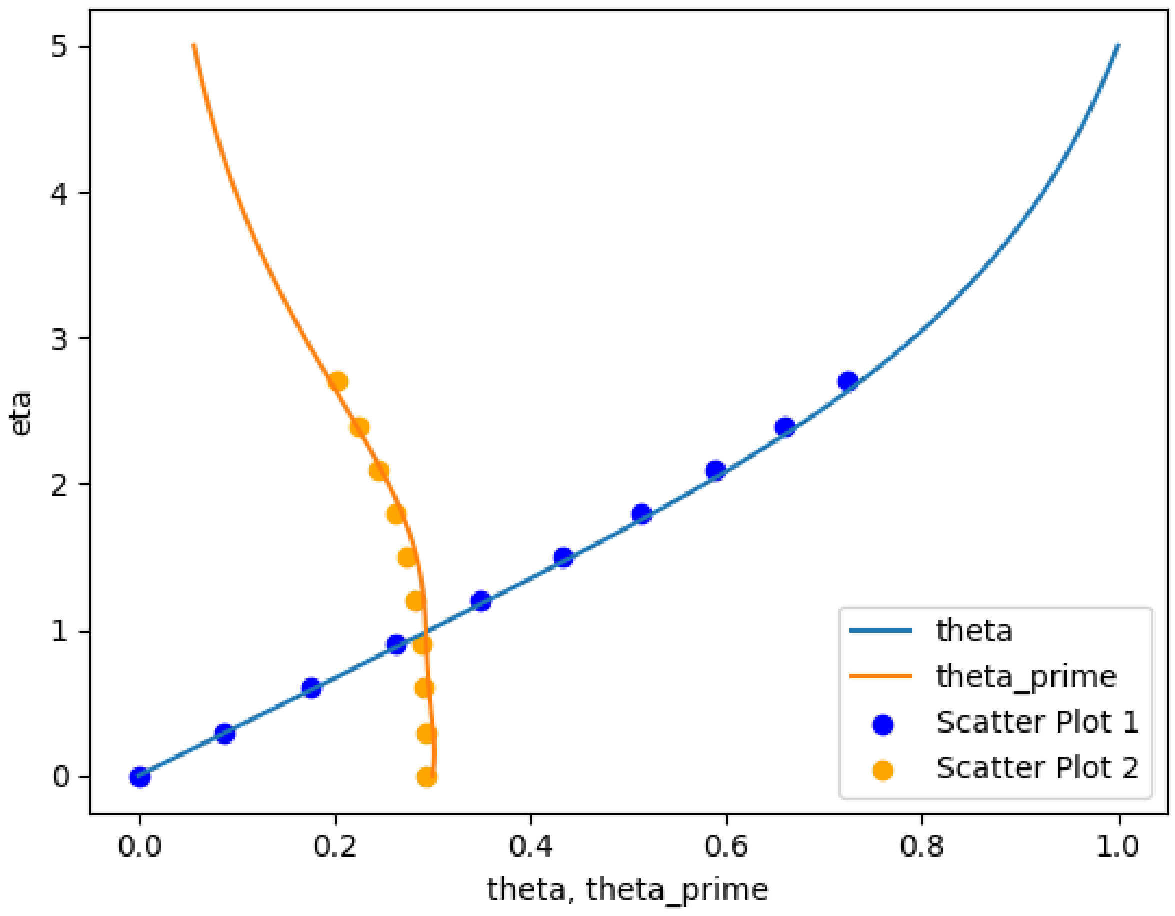

Figure A.15: Plot of $f, \frac{d\hat{f}}{d\eta}$ and $\frac{d\hat{f}^2}{d\eta^2}$ v/s the spatial variable η for Blasius Equation. The dots represent experimental values of the quantity whereas the lines represent model predictions.

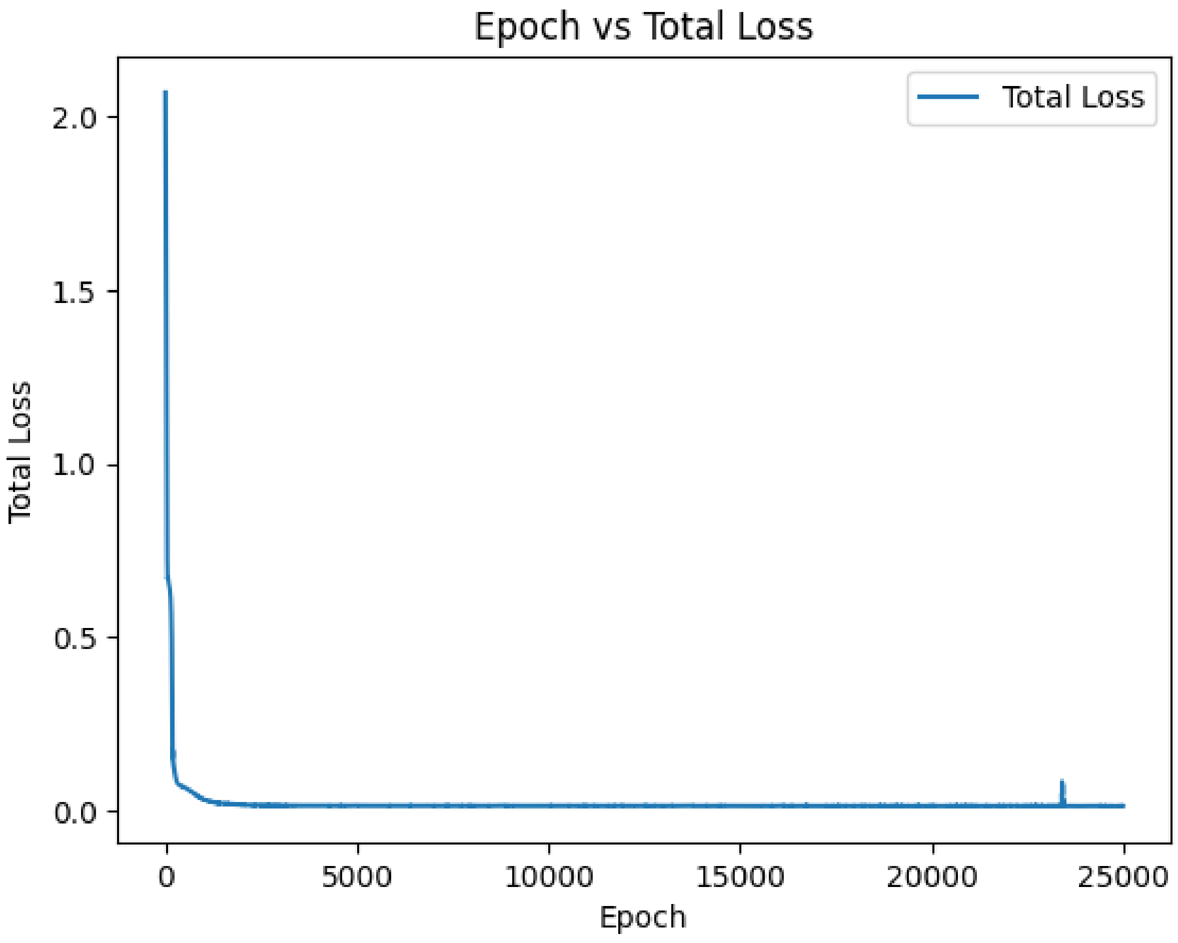

Figure A.16: Plot of Total loss vs Epoch for solving Flow problem

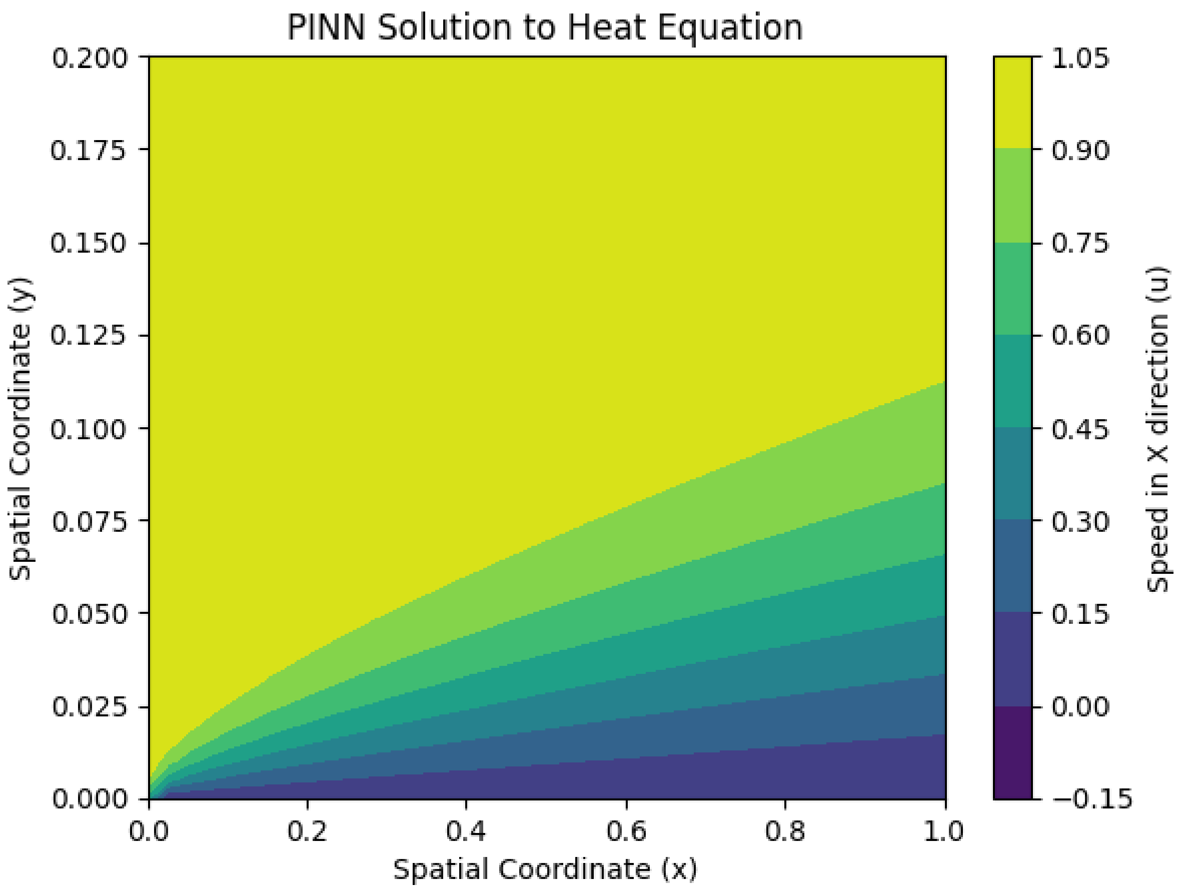

Figure A.17: Contour Plot of velocity in x direction (u) vs the spatial variable x and y for solving the flow problem.

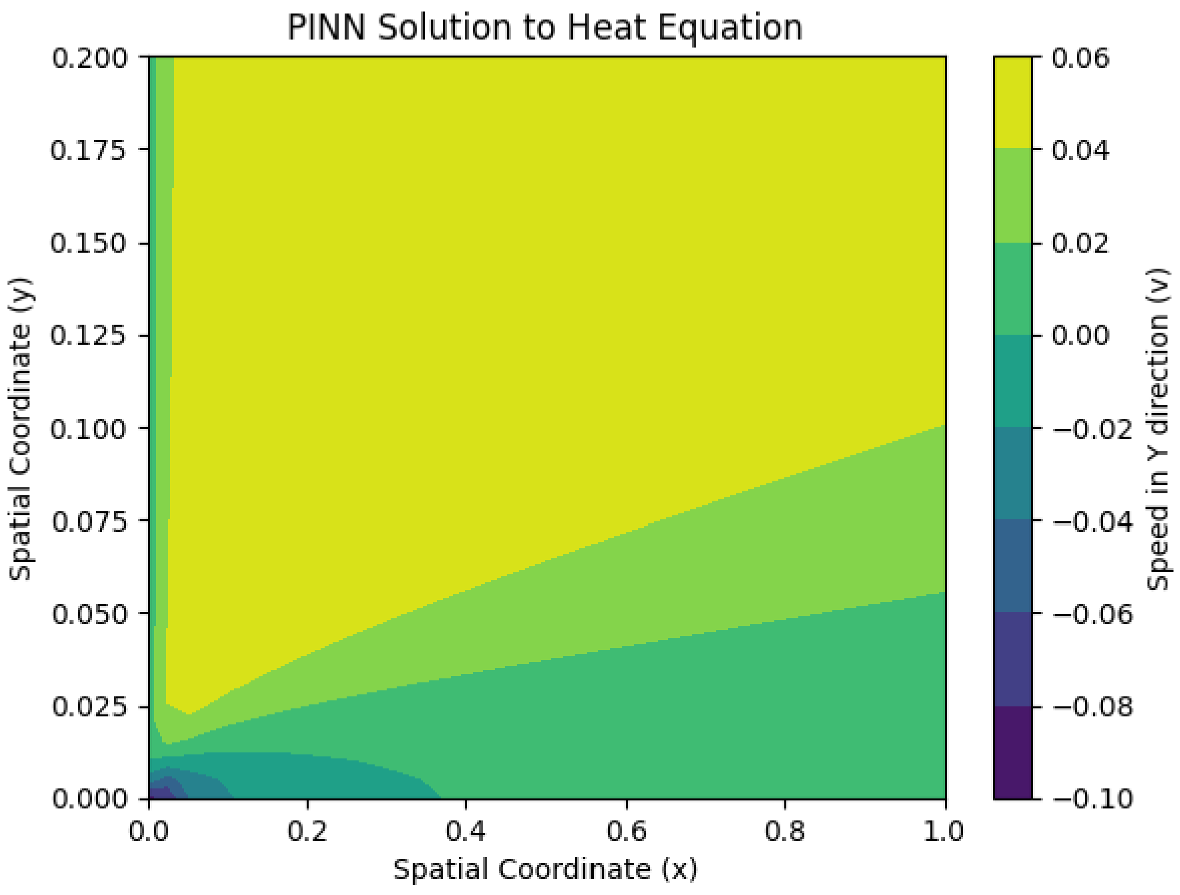

Figure A.18: Contour Plot of velocity in y direction (v) vs the spatial variable x and y for solving the flow problem.

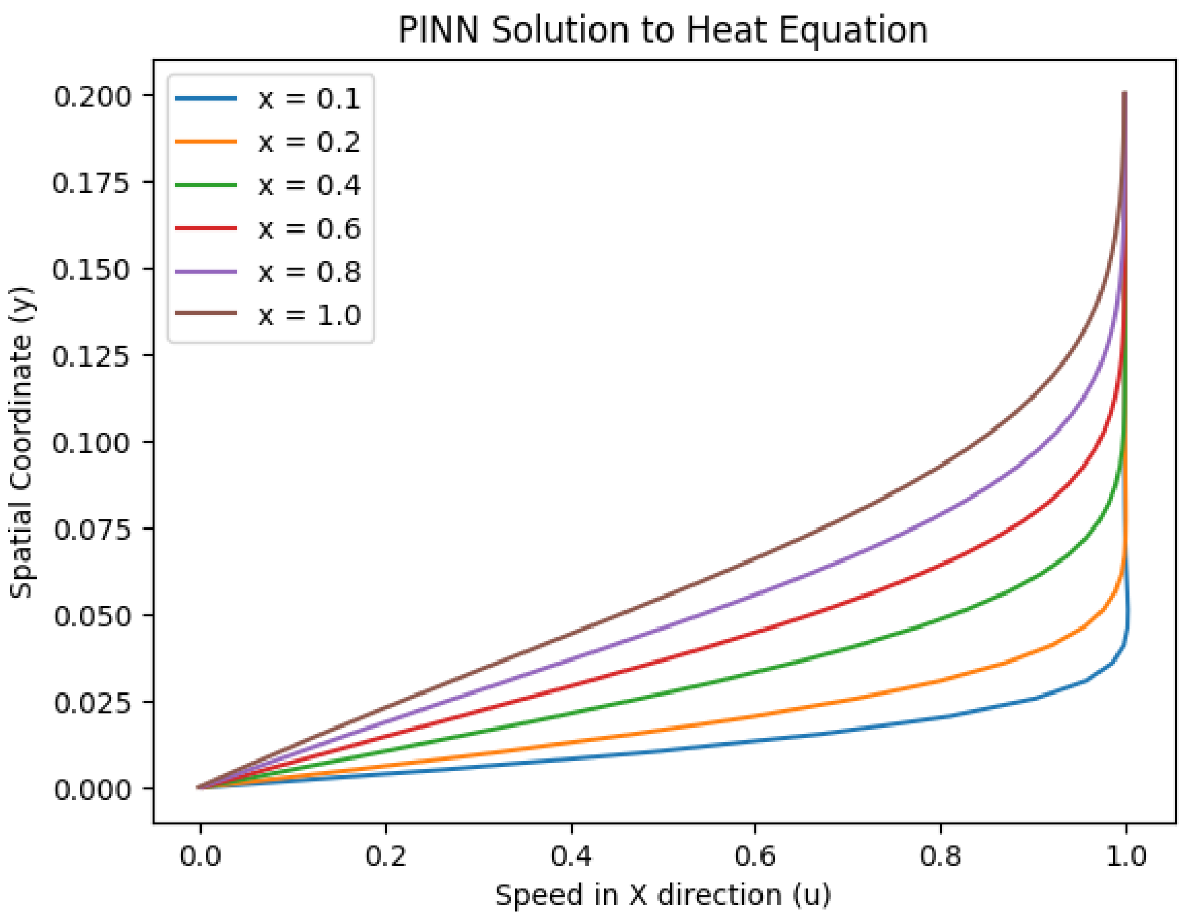

Figure A.19: Profile of Velocity in x direction (u) v/s x

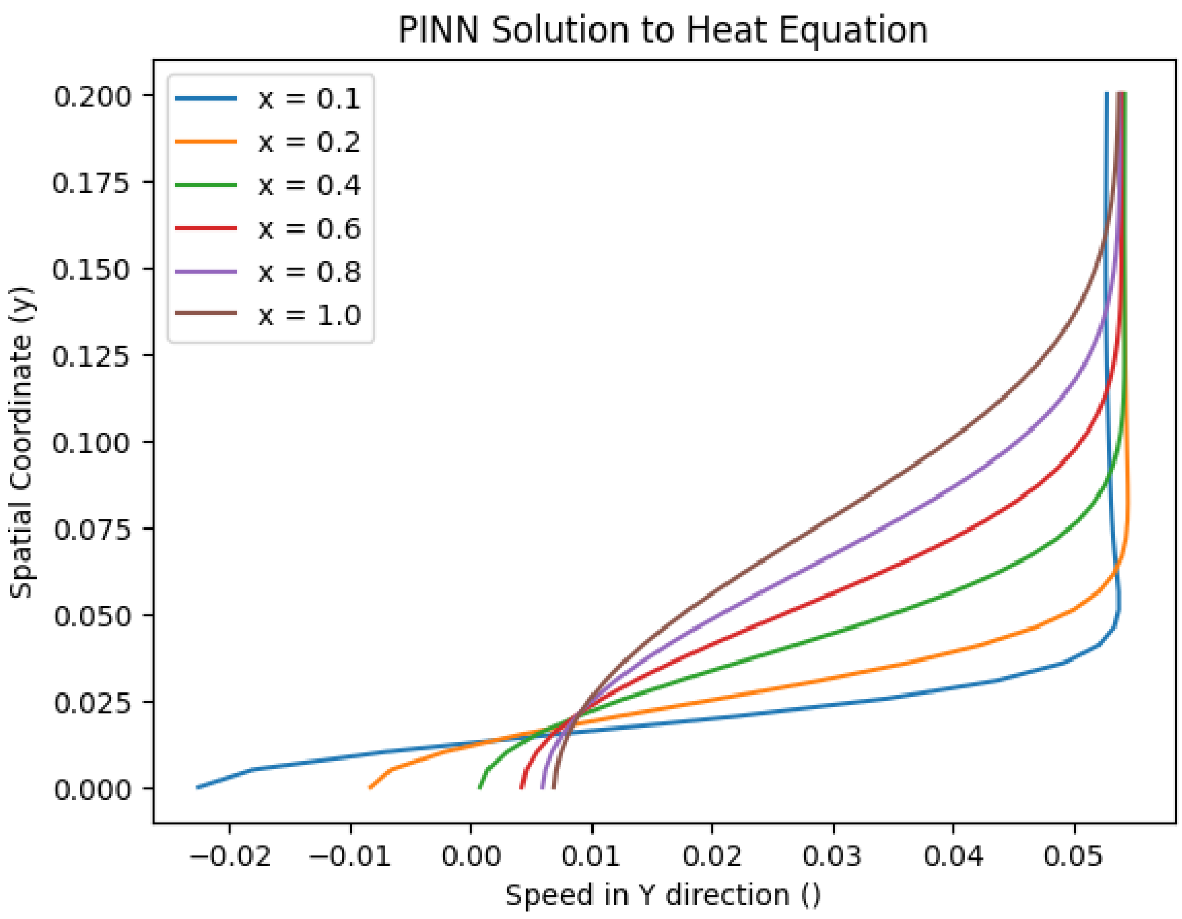

Figure A.20: Profile of Velocity in y direction (v) v/s x

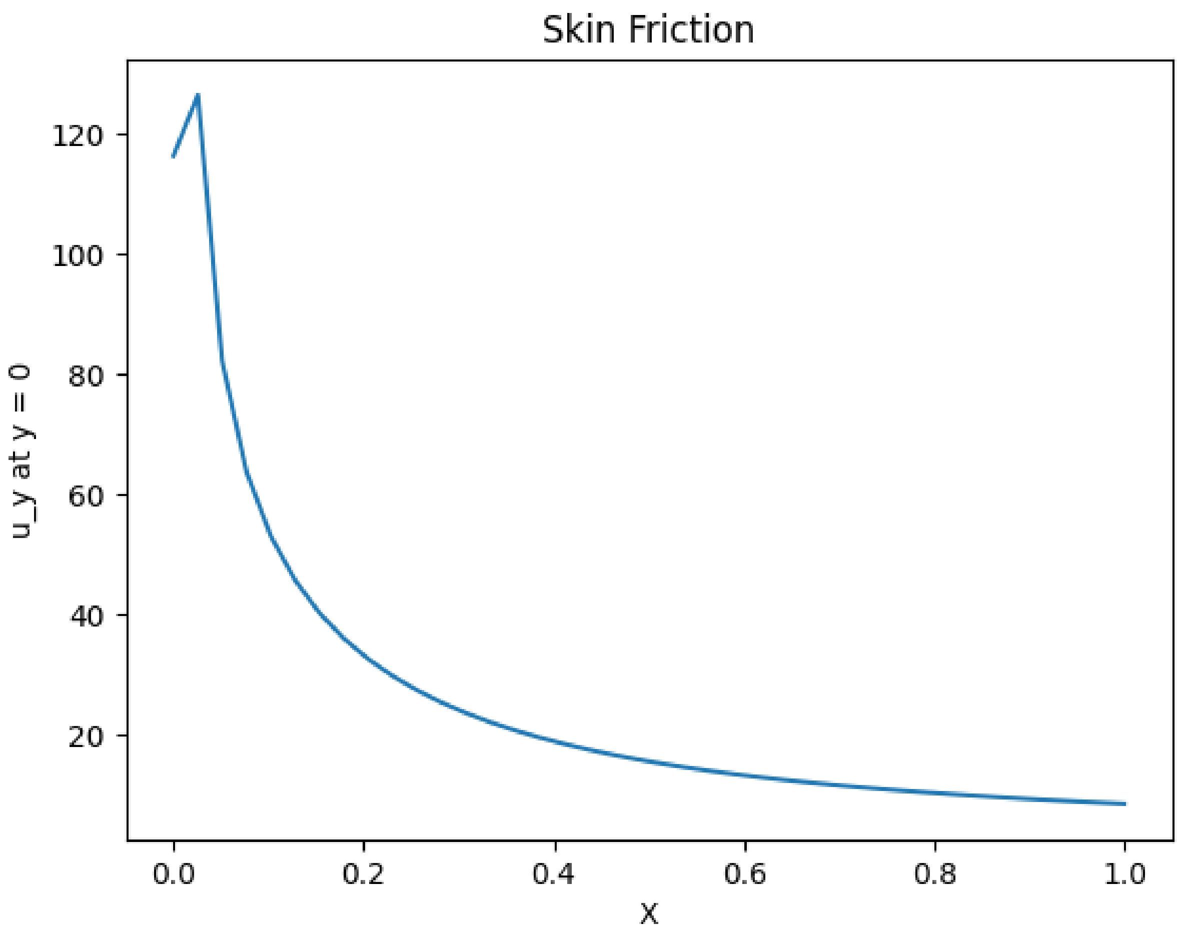

Figure A.21: Plot of skin friction vs the spatial variable *x*

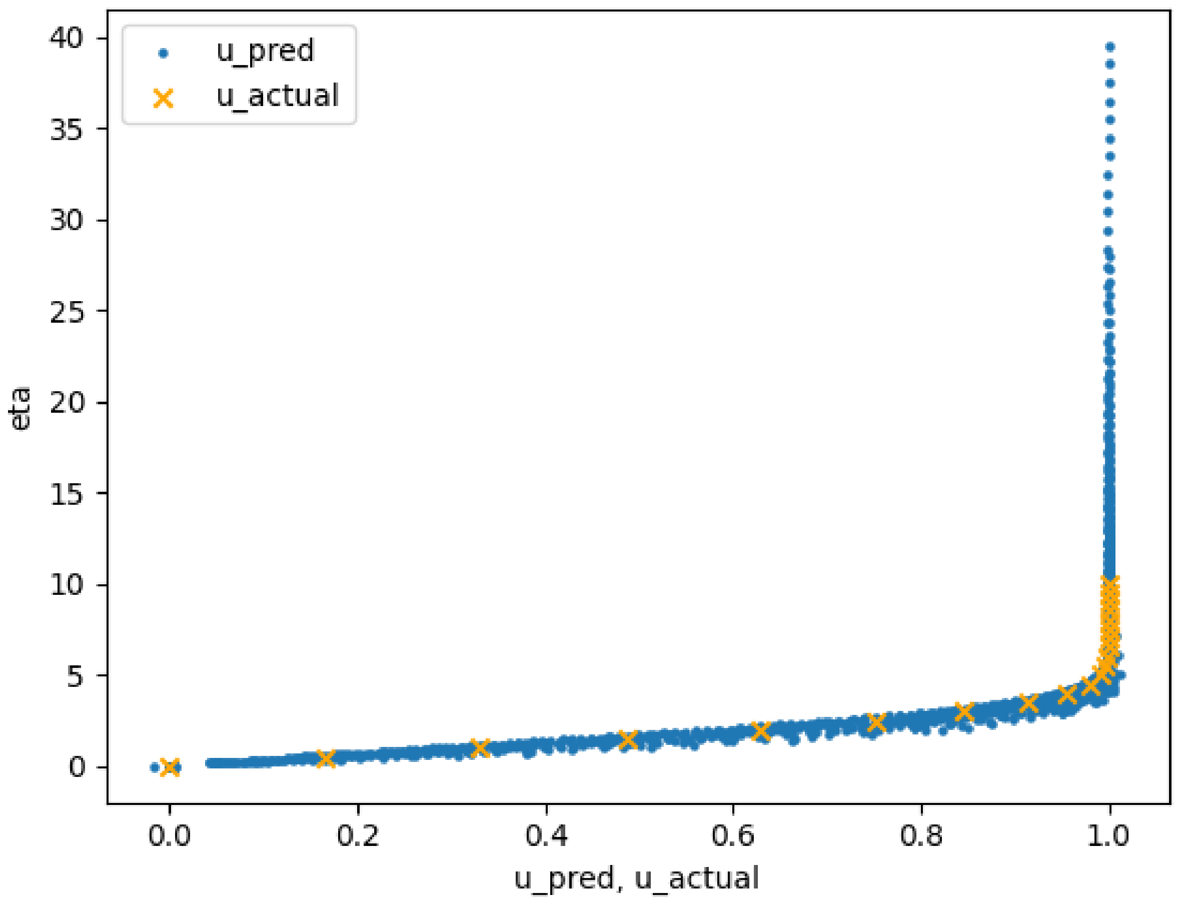

Figure A.22: Validation plot for solution of the flow problem

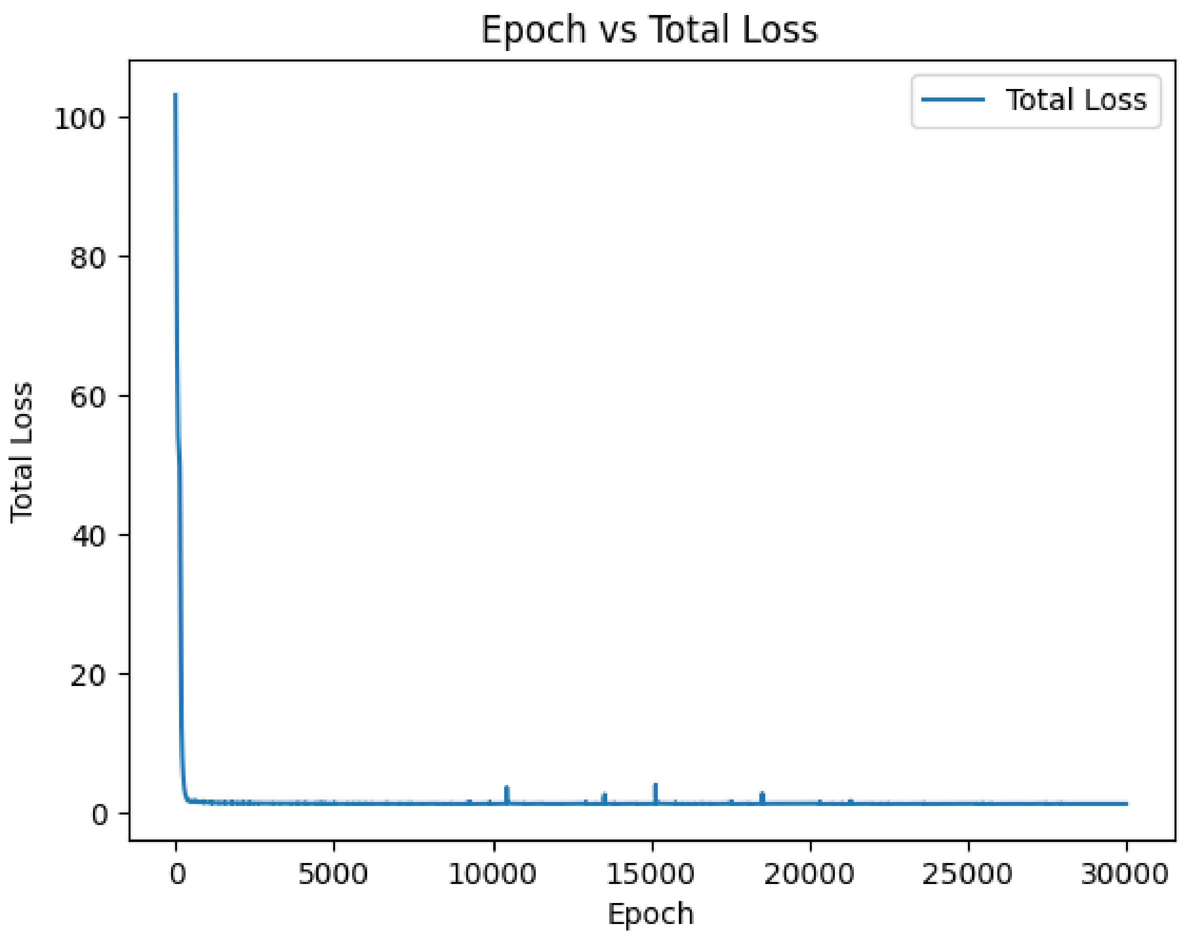

Figure A.23: Plot of Total loss vs Epoch for solving Convective Flow problem

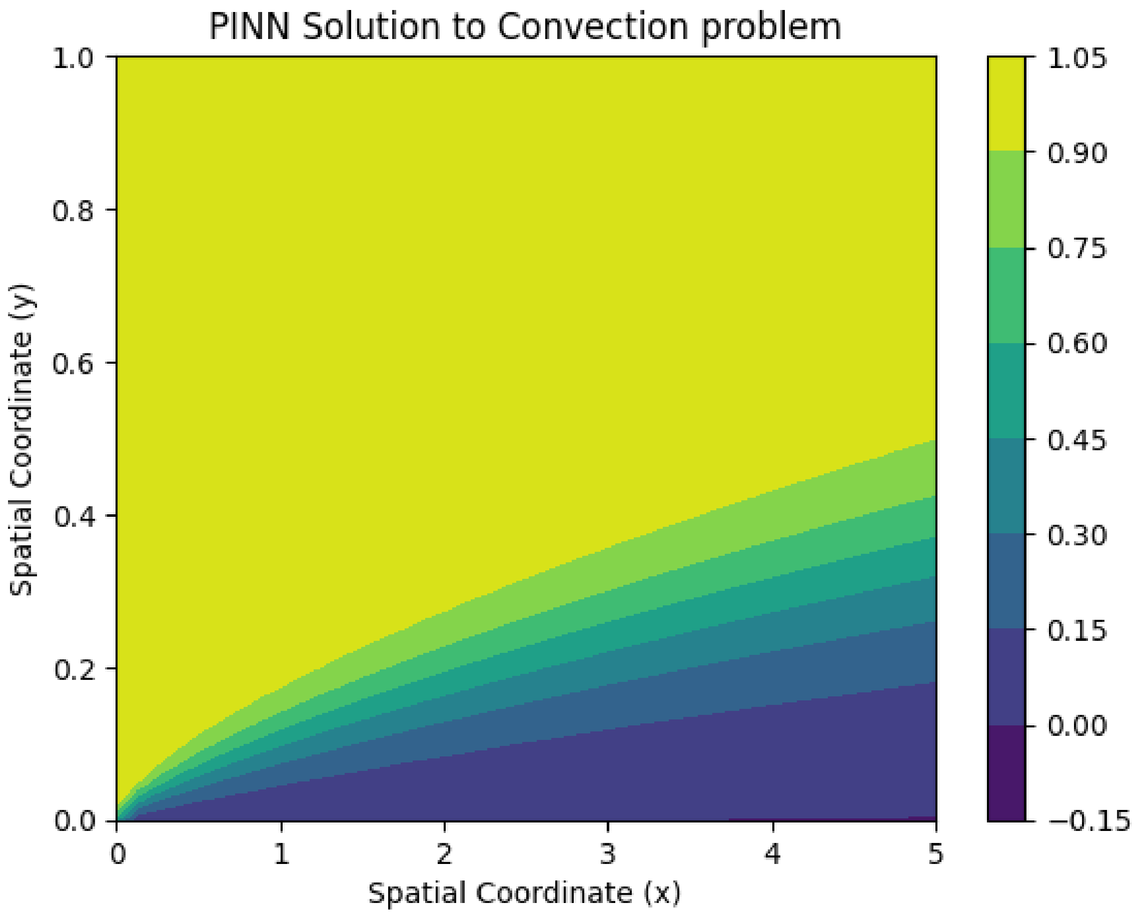

Figure A.24: Contour Plot of velocity in x direction (u) vs the spatial variable x and y for solving the flow problem.

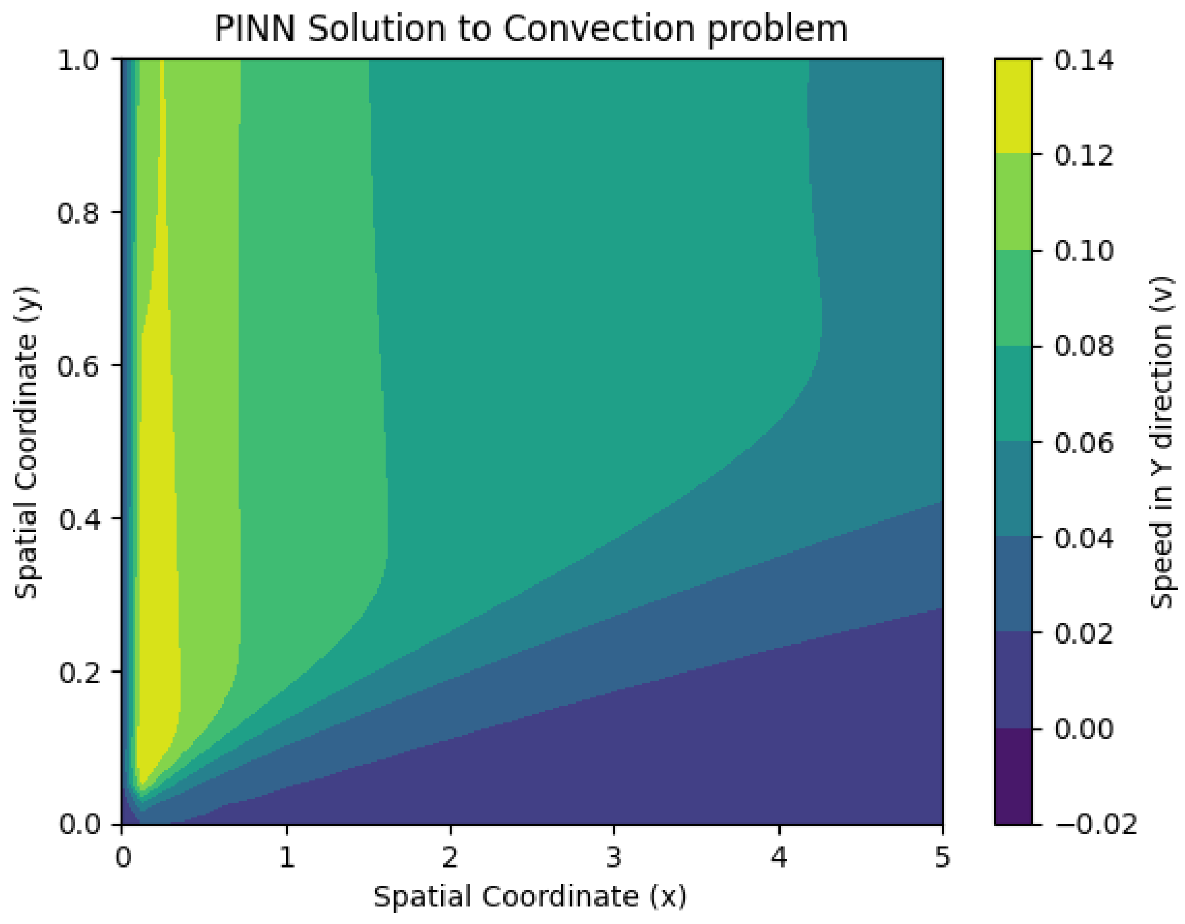

Figure A.25: Contour Plot of velocity in y direction (v) vs the spatial variable x and y for solving the flow problem.

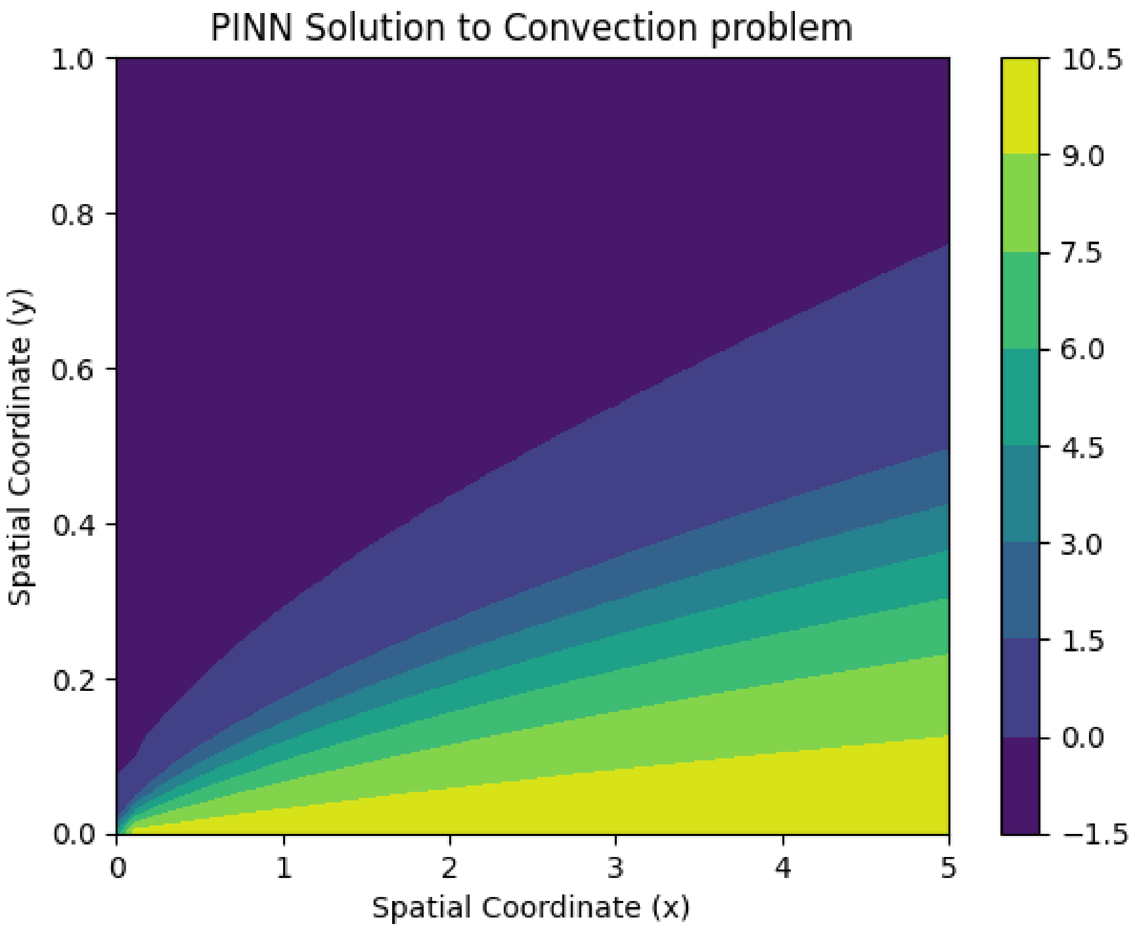

Figure A.26: Contour Plot of temperature T vs the spatial variable x and y for solving the flow problem.

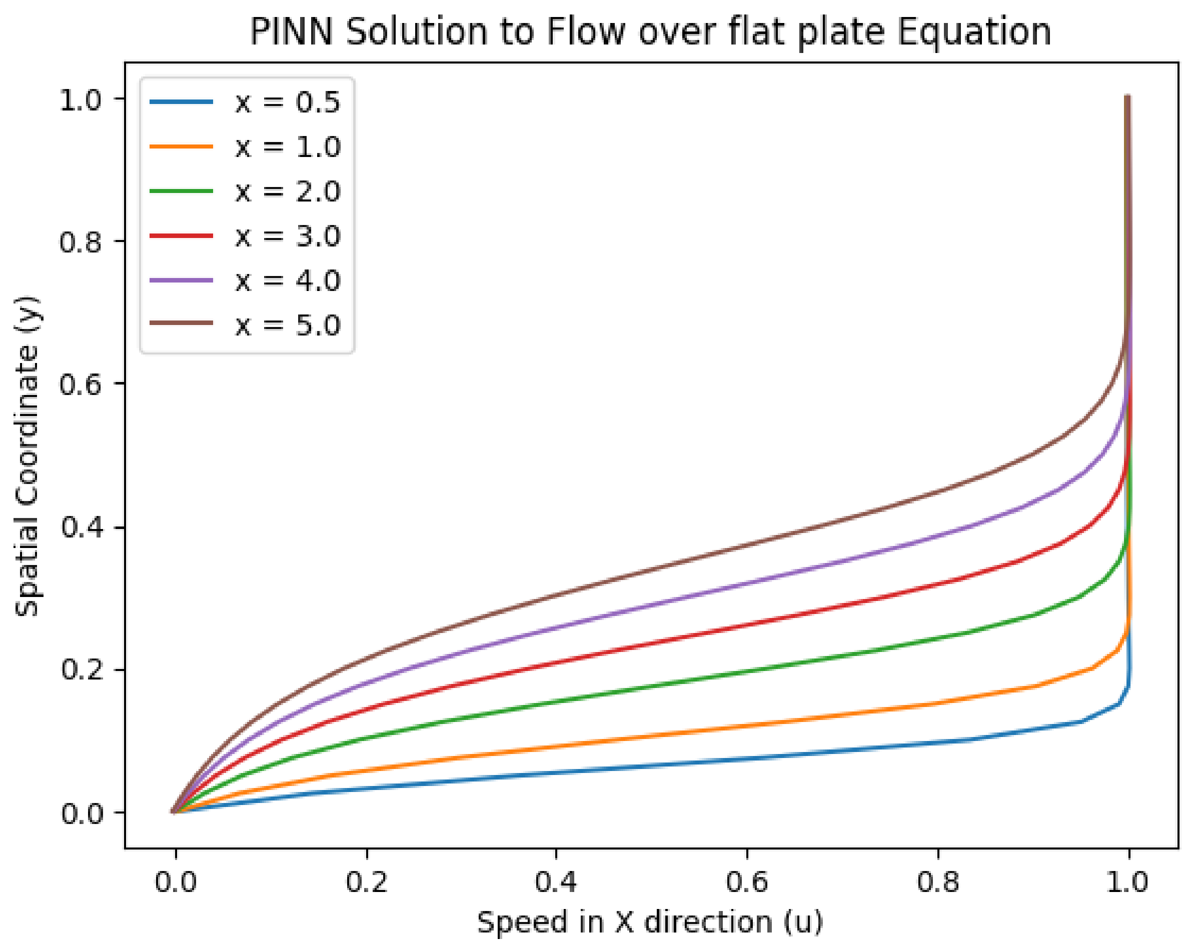

Figure A.27: Profile of Velocity in x direction (u) v/s x

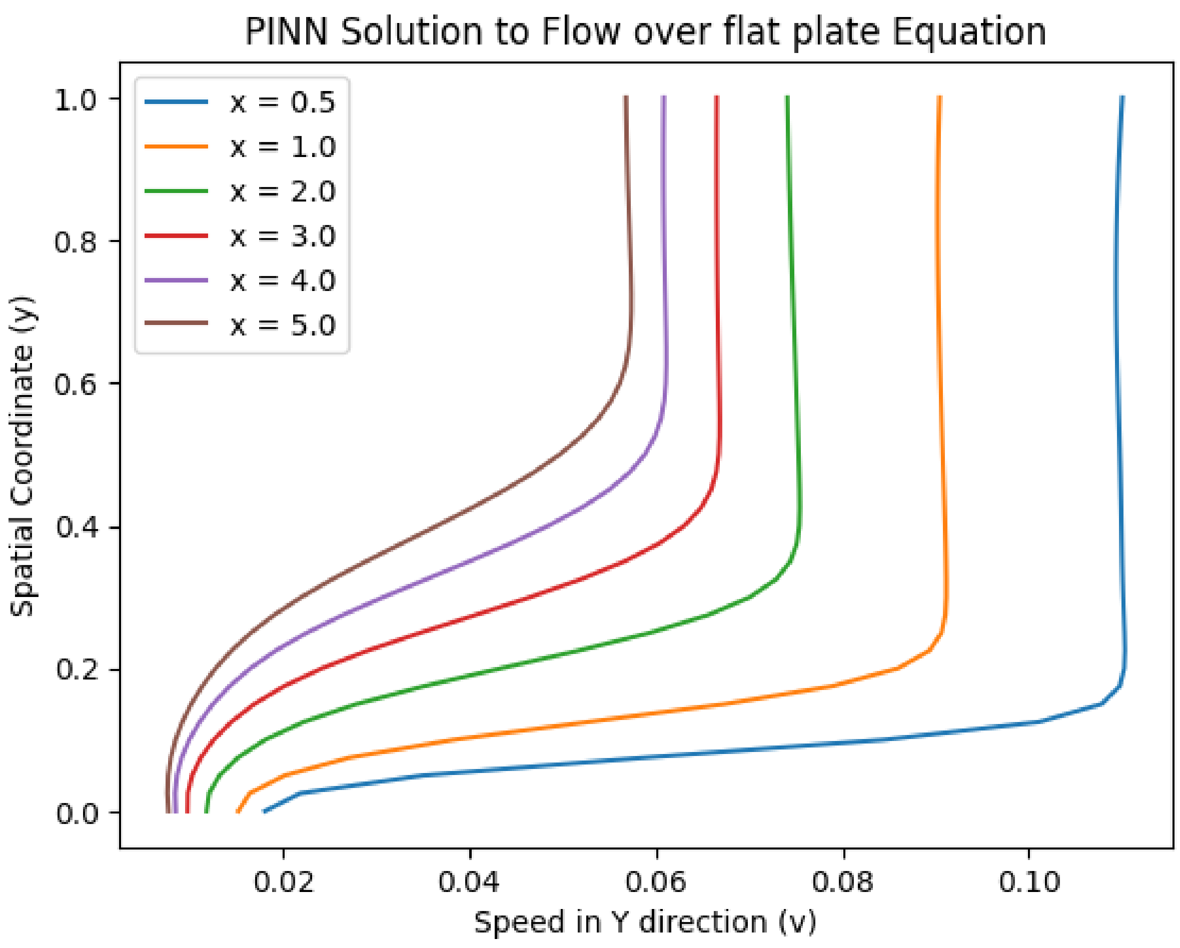

Figure A.28: Profile of Velocity in *y* direction (*v*) v/s *x*

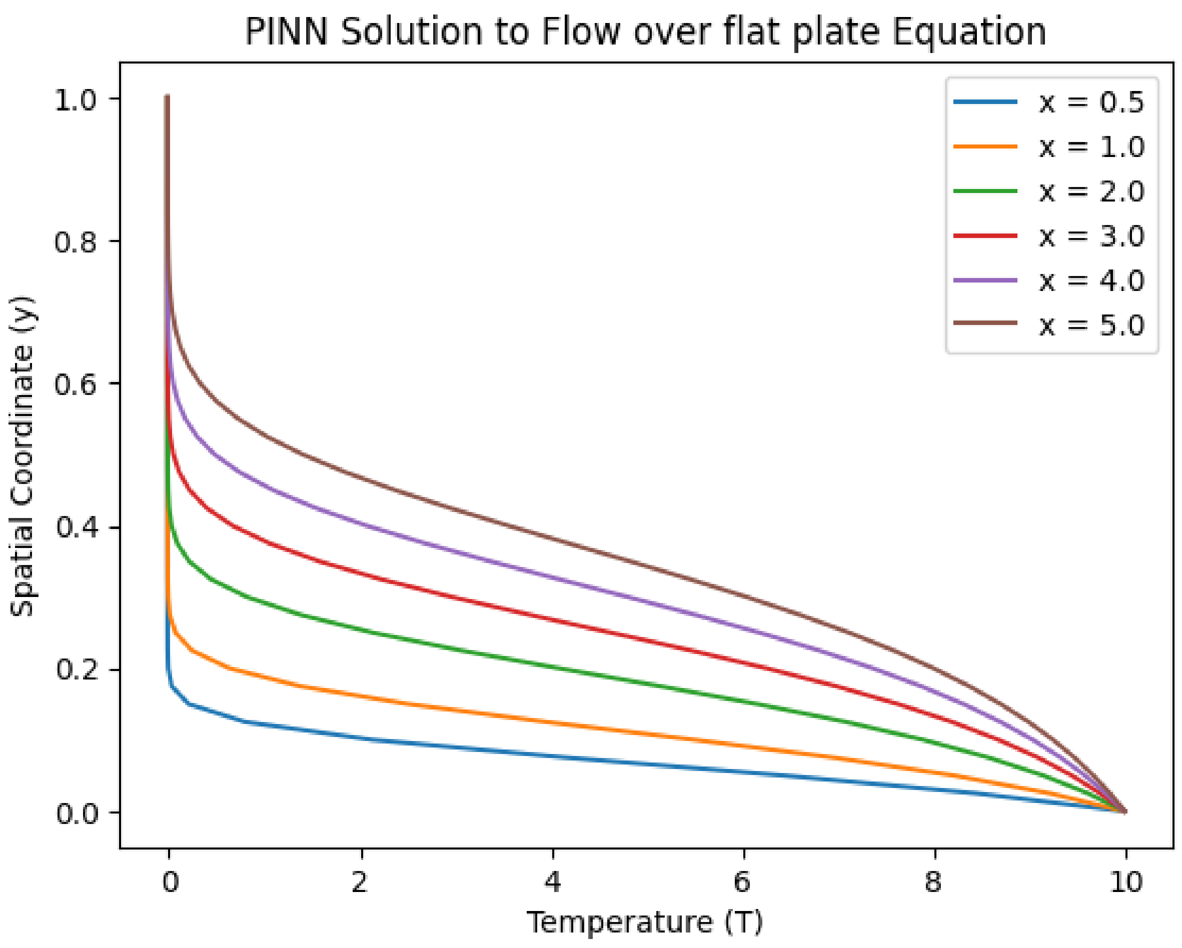

Figure A.29: Profile of Temperature (T) v/s x

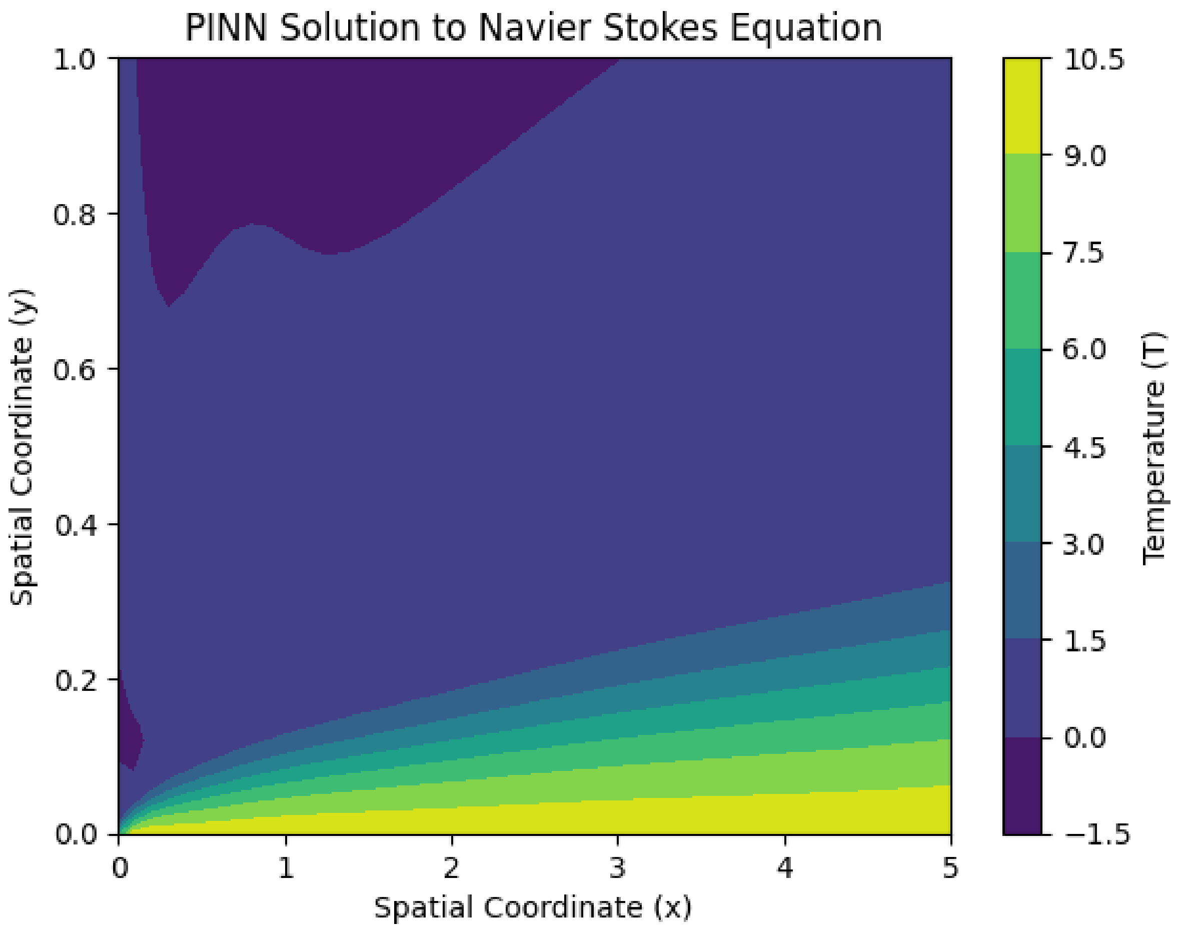

Figure A.30: Contour Plot of velocity in x direction (u) vs the spatial variable x and y for solving the flow problem.

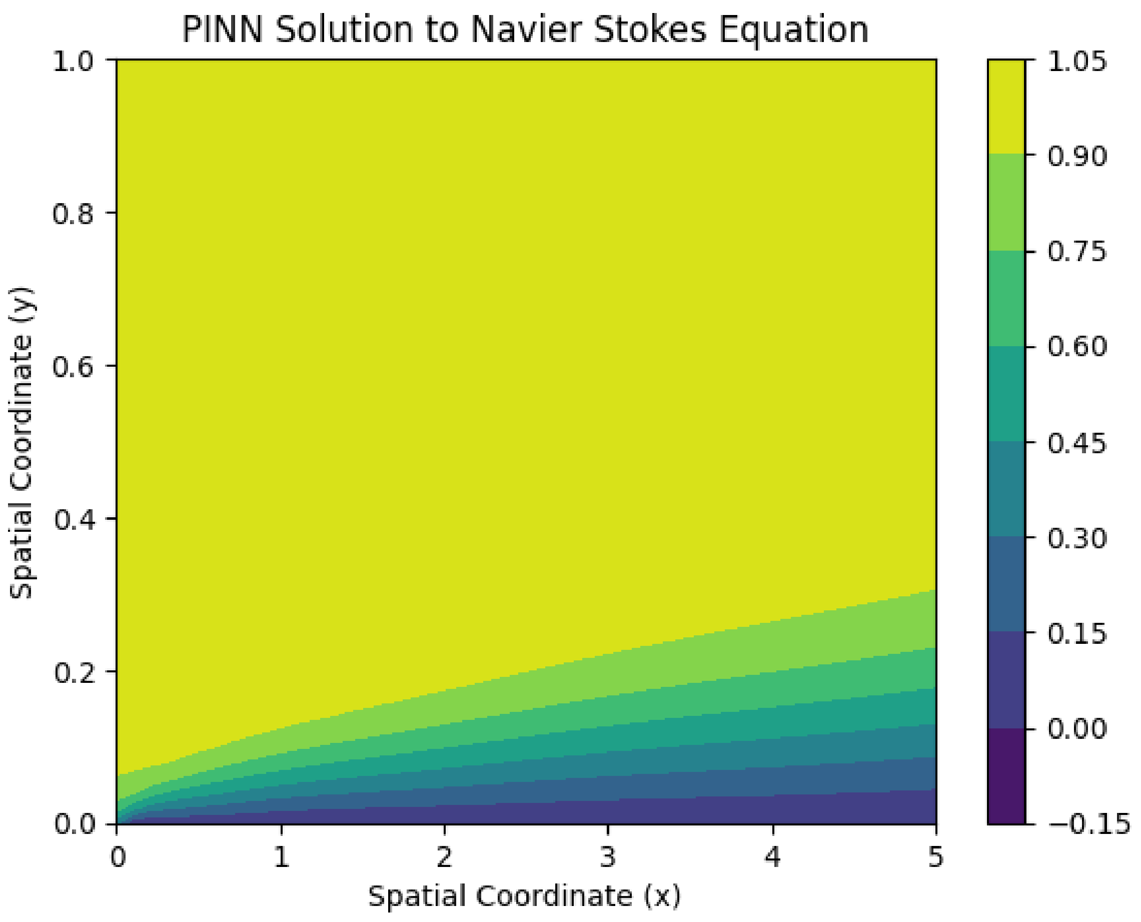

Figure A.31: Contour Plot of velocity in y direction (v) vs the spatial variable x and y for solving the flow problem.

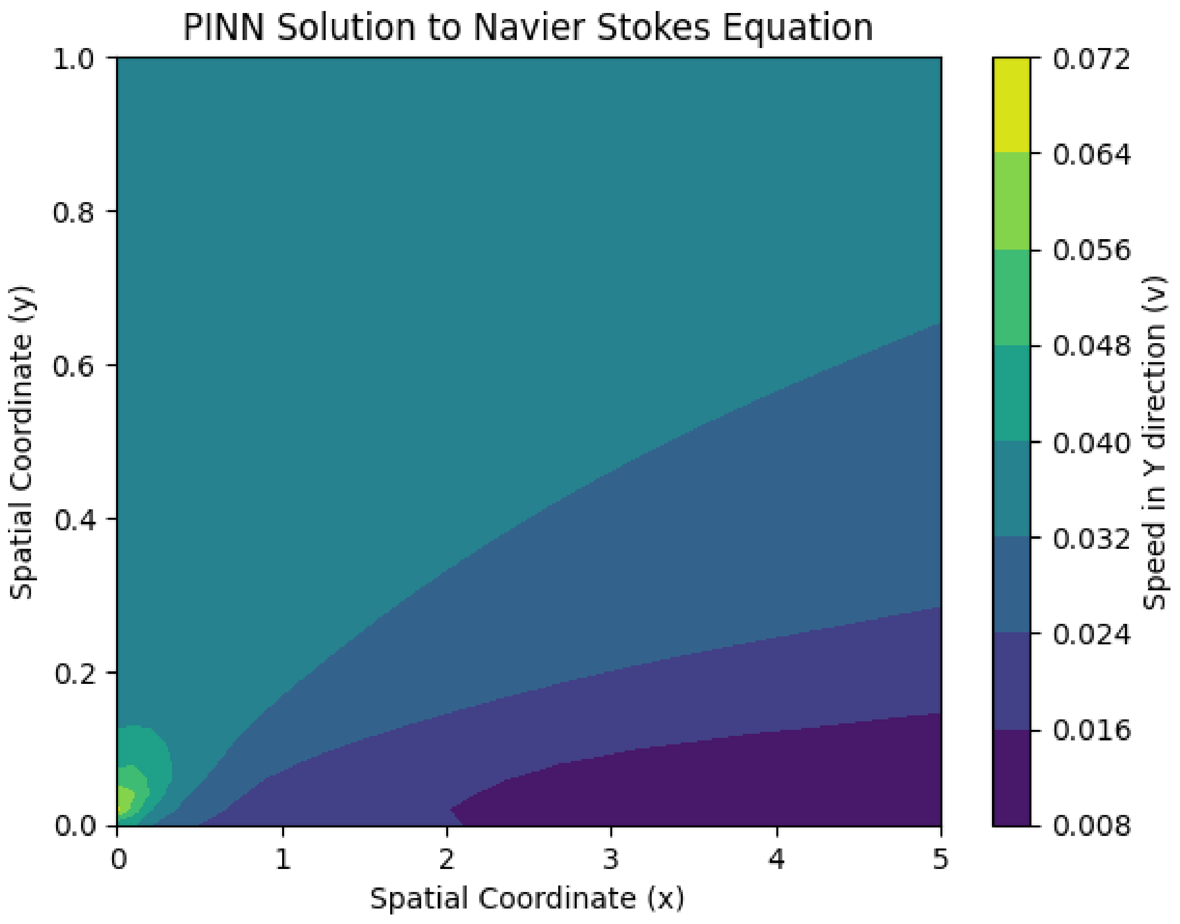

Figure A.32: Contour Plot of temperature T vs the spatial variable x and y for solving the flow problem.

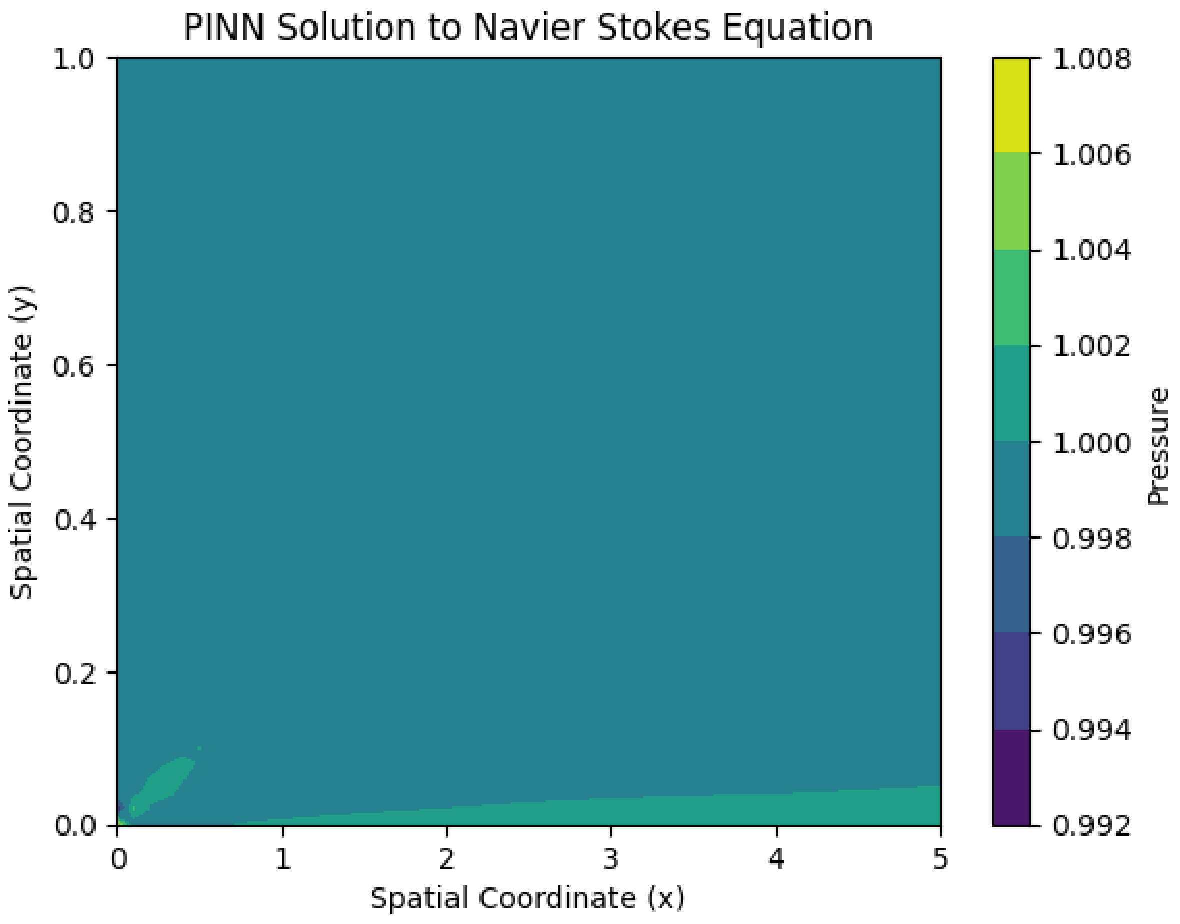

Figure A.33: Contour Plot of Pressure P vs the spatial variable x and y for solving the flow problem.

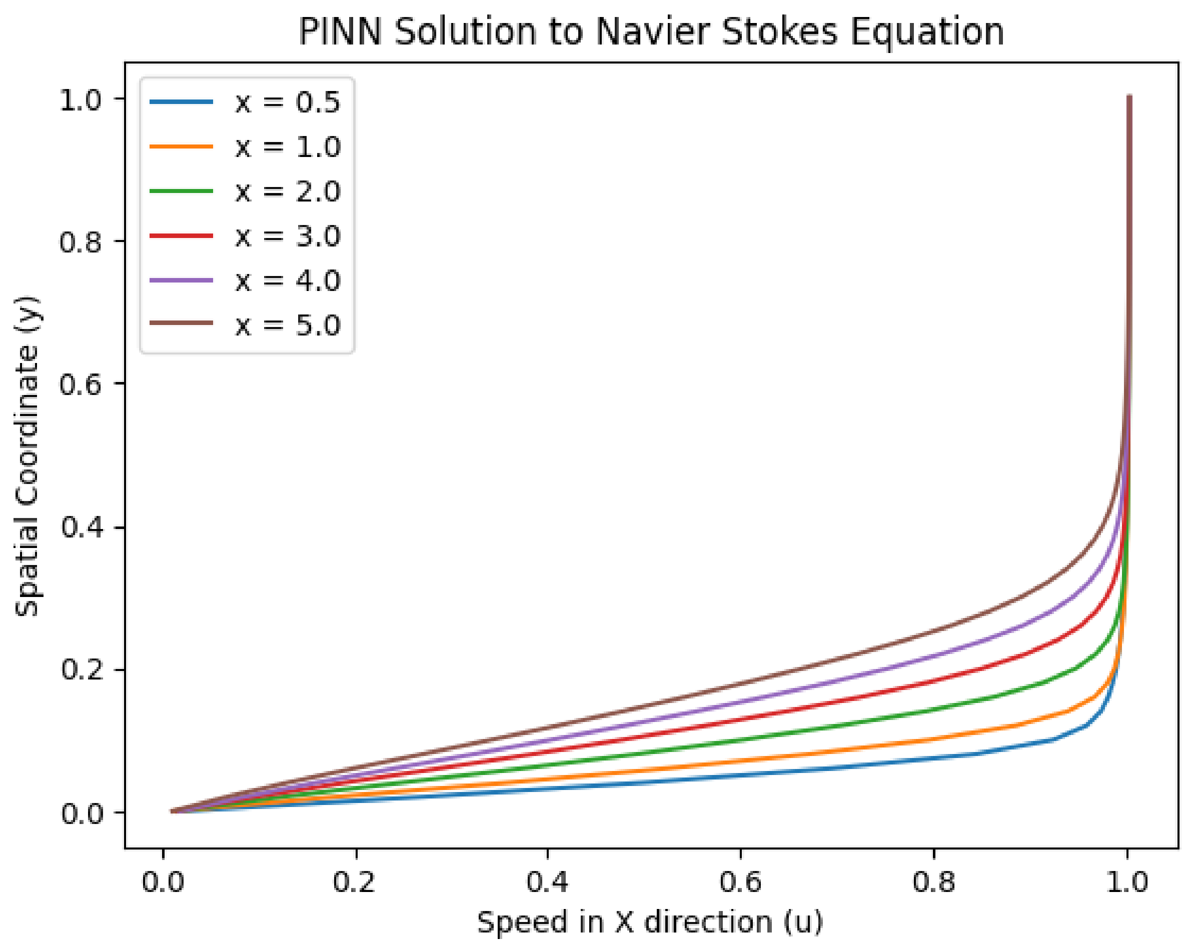

Figure A.34: Profile of Velocity in x direction (u) v/s x

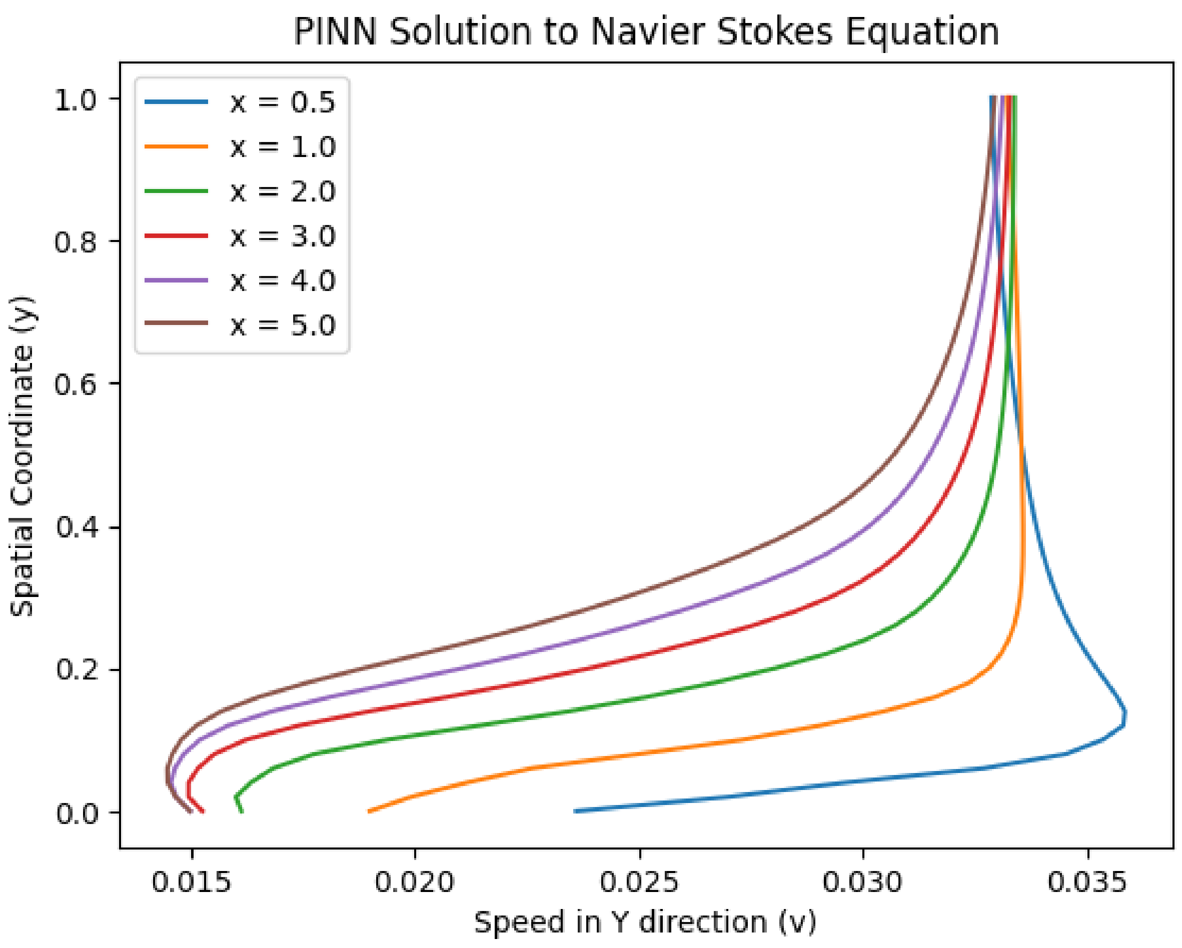

Figure A.35: Profile of Velocity in y direction (v) v/s x

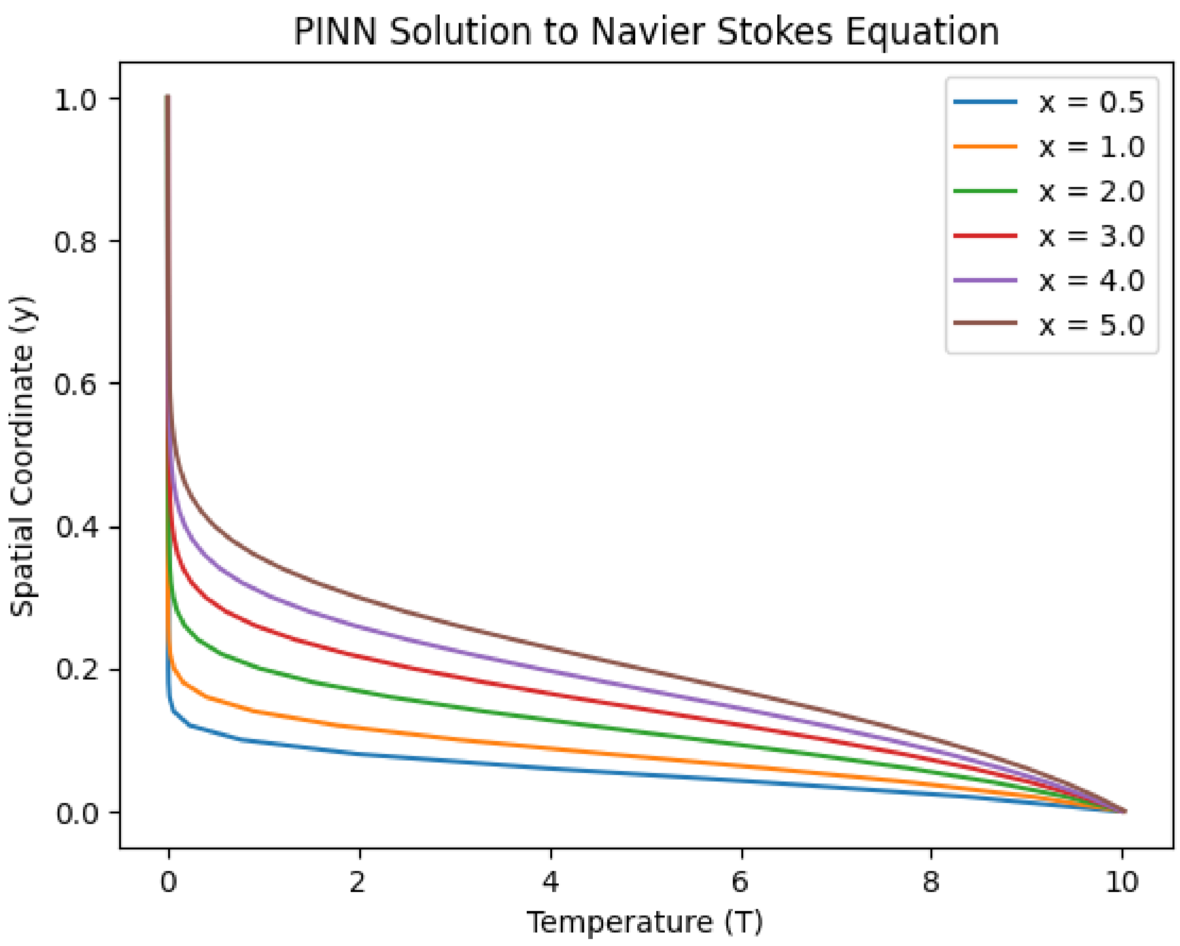

Figure A.36: Profile of Temperature (T) v/s x

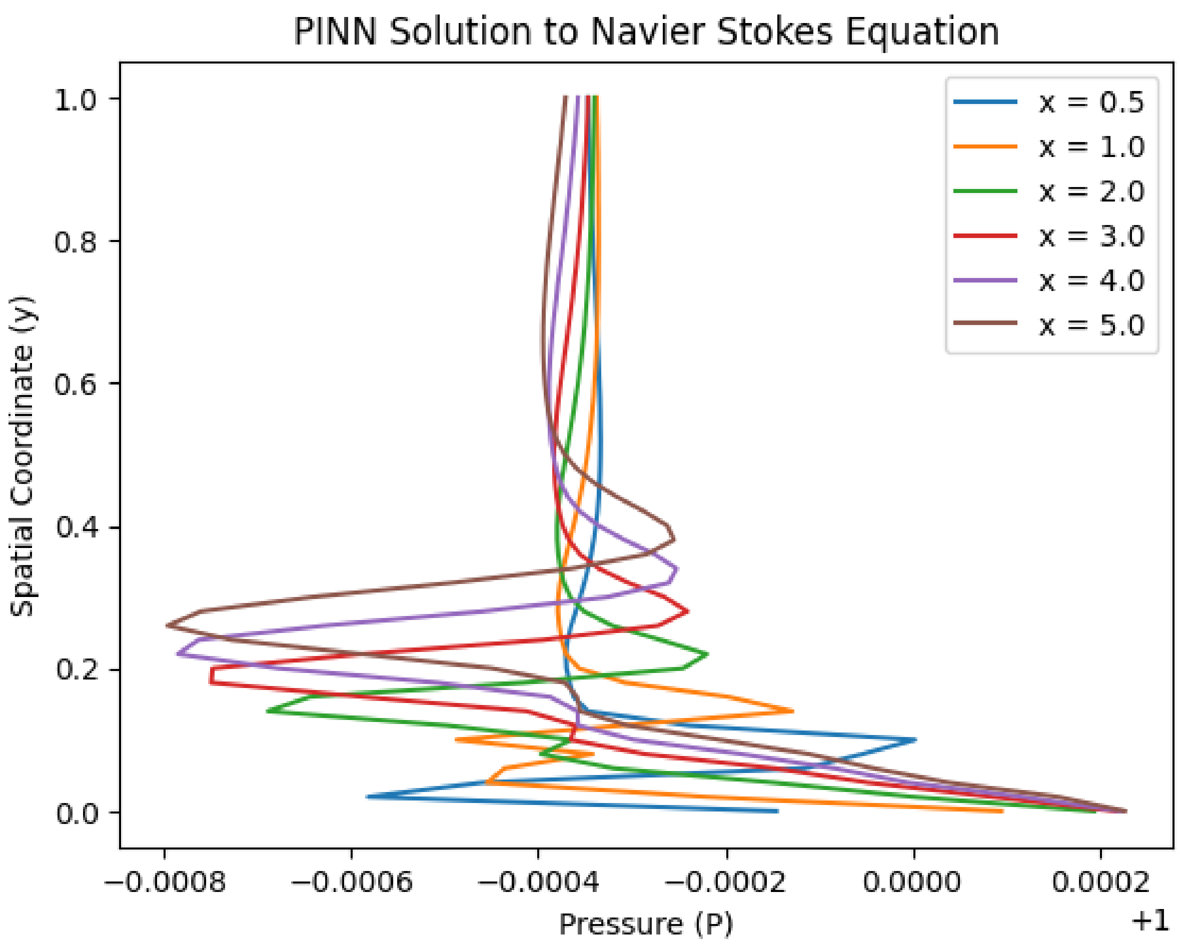

Figure A.37: Profile of Pressure (*P*) v/s *x*

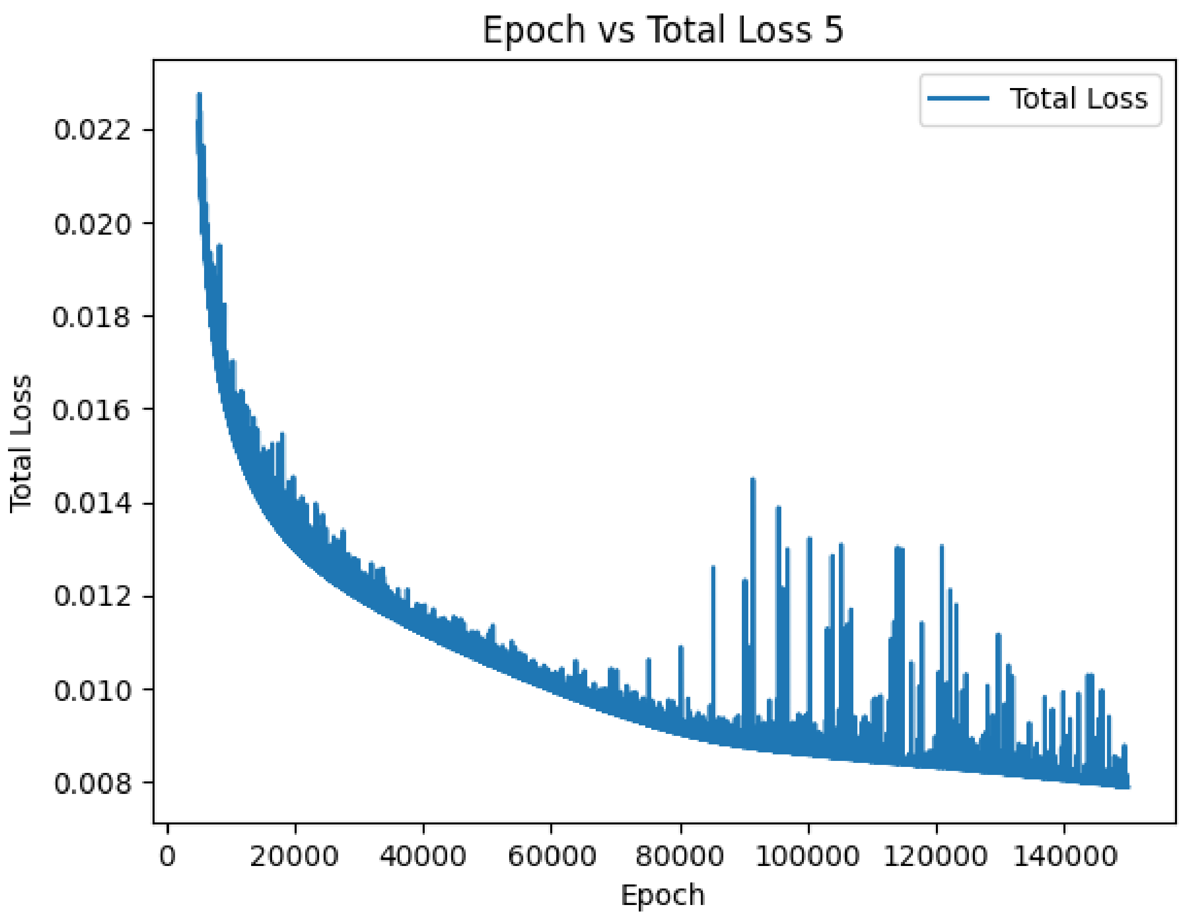

Figure A.38: Plot of Total loss vs Epoch for Lid driven cavity problem.

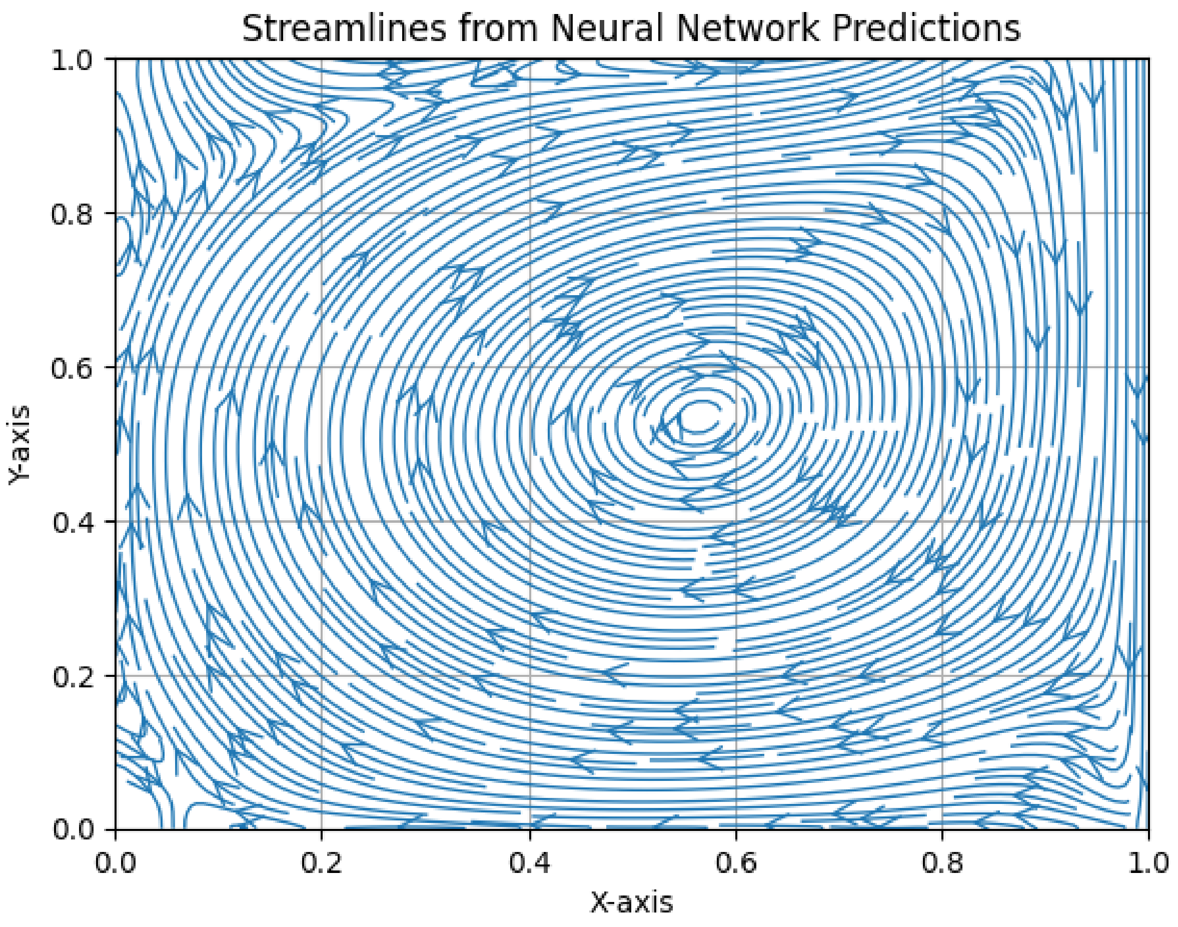

Figure A.39: Stream line plot of the Lid Driven Cavity

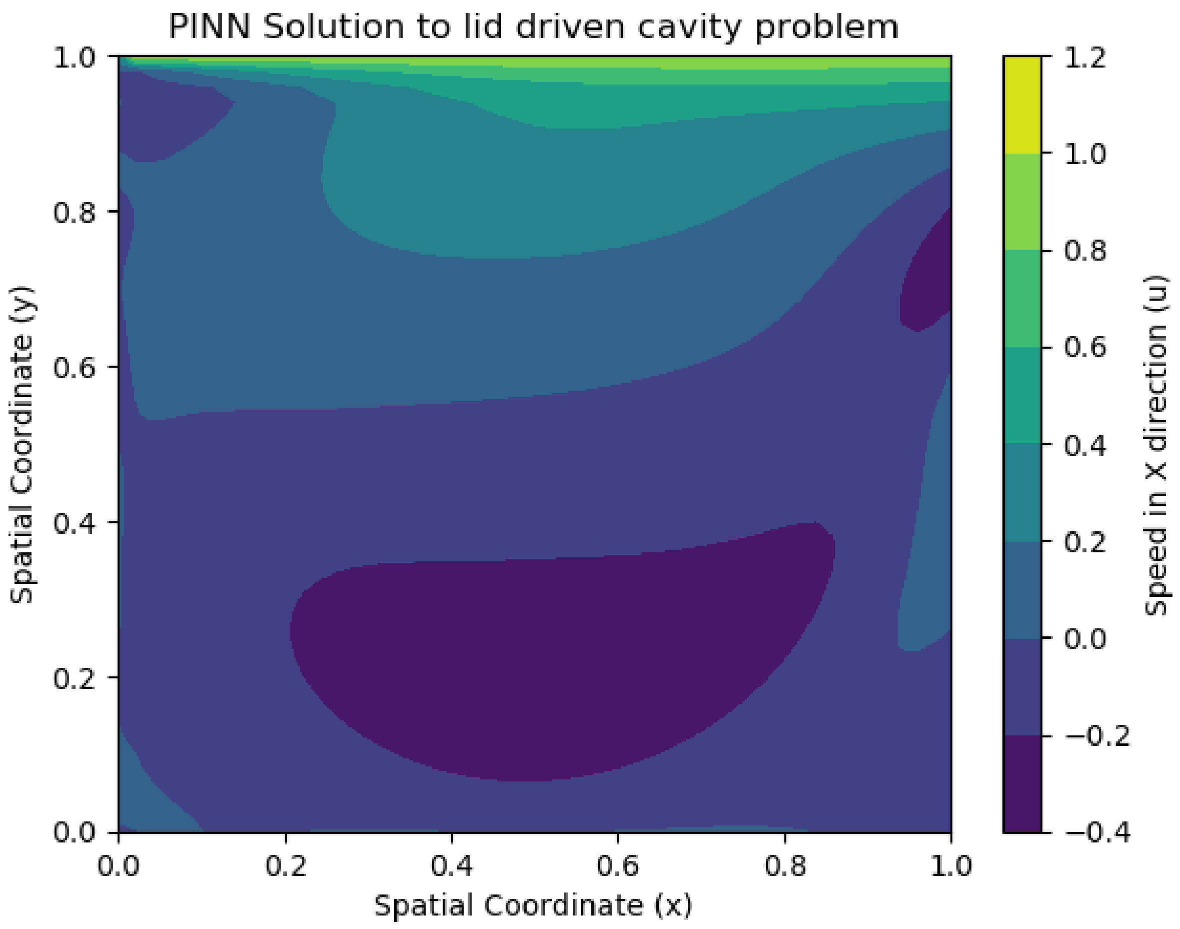

Figure A.40: Contour Plot of velocity in *x* direction (*u*) vs the spatial variable *x* and *y* for Lid driven cavity problem.

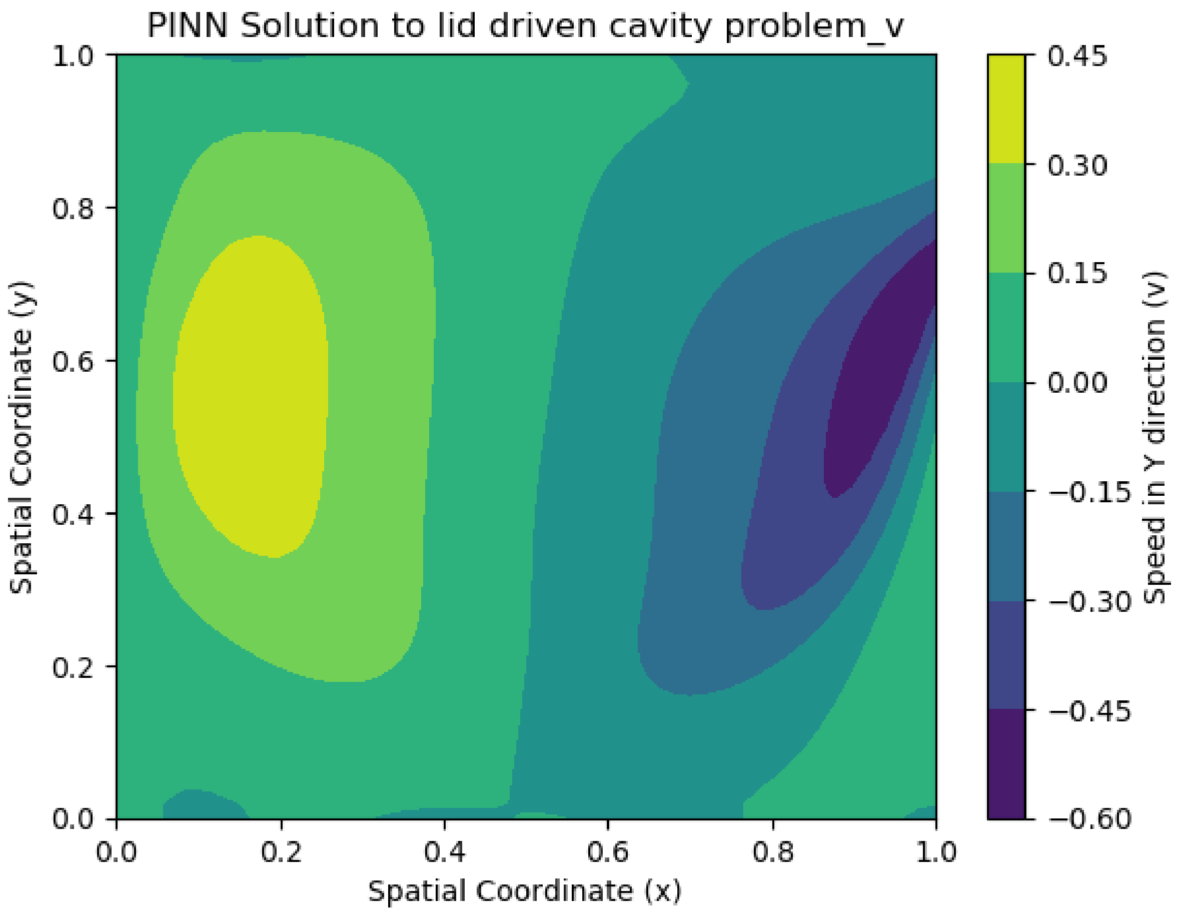

Figure A.41: Contour Plot of velocity in y direction (v) vs the spatial variable x and y for Lid driven cavity problem.

Appendix B

Tables

Number of Training samples taken (N)	Standard deviation (σ)	Prandtl Number (Pr)
1000	0	0.698195
100	0	0.698209
10	0	0.692760
10	0.001	0.697747
10	0.01	0.656837
100	0.01	0.611053

Table B.1: The predicted values of Prandtl Number Pr for different number of training samples N and values of standard deviation σ

References