PINN for Heat Conduction PDEs

The general formula for heat conduction problems is:

Governing Differential Equation:

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

Boundary conditions

$x = 0 : T = T_s$

$As\ x \to \infty : T \to T_0$

Initial condition

$t = 0 : T = T_0$

Before defining the model architecture, there are a few other things we need to take into consideration. First of all, for each and every heat conduction problem there are some information given by the question, including initial temperature of the rod, length of the rod, the temperature at the top and bottom of the rod and the thermal diffusivity. In my project, I decided to get these variables as an input from user, so that the model can be trained on any given problem.

1) User inputs:
   Following variables are given by user an input: alpha, length of the rod, time interval, initial temperature of the rod and the temperature at the top and bottom of the rod.
2) Data Generation:
   The required data for training our model is generated using the information provided by user (inputs).
   - Collocation points: These points are specifically used while training PINNs, and are sampled randomly in the domain of the problem. Our model would be trained in a way that it satisfies the PDE in these points as closely as possible and tries to make the error 0. For my model I chose 500 collocation points.
   - Boundary points: These are located at the boundary of the rod and they enforce boundary conditions in a way that the model satisfies the

boundary conditions at these points. I set 100 boundary points for training. 100 boundary points means that we get points (0, t) and (L, t) where L is the length of the rod, for t = 0, …, 100. The reason behind this concept is that our neural network should have enough points to learn effectively.

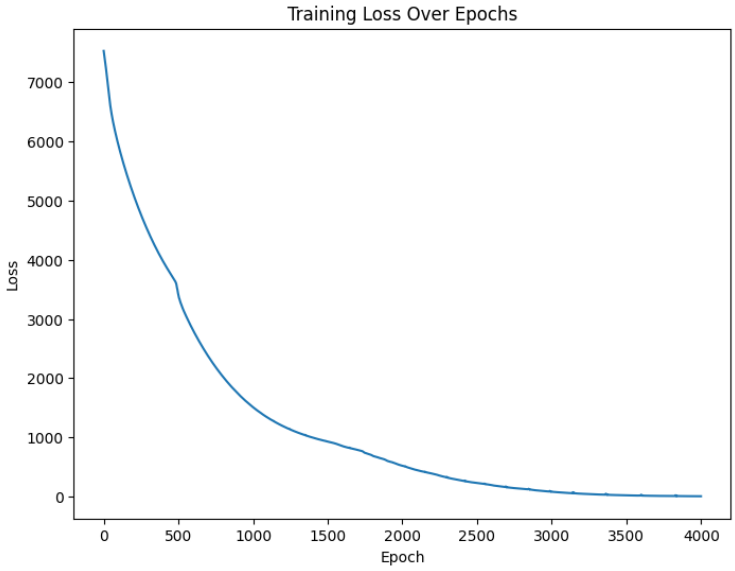
- Initial Condition points: These points are the temperature distribution of the rod, where t = 0. Using these points, we ensure that the network has correct starting points for training and the temperature distribution is preserved. I set 100 points for this one as well.

Using these points, we generate the training data based on the information we got from user.

3) Model Architecture:
The model has two inputs, x and t, which represent the position and time. Then I used 3 hidden layers each with 50 neurons and Tanh as the activation function. The reason behind choosing Tanh is that it outputs both positive and negative values and helps with predicting physical quantities. Then I used Adam as the optimizer with 0.001 learning rate. Adam is a very popular optimizer and is used widely in different NNs.
    - **Loss function:** In my opinion, the most important part of PINNs is its loss function. Since we enforced three type of points on the model, our total loss is comprised of 3 parts.
    - **PDE Loss:** This loss ensures that our model respects and satisfies the PDE equation at collocation points, and is computed using the temperature predicted by the model and the derivatives: u_x, u_t and u_xx. We use the Mean Squared Error to get the error for all collocation points.
    - **Boundary Loss:** This loss ensures that the boundary temperature predicted by the model satisfies the boundary conditions. We get the MSE of predicted u and the actual u at boundary points.
    - **Initial Condition Loss:** This loss ensures the initial temperature distribution is satisfied

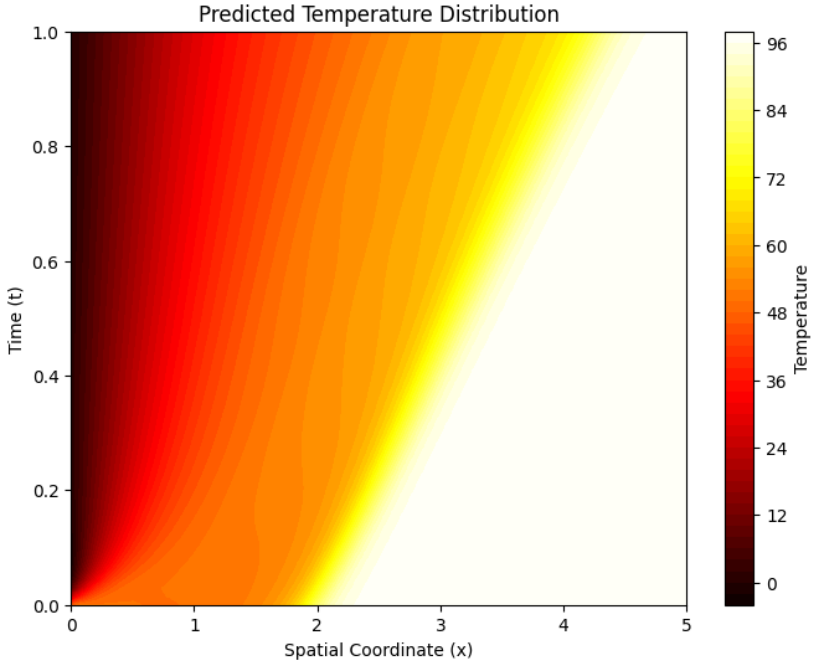
The sum of these 3 values makes up the total loss that is used for training the model

4) Training Loop: I trained the model for 4000 epochs and I tracked the loss during training.



It is evident that the training loss has dropped at first and after 2000 epochs it slowly converges to 0.

5)Result:



This heat map shows the temperature distribution in the given time interval and for the given rod, which was predicted by the model.