APPROACH FOR SOLVING THE PROBLEM

# Burgers Equation

To solve the given partial differential equation, I implemented a Physics-Informed Neural Network (PINN) using the DeepXDE library. The model was trained to approximate the solution u(x,t) by minimizing the physics-based loss derived from the PDE, while also ensuring that the initial and boundary conditions are satisfied.

Highlights of the Approach:

* + *Network Design:*

A deep neural network with three hidden layers, each containing 50 neurons, was used. The tanh activation function was chosen for its

smoothness and effectiveness in approximating continuous functions.

* + *Boundary Condition Handling:*

An output transformation of the form (1 − 𝑥2). 𝑦 was applied to naturally enforce the zero boundary values at the domain ends.

* + *Loss Weight Adjustment:*

To improve the model’s accuracy at the start of the simulation, the loss

term corresponding to the initial condition was given greater importance during training.

# Navier Stokes Equation

This system describes the behaviour of 2D fluid flow, capturing both the velocity components (u,v)and pressure p, while maintaining incompressibility through the continuity equation. I reproduced the classic flow-over-cylinder benchmark using DeepXDE, leveraging the built-in cylinder\_nektar\_wake.mat dataset, which includes time-series data for velocity and pressure fields.

Key Aspects of the Implementation:

* + *Domain Setup:*

The computational domain consists of a rectangular region featuring a circular cylinder as an internal obstacle.

* + *Neural Network Architecture:*

A 6-layer fully connected network (FNN) was employed to approximate the physical quantities: u,v and p.

* + *Incorporation of Observational Data:*

Velocity and pressure data from known observation points were enforced using PointSet boundary conditions, enhancing the model's data

consistency.

* + *Parameter Estimation:*

The model was trained to learn the physical parameters λ1 (related to viscosity) and λ2 (pressure scaling) directly, treating them as trainable variables during optimization.

# Korteweg-de Vries Equation:

I used DeepXDE to solve the KdV equation using a PINN with trainable parameters

λ1 and λ2 , following the setup from the paper. Key Details:

* + Domain: x ∈ [−1,1], t ∈ [0,1], with periodic boundaries.
  + Initial Condition: u(x,0) =0.5 sech2(0.5x), representing a soliton.
  + Network: 3 hidden layers, 50 neurons per layer.
  + Objective: Learn both the solution u(x,t) and the governing parameters from physics-based constraints.