Solving Partial Differential Equations Using Deep Learning and Physical Constraints

https://www.mdpi.com/2076-3417/10/17/5917

Project by : Suyash Kapil (21411035) , Shivam Ojha (21321029)

Section 1: Summary

Partial differential equations (PDEs) find a lot of applications in science and engineering. Therefore, solving PDEs is a very important research area. There are already existing numerical methods to solve PDEs but we need more effective methods to solve PDEs. One of the fields which has caught a lot of attention from people globally is Machine Learning(ML) and Deep Learning(DL). DL itself has a lot of applications in Computer Vision and Natural language Processing. One of the most important tools in DL is Deep Neural Network. Neural Networks are inspired by the human brain and are a network of neurons which take combined efforts to make a decision (mostly to predict). Deep Neural Network, as the name suggests, is a neural network consisting of a lot of layers with each layer connected to other layers and containing a number of neurons. Neural Networks can fit continuous functions quite well. The paper uses Physics Informed Neural Networks(PINN) to study wave equations, KdV-Burgers equations and KdV equations. PDEs have been an important topic of interest in the course CHN 323 as it finds a lot of applications in Chemical Engineering like Heat transfer(like in boiling time of an egg) and Fluid Dynamics(like in Navier Stokes equations). MATLAB is used to train the neural networks to solve these three partial differential equations discussed in the paper. All plots are crafted using plot function in MATLAB.

Section 2 :Mathematical Formulation

Section 2.1 : Equations used

$$\mathcal{D}(\boldsymbol{u}(\boldsymbol{x}))=0 \; ; \; \boldsymbol{x}\in\Omega \quad (1)$$

$$\mathcal{B}(\boldsymbol{u}(\boldsymbol{x}))=0 \; ; \; \boldsymbol{x}\in\partial\Omega \quad (2)$$

Where u is unknown solution of the PDE and D is the differential operator while B stands for boundary condition of PDE.

$$f(\boldsymbol{x};\theta):=\mathcal{M}[\hat{u}\,(\boldsymbol{x};\theta)] \quad (3)$$

Where $\hat{u}(x;\theta)$ represents the neural network with input x and output u(x) , θ represents the neural network parameters like the weights.

The neural network contains L-1 hidden layers and an output layer with weight matrix

$W[k] \in \boldsymbol{R}^{n_k \times n_{k-1}}$ and bias vector $b[k] \in \boldsymbol{R}^{nk}$ for each layer *k* with n_k neurons.
f is a residual network.

$$J(\theta)=MSE_u+MSE_f \tag{4}$$

Where MSE_u and MSE_f are Mean squared errors of Neural networks f and û and J is the loss of physics informed neural network.

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u}\left|u^{\hat{}} - u\left(x_u^i, t_u^i\right)\right|^2 \tag{5}$$

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f}\left|f\left(x_f^i, t_f^i\right)\right|^2 \tag{6}$$

$u(x_{iu},t_{iu})$denotes training data from initial and boundary conditions and
$u(x_{if},t_{if})$ denotes the training data in the space-time domain

$$w* = \text{argmin}(J(w)) \quad w\in\theta \tag{7}$$
$$b* = \text{argmin}(J(b)) \quad b\in\theta \tag{8}$$

Due to unavailability of very high-end computational resources, The neural networks will have 10 neurons each for 6 layers rather than 100 neurons per layer. Also, there a few changes in the parameters for neural networks as listed in sections 2.1.1 , 2.1.2 , 2.1.3 . The software used for the studies is MATLAB.

Section 2.1.1 : Wave Equations

In mathematical form, this wave equation is defined, as follows:

$$u_{tt} - cu_{xx} = 0 \ , \ x\in[0,1], t\in[0,1] \tag{9}$$

Where c is wave propagation velocity and u(t,x) is a function of x and t. Also , it is assumed that c= 1 m/s

Initial conditions and Dirichlet boundary conditions:

$$u(0,x)=\sin(\pi x)/2$$
$$u_t(0,x)=\pi\sin(3\pi x)$$
$$u(t,0)=u(t,1)=0$$

The true solution of the above equation is

$$u(t,x) = \sin(\pi x)\cos(\pi t)/2 + \sin(3\pi x)\sin(3\pi t)/3 \tag{10}$$

Is the true solution of the wave equation

$$f(t,x) := u_{tt} - u_{xx} \tag{11}$$

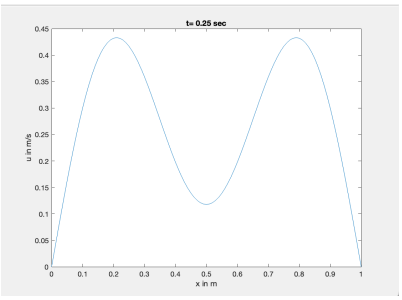


Fig 1

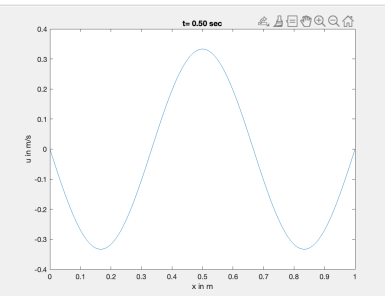


Fig 2

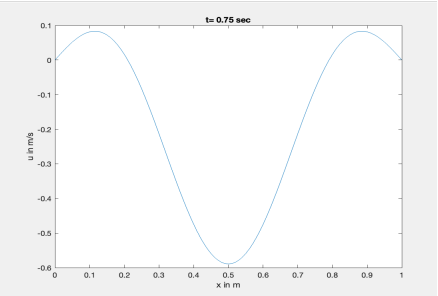


Fig 3

Fig 1, Fig 2, Fig 3 : u vs x at t=0.25 sec,0.5 sec and 0.75 sec.
Graphs were plotted using MATLAB. These plots are based on equation 10 giving the exact solution of the wave equation.

Constants and Parameters

Constants/Parameters	Values/meaning
C	1 m/s
t	Time in seconds (0.25,0.5,0.75 fixed)
x	Space in m
u	Wave velocity in m/s
Neural Network features	2 (x and t)
Optimiser	ADAM
Training Examples	32000
Validation Examples	4000
Test Examples	4000
Loss function	Mean Squared Error
Activation Function	tanh

Number of layers	6
Number of Neurons in each layer	10

Section 2.1.2 : KdV-Burgers Equation

The KdV–Burgers equation, also known as the Korteweg-de Vries–Burgers equation, is a nonlinear partial differential equation (PDE) that combines elements of both the Korteweg-de Vries (KdV) equation and the Burgers equation. It is used to describe the behavior of waves and fluid flows in various physical systems, including those involving the flow of liquids containing bubbles and the flow of liquids in elastic tubes.

In mathematical form, the KdV–Burgers equation is defined, as follows:

$$u_t + \upsilon u u_x + \nu u_{xx} + \mu u_{xxx} = 0$$

where α, β, and γ are all non-zero real constants

For the above Equation, the values of u,v,μ are given as $1, -0.075, \pi/1000$, respectively, and deterministic initial conditions are given. In mathematical form, the nonlinear KdV–Burgers equation with initial conditions studied is defined, as follows:-

Initial conditions is defined as:-

$$u(x,0) = -\frac{6\nu}{25\mu}\left[1 + \tanh\left(\frac{\nu x}{10\mu}\right) + \frac{1}{2}\operatorname{sech}^2\left(\frac{\nu x}{10\mu}\right)\right]$$

Exact solution of the KdV-Burgers equation is:-

$$u(x,t) = -\frac{6\nu}{25\mu}\left[1 + \tanh(\xi) + \frac{1}{2}\operatorname{sech}^2(\xi)\right]$$

where

$$\xi = \frac{\nu}{10\mu}\left(x + \frac{6\nu^2}{25\mu}t\right)$$

The above equation has been referenced from the following article.[1]

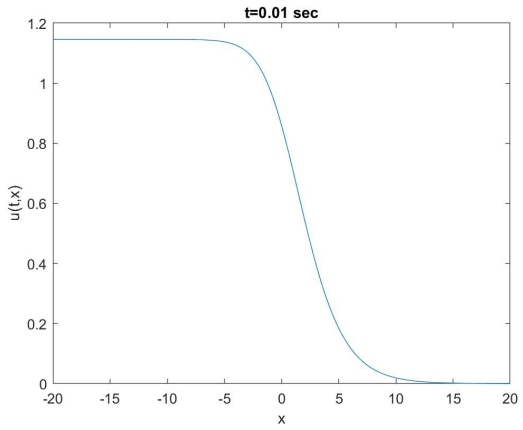


Fig 1

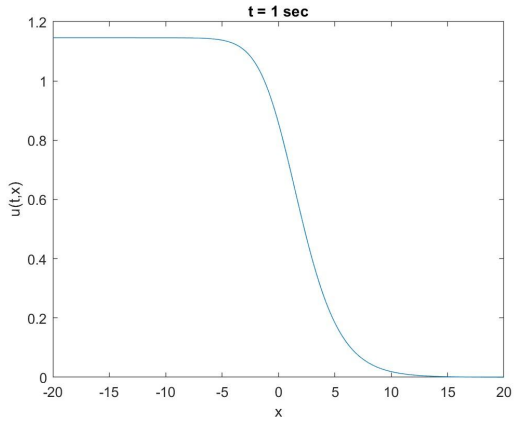


Fig 2

Fig 1, Fig 2, Fig 3 : u vs x at t = 0.01 sec,1 sec
Graphs were plotted using MATLAB. These plots were based on the above equations.

Constants and Parameters

Constants/Parameters	Values/meaning
C	1 m/s
t	Time in seconds (0.25,0.5,0.75 fixed)
x	Space in m
u	Wave velocity in m/s
Neural Network features	2 (x and t)
Optimiser	ADAM
Training Examples	32000
Validation Examples	4000
Test Examples	4000
Loss function	Mean Squared Error

Activation Function	tanh
Number of layers	6
Number of Neurons in each layer	10

Section 2.1.3 : KdV Equation

The KdV equation is important for understanding the nature of solitons and the interaction of two or more solitons.
KdV equation when the initial condition of $u(0,x)$=6 $\text{sech}^2 x$ is:-

$$u(x,t) = 12*(3+4\cosh(2x-8t)+\cosh(4x-64t))/(3\cosh(x-28t)+\cosh(3x-36t))^2$$

Also, the physics-informed neural network $f(t,x)$ is defined, as follows:

$$f(t,x) = u_t+6uu_x+u_{xxx}$$

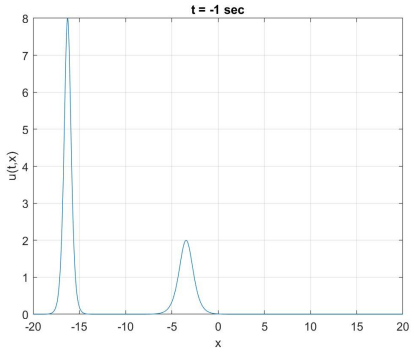


Fig 1

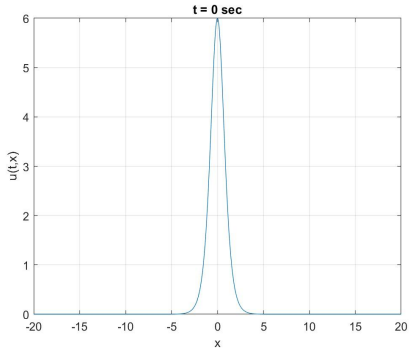


Fig 2

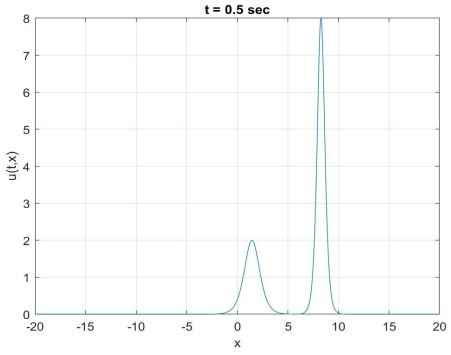


Fig 3

Fig 1, Fig 2, Fig 3 : u vs x at t = -1 sec,0 sec, 0.5 sec
Graphs were plotted using MATLAB. These plots were based on the above given equations.

Constants and Parameters:

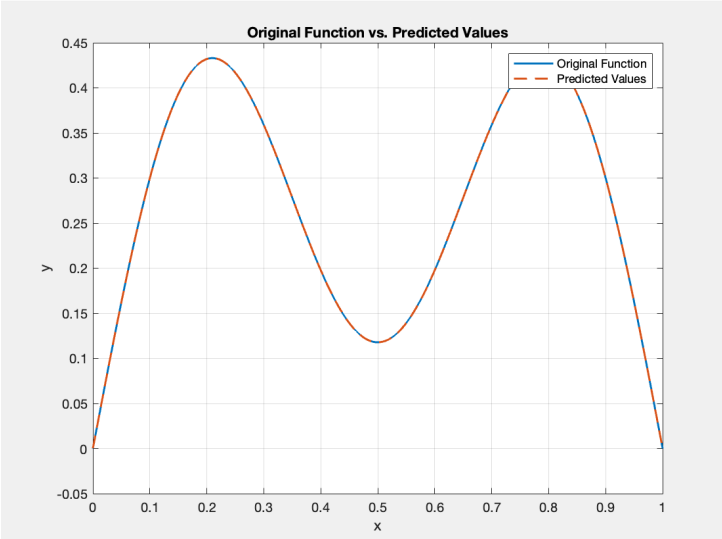
Constants/Parameters	Values/meaning
C	1 m/s
t	Time in seconds (0.25,0.5,0.75 fixed)
x	Space in m
u	Wave velocity in m/s
Neural Network features	2 (x and t)
Optimiser	ADAM
Training Examples	32000
Validation Examples	4000
Test Examples	4000
Loss function	Mean Squared Error
Activation Function	tanh
Number of layers	6
Number of Neurons in each layer	10

Section 3: Results

Section 3.1 : Wave equations

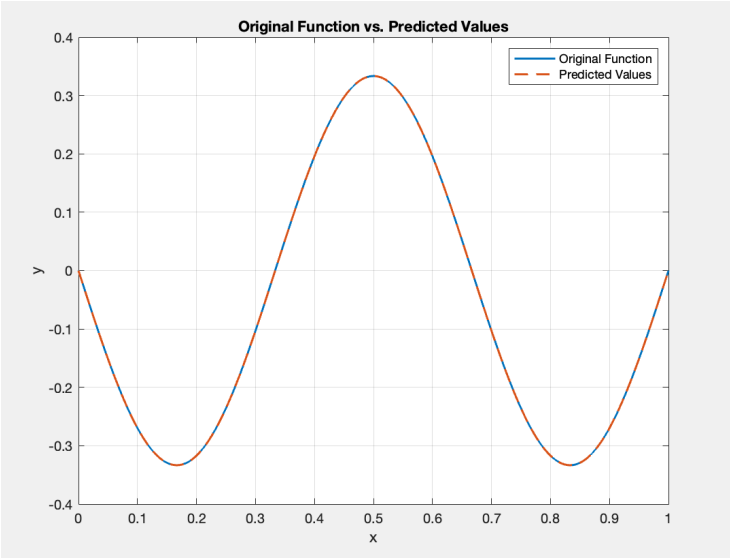
For t = 0.25 sec

u(y-axis) vs x(x-axis) graph :



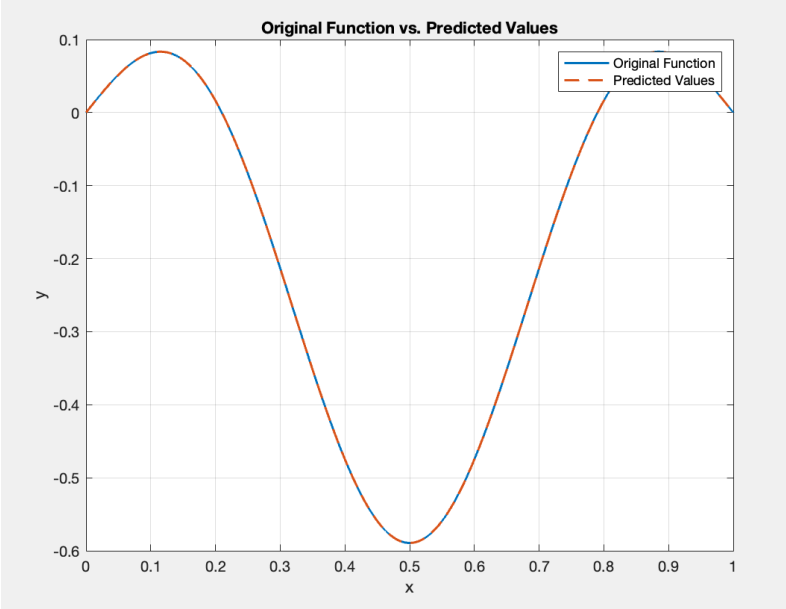
For t = 0.5 sec

u(y-axis) vs x(x-axis) graph :



For t = 0.75 sec

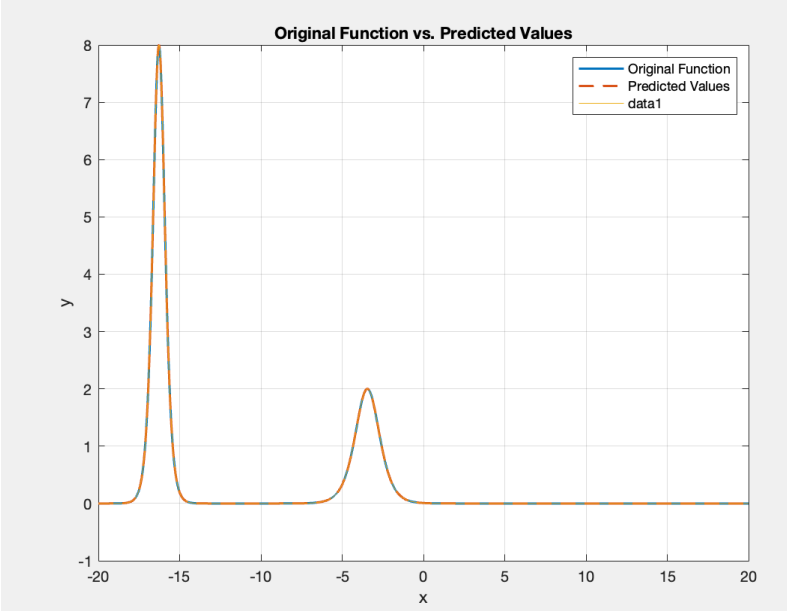
u(y-axis) vs x(x-axis) graph :



Section 3.2 : K-dV Equations

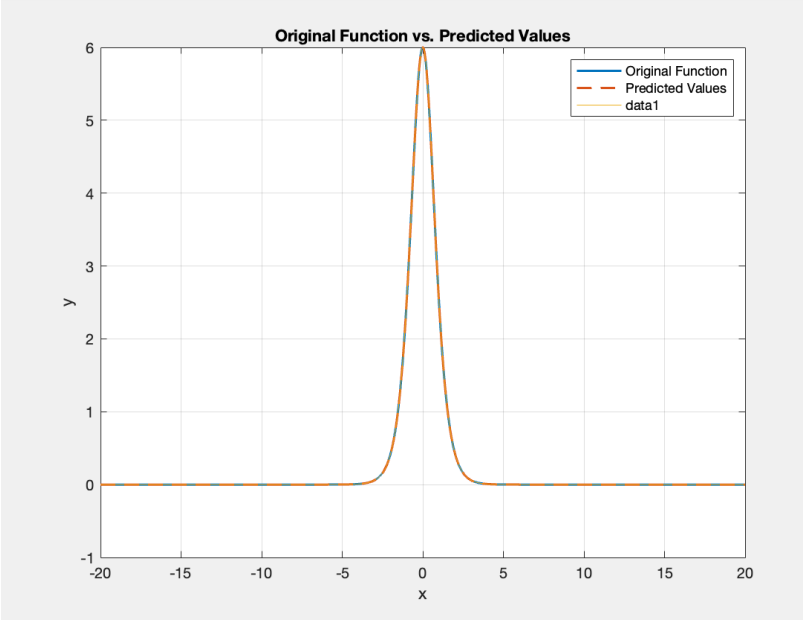
For t = -1 sec

u(y-axis) vs x(x-axis) graph :



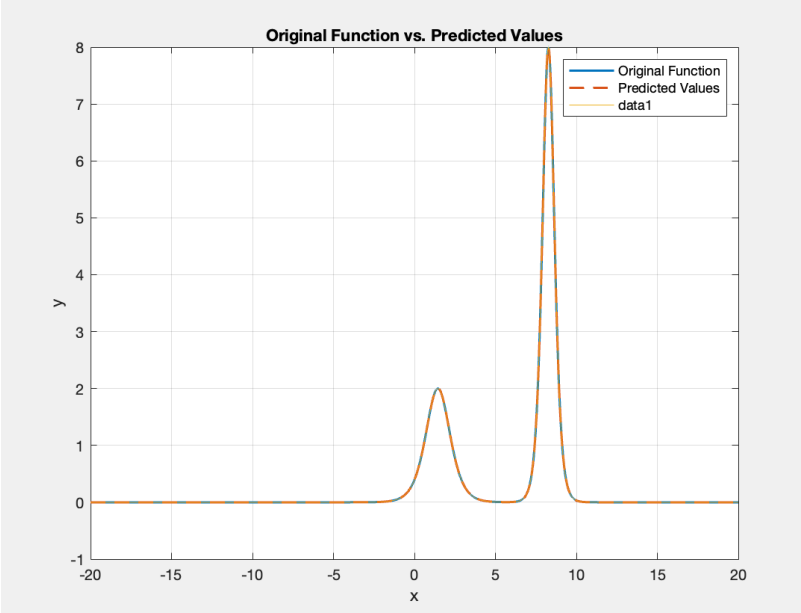
For t = 0 sec

u(y-axis) vs x(x-axis) graph :



For t=0.5 sec

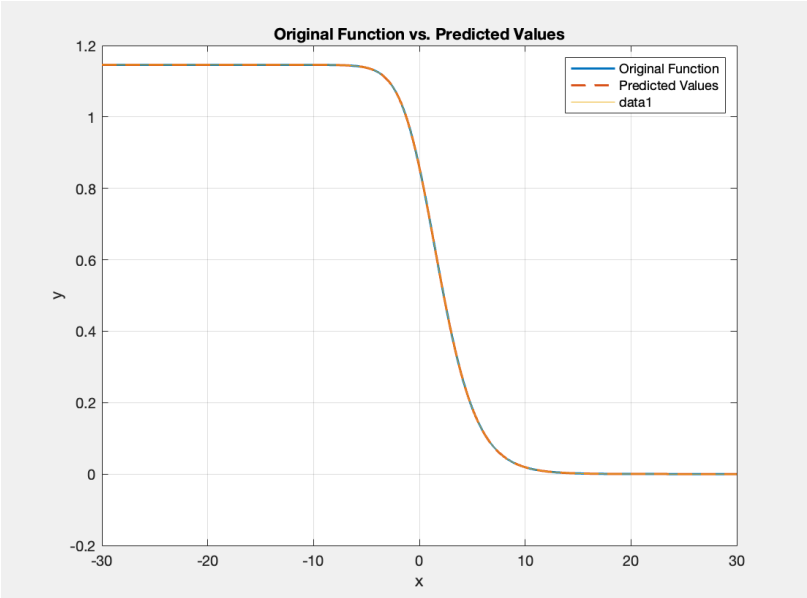
u(y-axis) vs x(x-axis) graph :



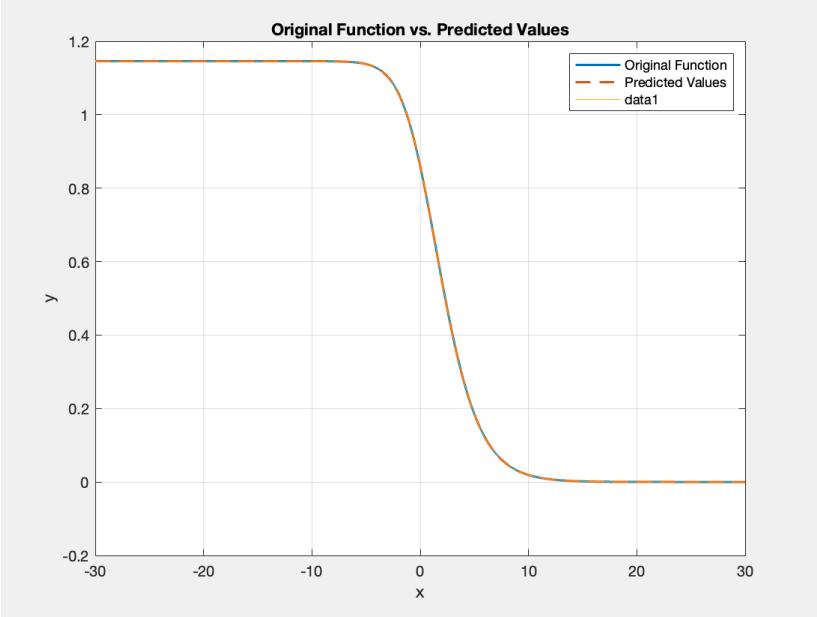
Section 3.3 : K-dV Burgers Equations

For t = 1 sec

u(y-axis) vs x(x-axis) graph :



For t = 0.01 sec
u(y-axis) vs x(x-axis) graph :



The data generated from the exact solutions for all equations given in the paper was split into train ,test and validation dataset. The Neural networks correctly predicted the values of u (target value) from input data (x and t). The study of different equations based on space and time gave us exact plots for fixed values of time. For fixed t , the value of target variable *u* can be now easily predicted. Similar approach can be used to find exact solutions for different partial differential equations. Though we predicted the values of u for fixed t and a finite domain of space, we can also predict the values of u for different values of t and a lot more values of x.

The neural network can be computationally heavy if we use a lot of neurons per layer and make the network deep but for simpler tasks like this one with 2 inputs and 1 output (this wasn't a classification problem but a regression problem), neural networks with just 6 layers and 10 neurons per layer worked quite well.

Section 4 :Individual Contributions

Suyash Kapil (21411035) :

- Drafted section 1, 2 , 2.1 , 2.1.1 and 3.
- Implemented Neural Networks for all equations

Shivam Ojha (21321029) :

- Drafted section 2.1.2 , 2.1.3
- Researched various research papers to find out the solution of KdV-Burgers equation