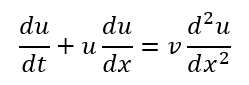
**Solving Burgers Equation with PINN**

# **Group Members**

Junchao Ma | Chiao Hsiao | Tiansheng Wu

# **Objective**

The primary goal for this project is to solve the Burgers Equation, a partial differential equation that is widely applied in the study of fluids, with a Physics Informed Neural Network (PINN), where:



# **Motivation**

The group solved the Burger's equation that combines nonlinear wave motion with linear diffusion to analyze the combined effect of nonlinear advection and diffusion. The major motive for the team is to investigate a method of pushing the PINN, which in Assignment 3 took one input, into a network that can take two inputs, thereby exploring the potential of the PINN in the territory of computational physics.

While varying the viscosity coefficient as a hyperparameter into the Burger’s equation to study the performance of the network, the team also tried to use it as the third input to the system, hoping to make the trained network universal to all different inputs of for studying its influence on the overall results.

# **Deliverable**

The results will be based on three separated programs and their outputs:

1. The first program will take two inputs [t;x] to generate a plot of u(x,t) in the t\_x span.
2. This program will also take two inputs [t;x] to generate the u plot. But in this program, the system will be trained with a low number of epochs. Instead, it will be trained several times on a variety of to study the influence of viscosity on the final result.
3. The last program takes three inputs [t;x;] to generate the plot. The network is redesigned to accept this new input matrix, and the presented result will be mainly plotted for =0.001 to be compared with the first program’s result.

# **Research Plan**

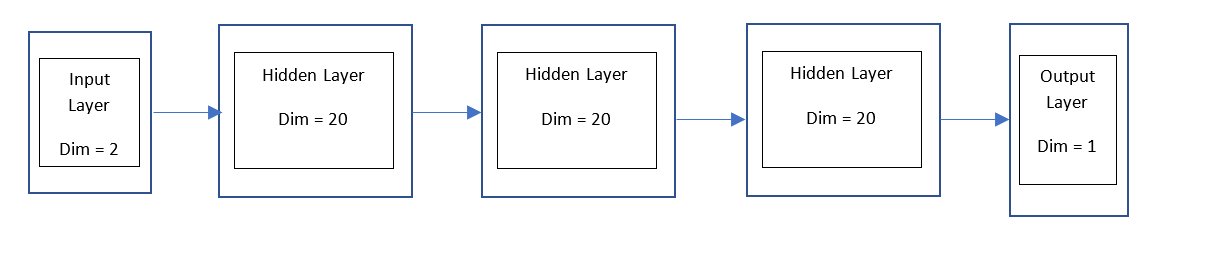
5.1 Brief Description of Algorithm

The group started with the PINN on fixed viscosity term . The network will be trained with 1000 randomly selected t and x pairs as inputs, together with a set of initial and boundary conditions. The network will be trained with several spanning from 0.001 to 0.09 to study ’s influence to u(x,t), each with a 2D plot on t\_x span and a 3D plot for the u surface.

Then, the once hyperparameter will be added as an additional input, making it of the form [t;x;]. The modified network will train with the inputs to generate the same plots as for part one, but this time with customized as a comparison.

5.2 Architecture

The original PINN has a depth of 4, containing one input layer with two inputs, followed by three hidden layers with width 20, and ends with an output layer with one output, shown in Fig 1.



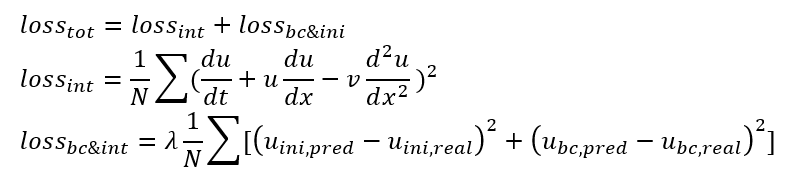
*Fig 1*: Original PINN with two inputs

Each hidden layer has a “*tanh*” activation function, with “*RandomNormal*” kernel and bias initializations, and a l2.regularization with parameter of 1e-6.

The modified PINN keeps most settings from the original network, with a smaller width of 16 for faster execution, and a different input dimension of 3.

5.3 Loss Function

The loss function for this network will consist of two parts: the interior loss and the boundary plus initial condition loss.



5.4 Training

The network will be trained with three configurations:

1. With the original PINN, viscosity fixed to 0.001, sample size 1000, epoch 6000
2. With the original PINN, viscosity varys from 0.001 to 0.09, each with sample size 1000, epoch 1000
3. With modified PINN, sample size 1000, epoch 5000

The conditions for the original PINN remains the same:

1. Interior condition
2. Boundary condition
3. Initial condition

The modified PINN will take one more input , which is the same for all conditions:

5.5 Validation

The hyperparameters are picked to balance the runtime and accuracy, where:

Width = 20 (original) /16 (modified);

Depth = 4;

Input dimension = 2 (original) / 3 (modified);

Output dimension = 1;

l2 regulation parameter = 1e-6;

= 10

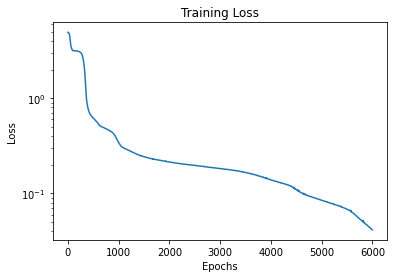
5.6 Testing

The output of the PINN will be compared with the numerical solution plots found on the internet to see if the network generates a close answer. To achieve a more vivid comparison, the final output of u(x,t) will be sliced on several cross sections on the t span, showing its difference along the temporal axis.

1. **Results and Discussion**

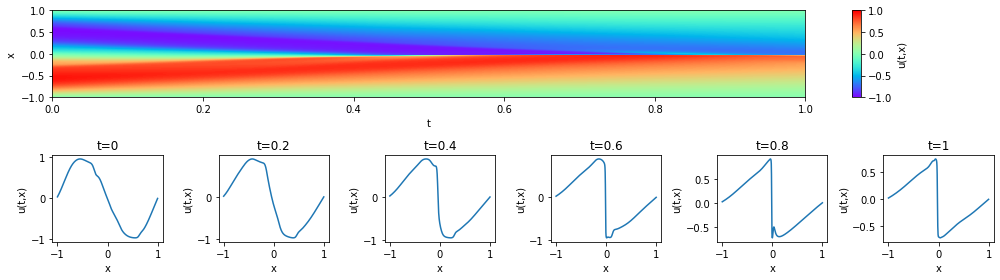
6.1 Original two-input model, = 0.001, N = 1000, Epoch = 6000

Epoch: 1 ; loss: 4.9398856 ; Epoch: 6000 ; loss: 0.04136759

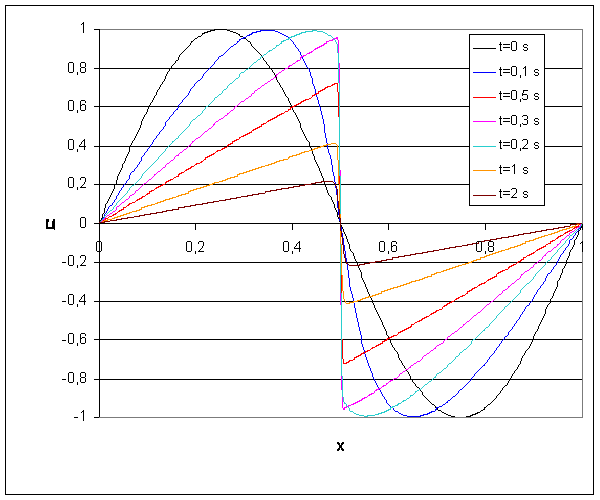


*Fig 3:* Training Loss for nu = 0.001

As Fig 3 indicates, the training loss keeps decreasing as the number of epoch increases. Even at 6000 epochs, it still shows a steady decrease in value. Increasing the training epoch will surely improve the network, but the time cost makes the team decide to stop at this point.



*Fig 4:* Result for u(x,t) on tx span, nu = 0.001

****

*Fig 5:* Numerical Solution Plots for u(x,t) on t cross-sections, nu = 0.001 [1]

Comparing the cross-sectional plots from the PINN and the numerical functions, the trends are very close. The function starts at the initial points with a negative sine wave, and the peak gradually shifts to the center at t increases. At = 0.001, the PINN indicates a magnitude around 1, which agrees with the numerical result.

In the PINN result, there are some noticeable noises in the surface, which may be caused by the comparably small sample size of 1000. The problem can also be solved by running more epochs to train the model, but both methods will significantly increase the runtime as a trade-off.

6.2 Original two-input, = 0.001, 0.01, 0.05, 0.09 for N = 1000, Epoch = 1000

In addition to the model with a fixed hyperparameter “Nu” and a large epoch number. The group also tested the performance of the algorithm using different values of “nu” including 0.001, 0.01, 0.05 and 0.09. In this report, we selected the training results for the case that “nu” = 0.01 and 0.05 to highlight the performance of the network due to page constraints of the report. As the figure 6 has shown, as epoch number increases, the training loss decreases drastically for both cases. What’s more, as the figure 7 and 8 has shown, as viscosity v increases, the shock’s magnitude at the same t fades, which fits the trend shown in figure 9.

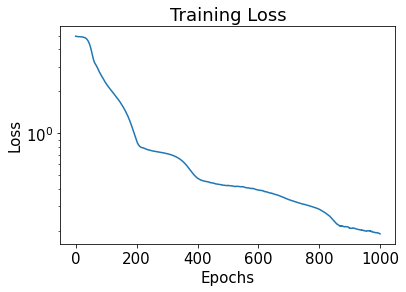
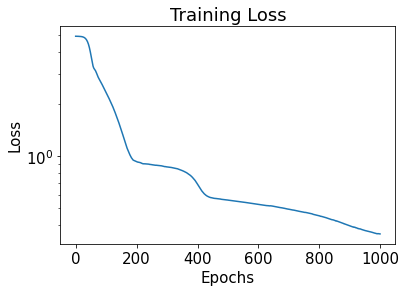
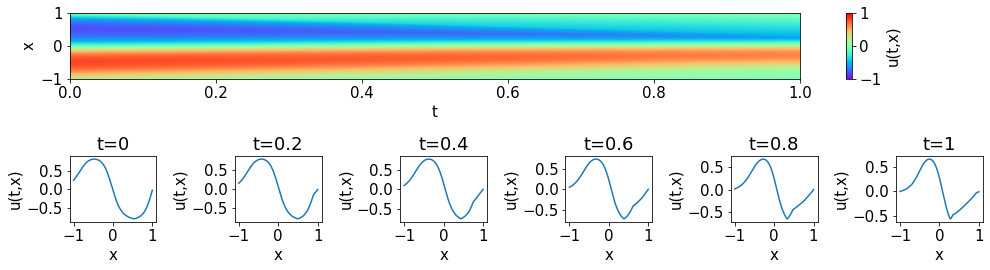


Fig 6: Training loss for nu = 0.01, 0.05 with epoch = 1000



*Fig 7*: Result for u(x,t) on tx span for nu = 0.01 when epoch = 1000

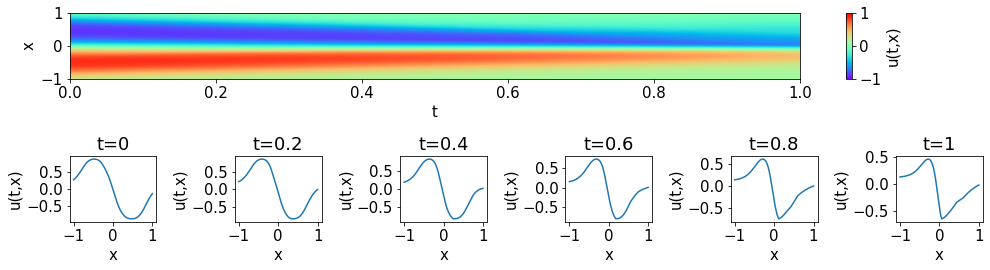


Fig 8: Result for u(x,t) on tx span for nu = 0.05 when epoch = 1000

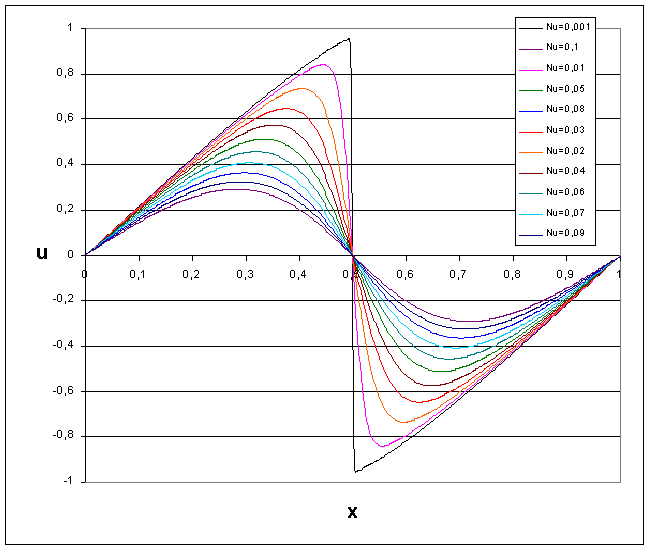


Fig 9: Numerical solution for u(x,t) at t = 0.3 for different [1]

6.3 Modified three-input model, N=10000 , epoch=10000

As Fig 10 indicates, the training loss starts from 50.32 in the first epoch, and gradually drops to 0.0425 at the 10000th, with a steady decreasing trend at this point. But the u(x,t), shown in Fig 11 with some features about the actual solution, still has a large difference compared to the result from the 2-input model. This situation could be caused by the lack of epochs, which prevents the model from further optimizing itself, due to the large number of 10000 samples in the setting. Also, to further increase the number of samples will help distinguish the details on the model, but again both of these methods will be expensive in runtime, which limits the team on further optimizing the functions.

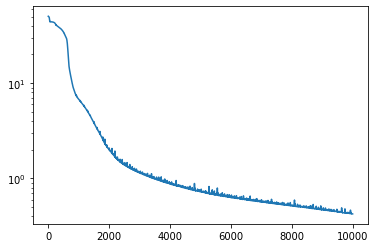
****

Fig 10: Training loss for epoch = 10000

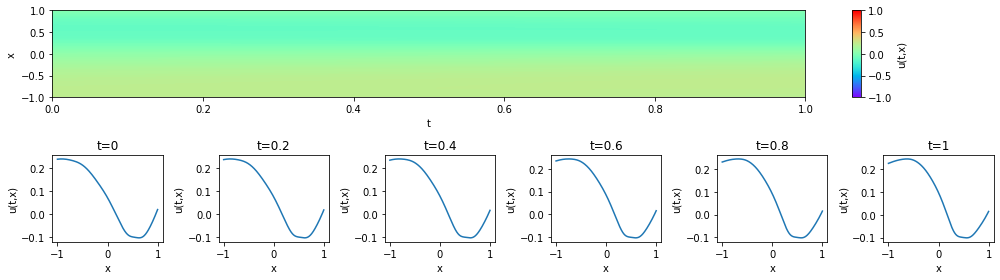
****

Fig 11: Result for u(x,t) on tx span for nu = 0.001

1. **Conclusions and Future Work**

In conclusion, our network has an acceptable performance when having two inputs, t and x, but becomes worse when the input size increases. This can be the result of the small input size considered in our program structure. This is not obvious with fixed but will make the 3-input model less conclusive. The training cost, on the other hand, makes further training too expensive to perform under current limits. Since the algorithm doesn’t have enough time to optimize to an optimal range, the model failed to be universal to all values.

As for future improvements, the first thing the group can try is to decrease the runtime for each step of gradient descent. Either by adjusting the algorithm, or by optimizing the training functions. Then, the team can increase both the sample size to get a more conclusive model, and the number of epochs to achieve a lower loss, making the predicted model closer to the actual numerical solutions.

**References**

***INTRODUCTION*. (n.d.). Retrieved December 3, 2021, from** [**http://hmf.enseeiht.fr/travaux/CD0001/travaux/optmfn/hi/01pa/hyb41/reportbid.htm**](http://hmf.enseeiht.fr/travaux/CD0001/travaux/optmfn/hi/01pa/hyb41/reportbid.htm)

**okada39. (2021). *Pinn\_burgers* [Python].** [**https://github.com/okada39/pinn\_burgers**](https://github.com/okada39/pinn_burgers) **(Original work published 2020)**

**Appendix**

1. 2-input model with nu = 0.001, N = 1000, epoch = 6000

=====================================

Model for nu=0.001

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense (Dense) (None, 20) 60

dense\_1 (Dense) (None, 20) 420

dense\_2 (Dense) (None, 20) 420

dense\_3 (Dense) (None, 1) 21

=================================================================

Total params: 921

Trainable params: 921

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Epoch: 1 ; loss: 4.9398856

Epoch: 250 ; loss: 3.0319195

Epoch: 500 ; loss: 0.6214795

Epoch: 750 ; loss: 0.47900057

Epoch: 1000 ; loss: 0.34329128

Epoch: 1250 ; loss: 0.2761363

Epoch: 1500 ; loss: 0.24320756

Epoch: 1750 ; loss: 0.2257475

Epoch: 2000 ; loss: 0.21380009

Epoch: 2250 ; loss: 0.20417045

Epoch: 2500 ; loss: 0.19703096

Epoch: 2750 ; loss: 0.18937819

Epoch: 3000 ; loss: 0.18260515

Epoch: 3250 ; loss: 0.17545755

Epoch: 3500 ; loss: 0.16623622

Epoch: 3750 ; loss: 0.15340784

Epoch: 4000 ; loss: 0.13923864

Epoch: 4250 ; loss: 0.12684716

Epoch: 4500 ; loss: 0.10997059

Epoch: 4750 ; loss: 0.09394233

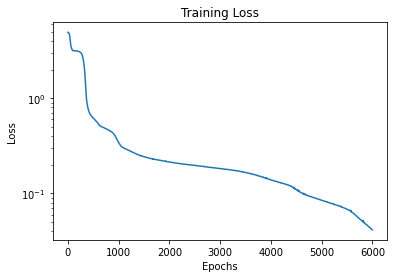
Epoch: 5000 ; loss: 0.08486827

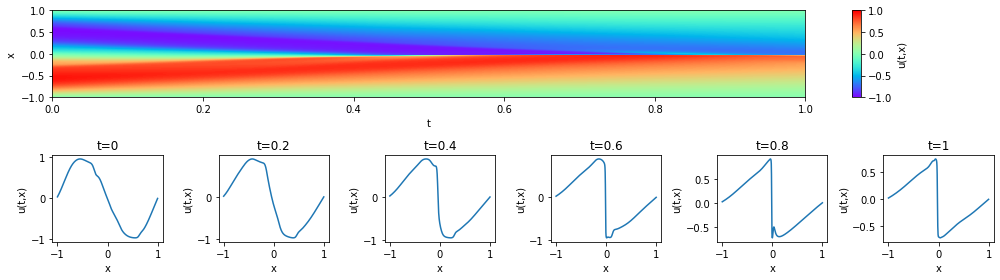
Epoch: 5250 ; loss: 0.07676418

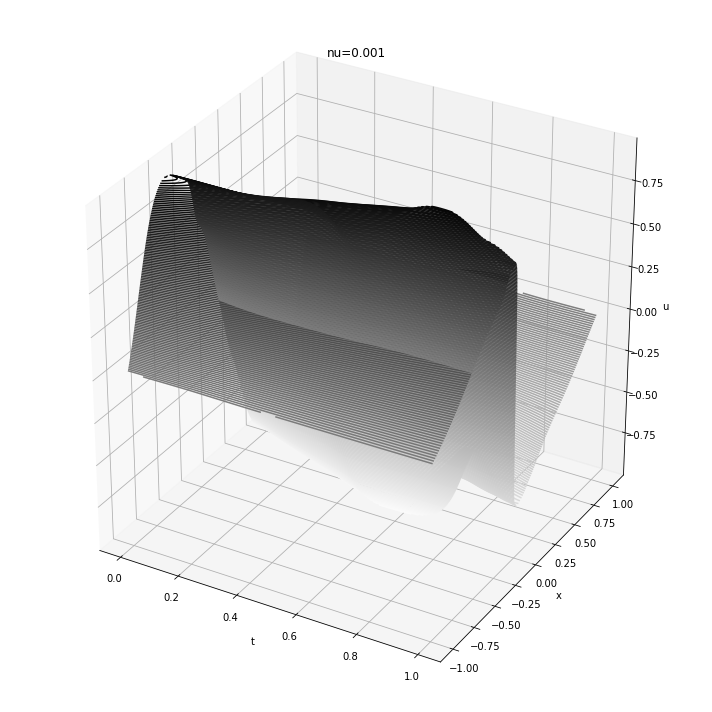
Epoch: 5500 ; loss: 0.0683226

Epoch: 5750 ; loss: 0.054058116

Epoch: 6000 ; loss: 0.04136759







1. 2- input model with nu = [0.001, 0.01, 0.05, 0.09], N = 1000, epoch = 1000

=====================================

Model for nu=0.001

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense (Dense) (None, 20) 60

dense\_1 (Dense) (None, 20) 420

dense\_2 (Dense) (None, 20) 420

dense\_3 (Dense) (None, 20) 420

dense\_4 (Dense) (None, 1) 21

=================================================================

Total params: 1,341

Trainable params: 1,341

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Epoch: 1 ; loss: 4.949896

Epoch: 50 ; loss: 3.9957569

Epoch: 100 ; loss: 2.4317877

Epoch: 150 ; loss: 1.655387

Epoch: 200 ; loss: 1.0423687

Epoch: 250 ; loss: 0.97648424

Epoch: 300 ; loss: 0.91672313

Epoch: 350 ; loss: 0.89269173

Epoch: 400 ; loss: 0.8365715

Epoch: 450 ; loss: 0.67128557

Epoch: 500 ; loss: 0.6126861

Epoch: 550 ; loss: 0.59710264

Epoch: 600 ; loss: 0.5804051

Epoch: 650 ; loss: 0.56394136

Epoch: 700 ; loss: 0.5503984

Epoch: 750 ; loss: 0.53299403

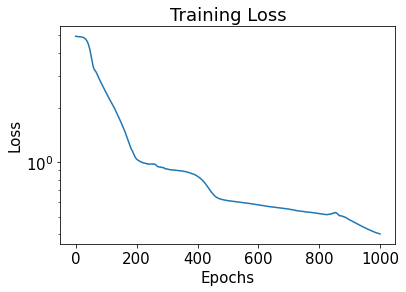
Epoch: 800 ; loss: 0.51960725

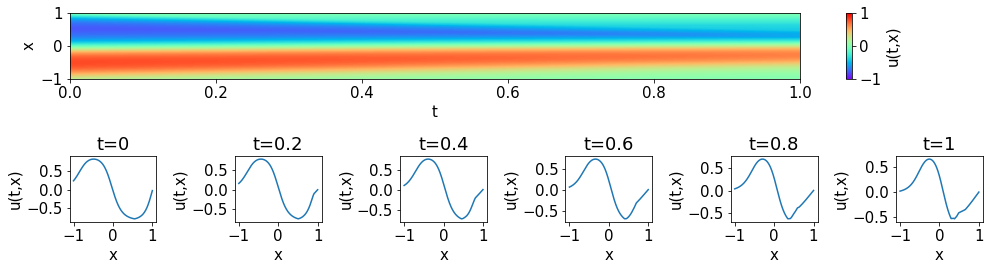
Epoch: 850 ; loss: 0.52505094

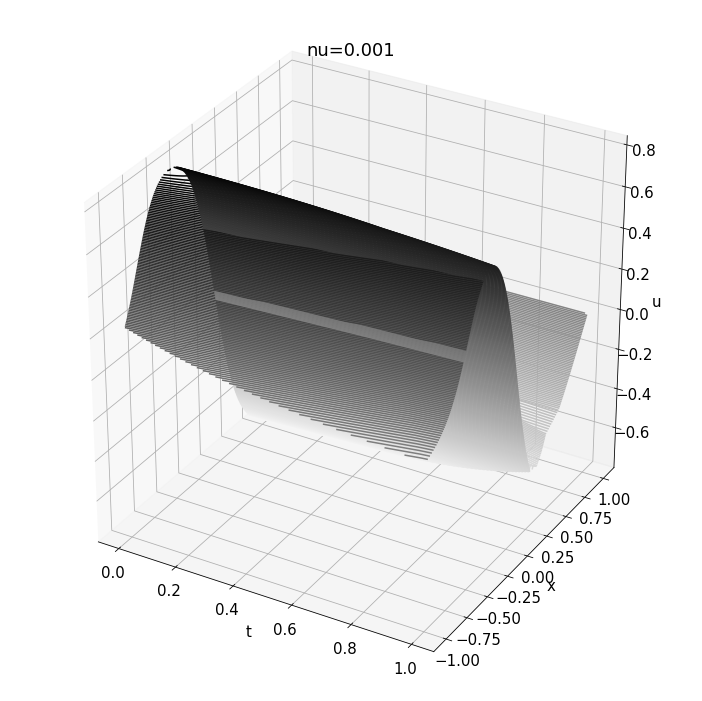
Epoch: 900 ; loss: 0.48020133

Epoch: 950 ; loss: 0.43403992

Epoch: 1000 ; loss: 0.40123558







=====================================

Model for nu=0.01

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense (Dense) (None, 20) 60

dense\_1 (Dense) (None, 20) 420

dense\_2 (Dense) (None, 20) 420

dense\_3 (Dense) (None, 20) 420

dense\_4 (Dense) (None, 1) 21

=================================================================

Total params: 1,341

Trainable params: 1,341

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Epoch: 1 ; loss: 4.930735

Epoch: 50 ; loss: 3.9273958

Epoch: 100 ; loss: 2.3497388

Epoch: 150 ; loss: 1.4277513

Epoch: 200 ; loss: 0.9302664

Epoch: 250 ; loss: 0.8934682

Epoch: 300 ; loss: 0.8668783

Epoch: 350 ; loss: 0.8260935

Epoch: 400 ; loss: 0.69284475

Epoch: 450 ; loss: 0.5733429

Epoch: 500 ; loss: 0.5575891

Epoch: 550 ; loss: 0.5427774

Epoch: 600 ; loss: 0.52689964

Epoch: 650 ; loss: 0.51325375

Epoch: 700 ; loss: 0.49386522

Epoch: 750 ; loss: 0.47370613

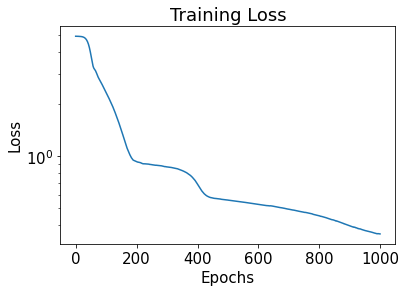
Epoch: 800 ; loss: 0.4524432

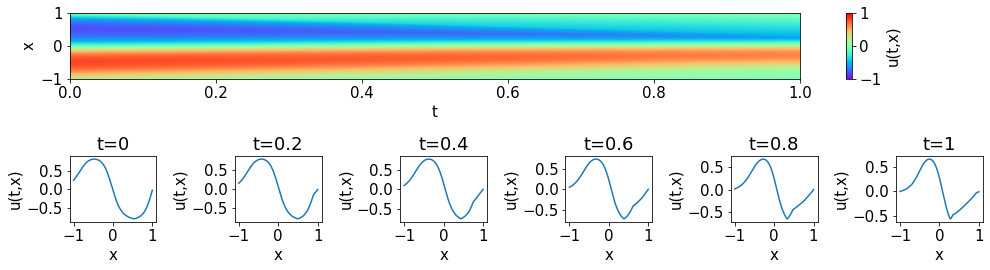
Epoch: 850 ; loss: 0.4256126

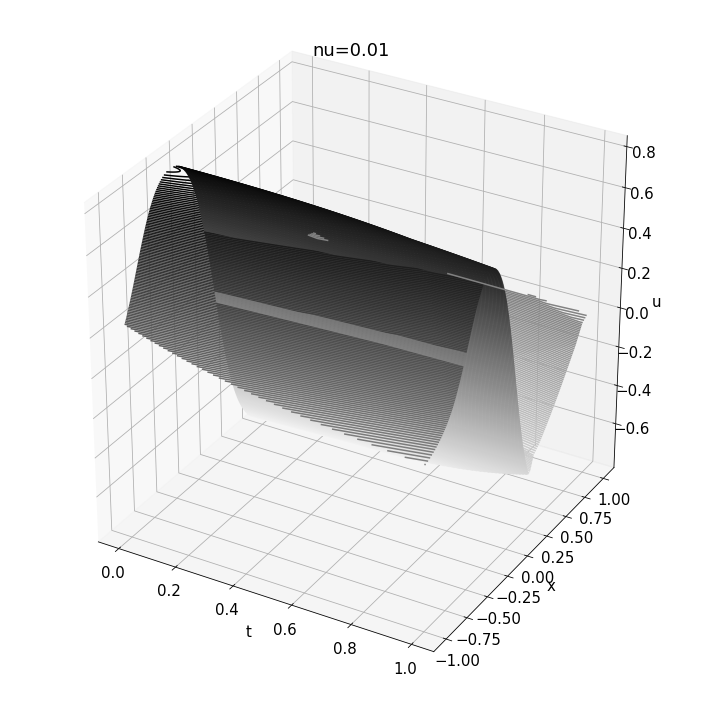
Epoch: 900 ; loss: 0.3964507

Epoch: 950 ; loss: 0.37183392

Epoch: 1000 ; loss: 0.35523582







=====================================

Model for nu=0.05

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense (Dense) (None, 20) 60

dense\_1 (Dense) (None, 20) 420

dense\_2 (Dense) (None, 20) 420

dense\_3 (Dense) (None, 20) 420

dense\_4 (Dense) (None, 1) 21

=================================================================

Total params: 1,341

Trainable params: 1,341

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Epoch: 1 ; loss: 4.9607067

Epoch: 50 ; loss: 4.117184

Epoch: 100 ; loss: 2.289267

Epoch: 150 ; loss: 1.6363257

Epoch: 200 ; loss: 0.89667374

Epoch: 250 ; loss: 0.7500971

Epoch: 300 ; loss: 0.71552175

Epoch: 350 ; loss: 0.6364465

Epoch: 400 ; loss: 0.47903106

Epoch: 450 ; loss: 0.44054863

Epoch: 500 ; loss: 0.4239094

Epoch: 550 ; loss: 0.41495508

Epoch: 600 ; loss: 0.39287883

Epoch: 650 ; loss: 0.3685439

Epoch: 700 ; loss: 0.3355063

Epoch: 750 ; loss: 0.3102761

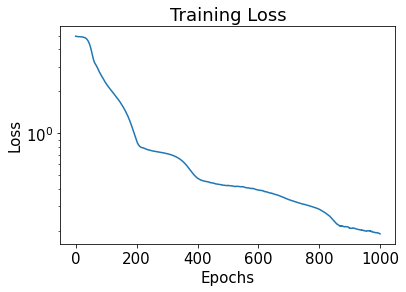
Epoch: 800 ; loss: 0.2862444

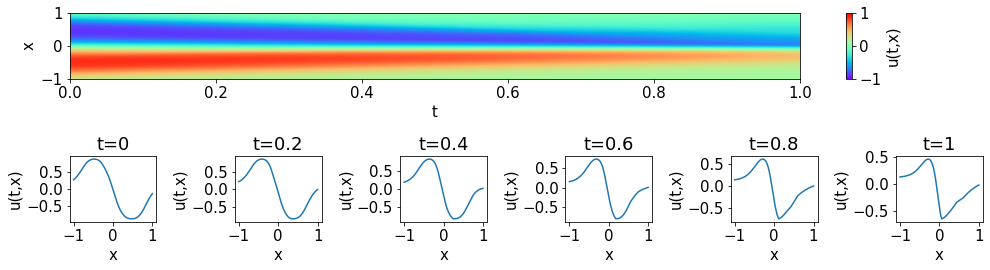
Epoch: 850 ; loss: 0.23416919

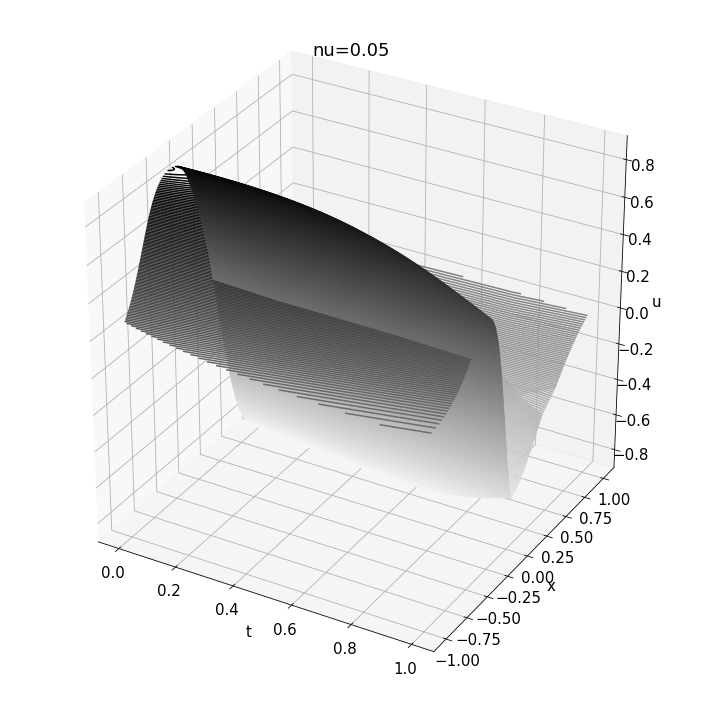
Epoch: 900 ; loss: 0.21088244

Epoch: 950 ; loss: 0.20028551

Epoch: 1000 ; loss: 0.1903201







=====================================

Model for nu=0.09

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense (Dense) (None, 20) 60

dense\_1 (Dense) (None, 20) 420

dense\_2 (Dense) (None, 20) 420

dense\_3 (Dense) (None, 20) 420

dense\_4 (Dense) (None, 1) 21

=================================================================

Total params: 1,341

Trainable params: 1,341

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Epoch: 1 ; loss: 4.952995

Epoch: 50 ; loss: 4.320343

Epoch: 100 ; loss: 2.665732

Epoch: 150 ; loss: 1.411202

Epoch: 200 ; loss: 0.70734787

Epoch: 250 ; loss: 0.6883949

Epoch: 300 ; loss: 0.67067695

Epoch: 350 ; loss: 0.64746064

Epoch: 400 ; loss: 0.627109

Epoch: 450 ; loss: 0.57221216

Epoch: 500 ; loss: 0.4904956

Epoch: 550 ; loss: 0.4693532

Epoch: 600 ; loss: 0.4598919

Epoch: 650 ; loss: 0.45521724

Epoch: 700 ; loss: 0.44749963

Epoch: 750 ; loss: 0.43636066

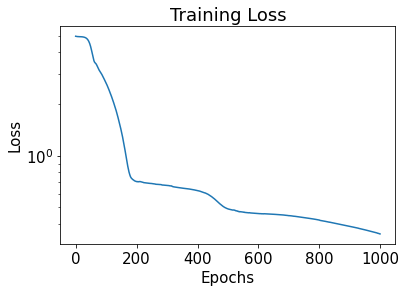
Epoch: 800 ; loss: 0.42200258

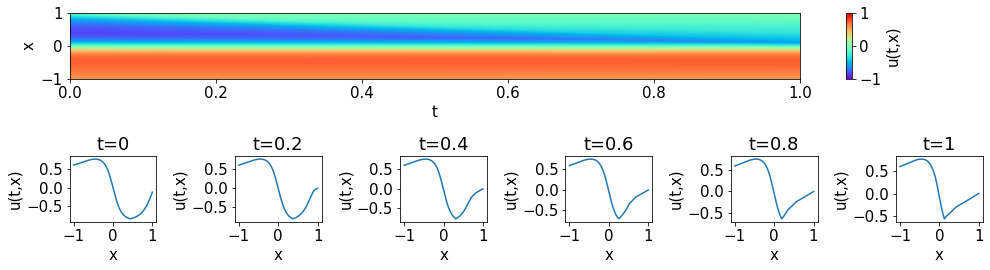
Epoch: 850 ; loss: 0.4051593

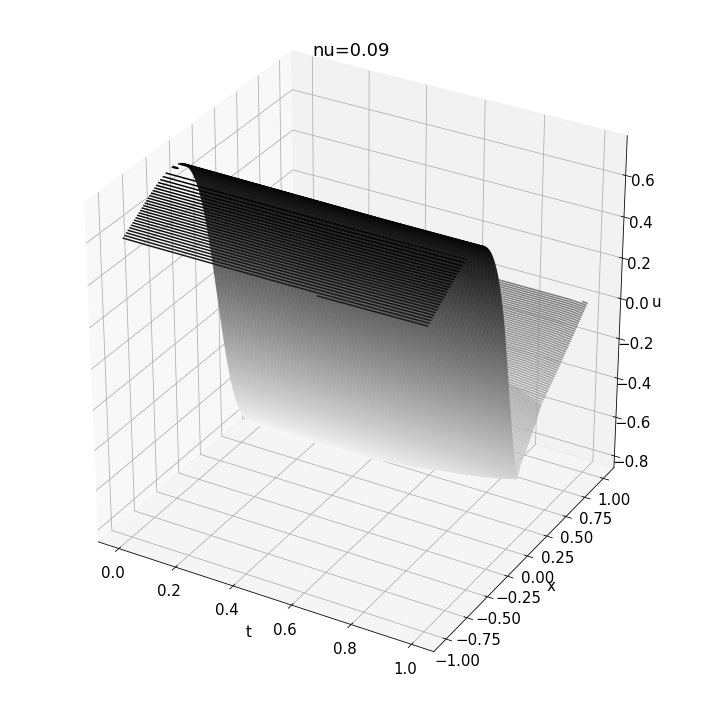
Epoch: 900 ; loss: 0.3883215

Epoch: 950 ; loss: 0.37032557

Epoch: 1000 ; loss: 0.3503912







1. 3-input model with N = 10000, epoch = 10000, plot with nu = 0.001

Epoch: 1 ; loss: 4.845379

Epoch: 250 ; loss: 3.790663

Epoch: 500 ; loss: 2.3096476

Epoch: 750 ; loss: 0.7604588

Epoch: 1000 ; loss: 0.67230153

Epoch: 1250 ; loss: 0.5691591

Epoch: 1500 ; loss: 0.4832564

Epoch: 1750 ; loss: 0.40860897

Epoch: 2000 ; loss: 0.3441502

Epoch: 2250 ; loss: 0.2973544

Epoch: 2500 ; loss: 0.26343682

Epoch: 2750 ; loss: 0.24554291

Epoch: 3000 ; loss: 0.2269408

Epoch: 3250 ; loss: 0.21647513

Epoch: 3500 ; loss: 0.20851037

Epoch: 3750 ; loss: 0.20223492

Epoch: 4000 ; loss: 0.1997245

Epoch: 4250 ; loss: 0.19637673

Epoch: 4500 ; loss: 0.18878676

Epoch: 4750 ; loss: 0.19173425

Epoch: 5000 ; loss: 0.1824739

Epoch: 5250 ; loss: 0.17986342

Epoch: 5500 ; loss: 0.17673168

Epoch: 5750 ; loss: 0.17398551

Epoch: 6000 ; loss: 0.17160301

Epoch: 6250 ; loss: 0.17050919

Epoch: 6500 ; loss: 0.16886981

Epoch: 6750 ; loss: 0.16626336

Epoch: 7000 ; loss: 0.1649211

Epoch: 7250 ; loss: 0.1643345

Epoch: 7500 ; loss: 0.16229749

Epoch: 7750 ; loss: 0.16350773

Epoch: 8000 ; loss: 0.16043231

Epoch: 8250 ; loss: 0.1585056

Epoch: 8500 ; loss: 0.15749173

Epoch: 8750 ; loss: 0.16026723

Epoch: 9000 ; loss: 0.15617026

Epoch: 9250 ; loss: 0.1549464

Epoch: 9500 ; loss: 0.15406938

Epoch: 9750 ; loss: 0.15320773

Epoch: 10000 ; loss: 0.15685886

