Final Report

One-Dimension Wave Propagation

Wei-Hsuan Shih, Yiran You, Zhilong Li

Motivation

One of our team members had experience using the numerical solver to do wave simulations and was unsatisfied with the performance. The numerical solver would require either a powerful computer or a lot of time to come up with the solution. We want to use this project as an opportunity to develop an alternative and hopefully more efficient method for her future wave simulation work.

Objective

Deep learning algorithms, Physics-Informed Neural Networks (PINNs) are applied to solve the one-dimension wave propagation. The wave equation is shown below.

$$u_{tt} = c^2 u_{xx}$$

By using machine learning methods, we hope to acquire a faster and more efficient solver with comparable precision and accuracy with respect to the traditional numerical solver.

Deliverable

A jupyter notebook file that contains all code for the network is delivered. We provide a sample trained model that can perform a prediction of wave propagation with a given initial condition. We compare the PINNs results and the results from the traditional method (numerical solver), documenting the accuracy and efficiency between the two methods. Graphs with results from the neural network and numerical solver overplotted are provided together with mean square error plots.

Research Plan

Description of the algorithm

The input is $< x_i, t_j, c_k >$ where x_i is the position of i-th collocation point. t_j means at time step j. c is the wave speed. The goal is to obtain predictions for 1D wave propagation with a given wave speed c, the same boundary and initial conditions. Notice that we train the model with a family of c which is ranging from 0.1 (m/s) to 0.9 (m/s). The output is a scalar value u, representing

the wave displacement on position x_i at t_j with a wave speed c. Our algorithm will be mapping the position and time to the wave displacement.

Architecture

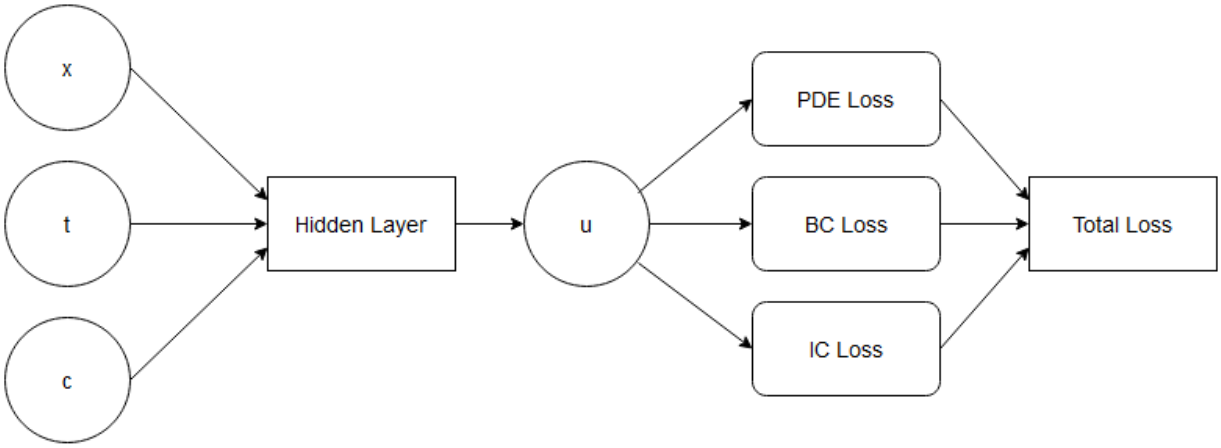

Figure 1. Architecture of the PINNs network

The network structure is a fully connected MLP with the following configuration:

- Input dimension d = 3
- Output dimension D = 1
- Depth = 10
- Width = 20
- Activation function = tanh
- No activation function at the output layer

Loss Function

The loss function is shown below

$$Loss_{total} = Loss_{ic} + Loss_{bc} + Loss_{r}$$

Where $Loss_{ic}$, $Loss_{bc}$, $Loss_{r}$ are the initial loss, boundary loss, and interior loss.

Initial Loss

Two initial conditions are given below:

1. $u(x, t = 0, c) = f(x)$, and $f(x)$ is a gaussian distribution shown in Figure 2.
2. $u_t(x, t = 0, c) = 0$

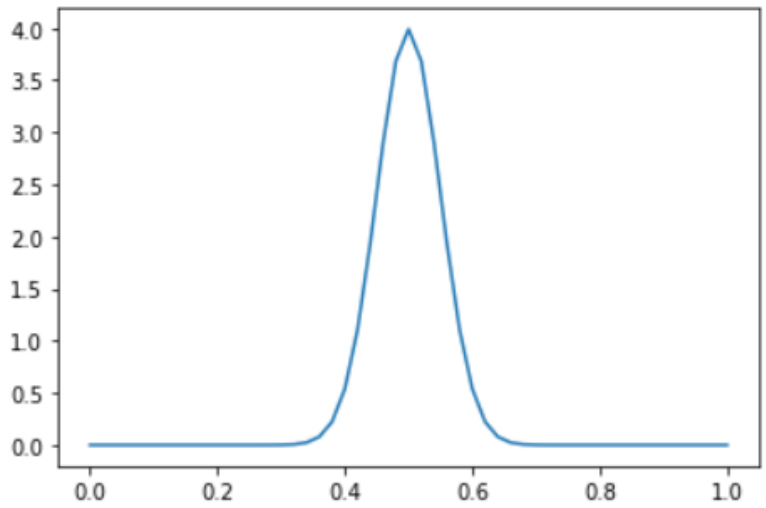

Figure 2. Initial condition gaussian distribution

The initial loss functions are defined as,

$$Loss_{ic} = Loss_{ic\_1} + Loss_{ic\_2}$$

$$Loss_{ic\_1} = \frac{\lambda_{ic\_1}}{N_{ic}} \times [u(x,\ t = 0,\ c,\ \theta) - u(x,\ t = 0,\ c)]^2 = \frac{\lambda_{ic\_1}}{N_{ic}} \times [u(x,\ t = 0,\ c,\ \theta) - f(x)]^2$$

$$Loss_{ic\_2} = \frac{\lambda_{ic\_2}}{N_{ic}} \times [u_t(x,\ t = 0,\ c,\ \theta) - u_t(x,\ t = 0,\ c)]^2 = \frac{\lambda_{ic\_1}}{N_{ic}} \times [u_t(x,\ t = 0,\ c,\ \theta) - 0]^2$$

Where $Loss_{ic\_1}$, $Loss_{ic\_2}$, λ_{ic_1}, λ_{ic_2}, N_{ic} are the initial loss, hyperparameter and number of dataset for different initial conditions.

Boundary Loss

Two boundary conditions are given:

1. $u_x(x = 0,\ t,\ c) = 0$
2. $u_x(x = L,\ t,\ c) = 0$

The boundary loss function is defined as,

$$Loss_{bc} = \frac{\lambda_b}{N_{bc}} \times \{[u_x(x = 0,\ t,\ c,\ \theta) - u_x(x = 0,\ t,\ c)]^2 + [u_x(x = L,\ t,\ c,\ \theta) - u_x(x = L,\ t,\ c)]^2\}$$

Where $Loss_{bc}$, λ_b, N_{bc} are the boundary loss, hyperparameter and number of dataset for boundary conditions.

Interior Loss

Interior loss is given below,

$$Loss_r = \frac{\lambda_r}{N_r}(u_{tt} - c^2 u_{xx})^2$$

Where $Loss_r$, λ_r, N_r are the interior loss, hyperparameter, and number of dataset for governing equations.

Training

We are using PINNs to train and solve the PDE. With given initial conditions, we sum up all the losses of all collocation points for every time step j. Then backpropagate this loss value to the whole network for updating the weights and biases.

Training data will be generated like below:

$$[< x_i, t_j, c_k > \ for\ c\ in\ [c_1, c_2, ..., c_k]\ for\ t_j\ in\ [t_1, t_2, ..., t_j]\ \ for\ x_i\ in\ [x_1, x_2, ..., x_i]]$$

Validation and Testing

To verify whether the model is correct and well-trained, we quantitatively evaluate it with the root mean square error (RMSE), the difference between the predicted results from our model and numerical results from the numerical solver with the same configuration (e.g. initial conditions, boundary conditions, and dx, dt).

The model has four hyperparameters λ_{ic_1}, λ_{ic_2}, λ_b, λ_r, representing the different weights of those losses. While doing network training, we monitor the result of our network and see if the weights need to be adjusted. For example, we predicted the result at $t = 0$, and see if it's fitting correctly with our given initial condition. If not, we can set a higher weight of initial loss to balance.

Results and Discussion

In this study, the purpose is to make a prediction for one-dimension wave propagation by using Physics informed neural networks (PINNs) and achieve the results comparable to the numerical solver with acceptable accuracy. Shown below in Figure 3 is the result generated from our trained neural network at the given condition.

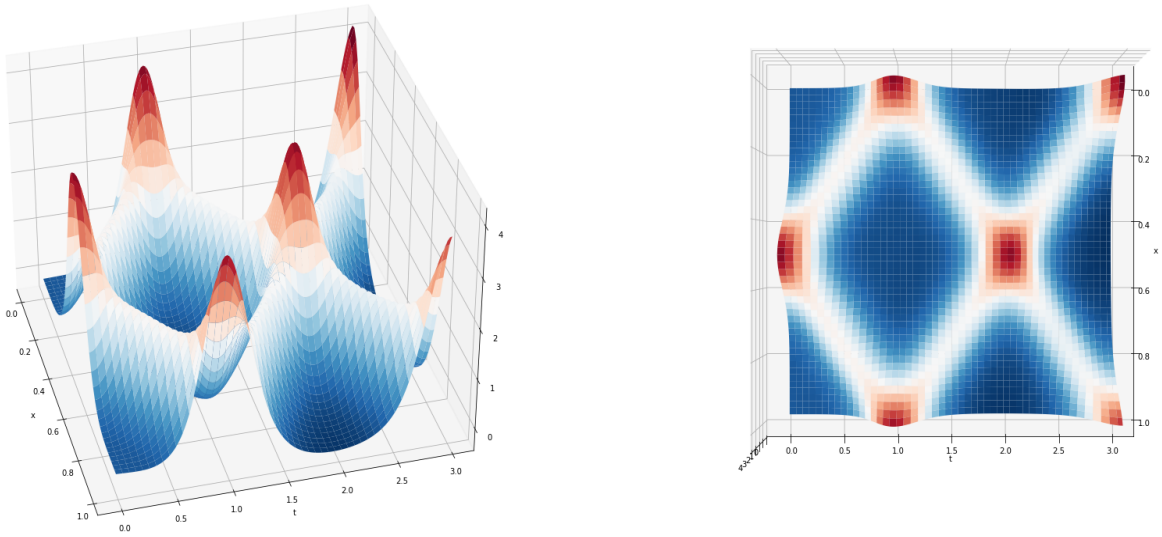

Figure 3. The prediction by neural network at c = 0.5 (m/s) and t from 0-3 (sec).

To verify our neural network, we overplotted the results generated from the network and the ones from the numerical solver with the same initial conditions. The results are shown below. We can see in Figure 5 the results from our network fits the numerical results very well at each time step. The mean square error graph Figure 4 further confirmed our findings. Although the error is increasing over time, the values are still within an acceptable range.

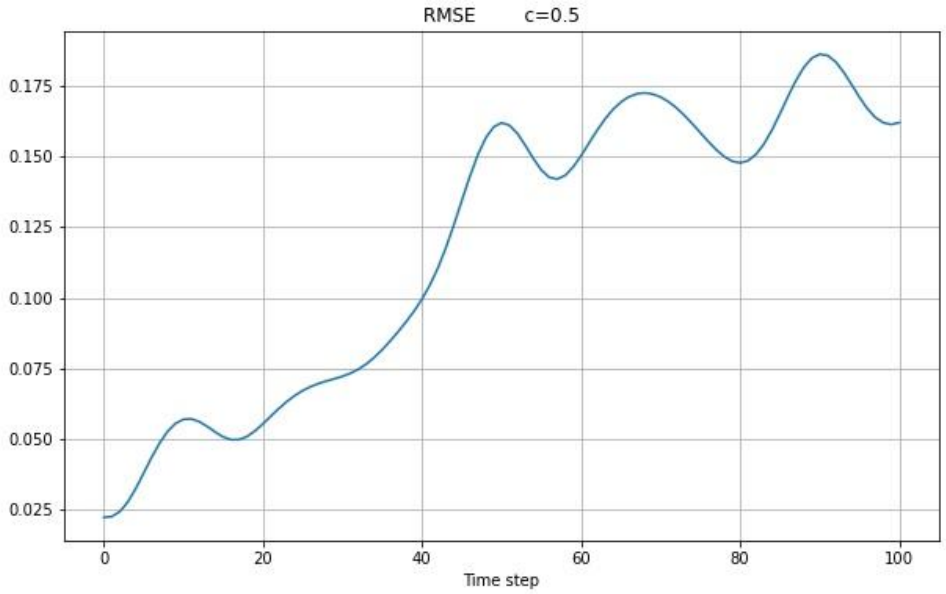

Figure 4. Root Mean Square Error plot

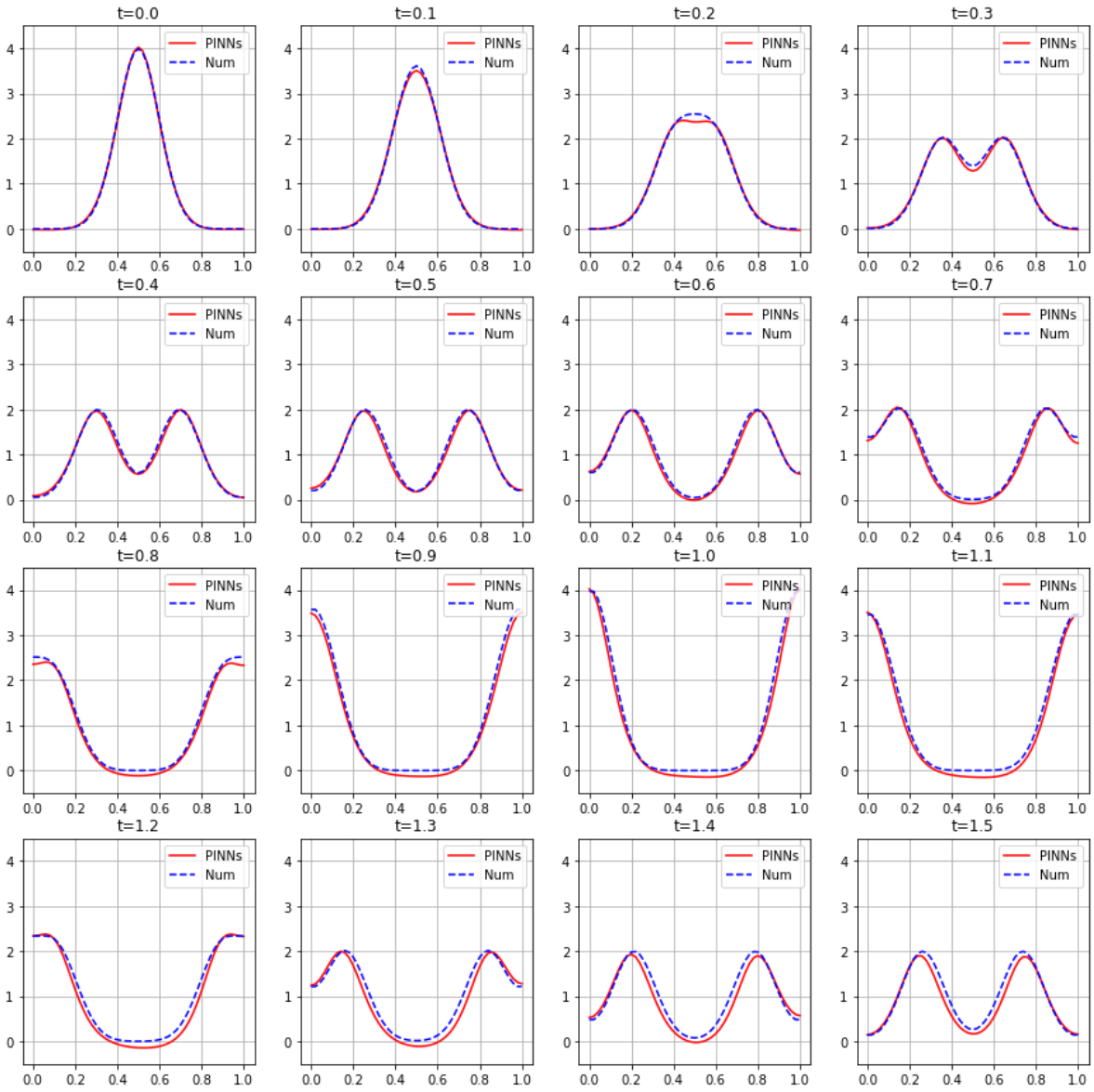

Figure 5. Overplot of predicted results and numerical results at c = 0.5 (m/s).

In our submission package, more graphs and gif animations are available that cover a wider range of c values. While we trained our model at a specific timespan of 0-2 (sec) with the wave speed from 0.1-0.9 (m/s), it was able to obtain decent predictions till 3 (sec) as shown in Figure 3. Moreover, the model can understand what c is, and it can even make reasonable predictions when $c > 0.9$, for example $c = 1.5 m/s$, beyond the range we have in the training data.

Conclusions and Future Work

The model is a good candidate for the prediction of one-dimension wave propagation, verified by several testing cases at different wave speeds, c. It not only has low RMSE but performs well to give a successful conclusion being compatible with results from the numerical solver.

References

[1] Raissi, M., Perdikaris, P. and Karniadakis, G., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378, pp.686-707.

[2] Mishra, S. and Molinaro, R., 2021. Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs. IMA Journal of Numerical Analysis,.

[3] Cai, S., Wang, Z., Wang, S., Perdikaris, P. and Karniadakis, G., 2021. Physics-Informed Neural Networks for Heat Transfer Problems. Journal of Heat Transfer, 143(6).