PINN to predict the oscillations of a damped simple pendulum

Group No. – 0 2

Deepak Petchimuthu, Haitham, Arnav, Omkar, Jakov

Team Introduction

Arnav Anand

Data Set Generation

Using approximation methods create a dataset and split it in test and train datasets

Haitham Narkour

Basic Neural Network

Create and train a basic neural network to predict output for test set and check output

Omkar Kunjir

Physics Informed Neural Network

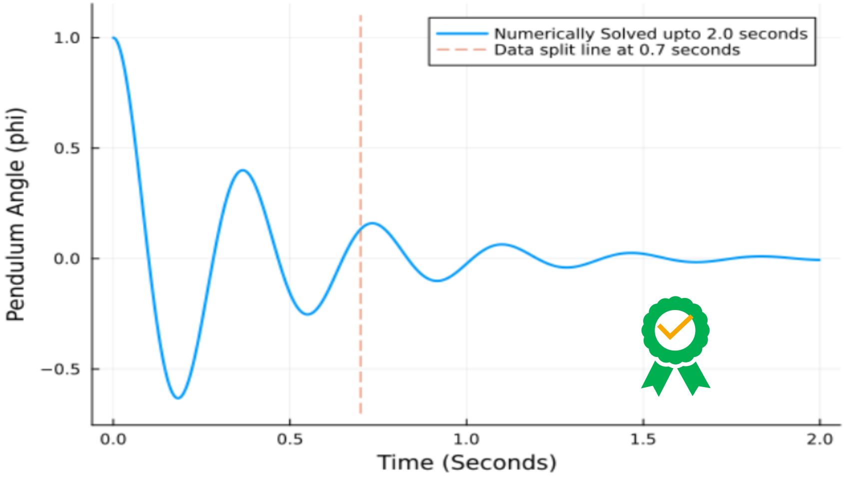
Incorporate the effects of output in the loss function to improve accuracy for test dataset

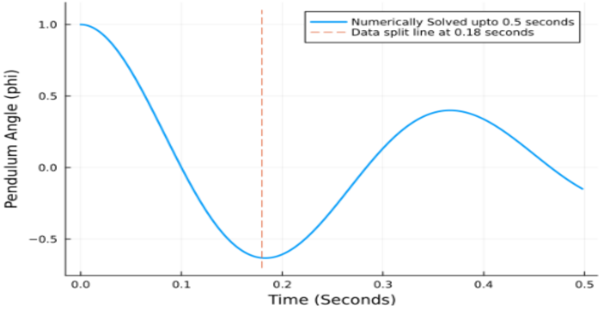
Jakov Bilić

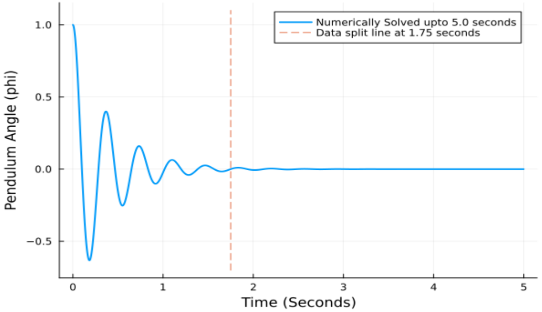
Result Optimization

Optimize architecture of neural networks to improve accuracy & minimize resources.

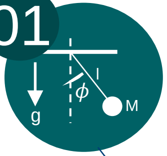
Data Set Generation







01



Generating a mathematical model of a damped oscillation of single mass pendulum

- $\ddot{\phi} + c\dot{\phi} + \omega^2\phi = 0$ with $\omega = \sqrt{\frac{g}{l}}$

02

Solution of ODE by numerical approximation methods:
Euler and Central Difference Methods.
For both methods:

- $dt = 0.001$
- timesteps $= 2000$

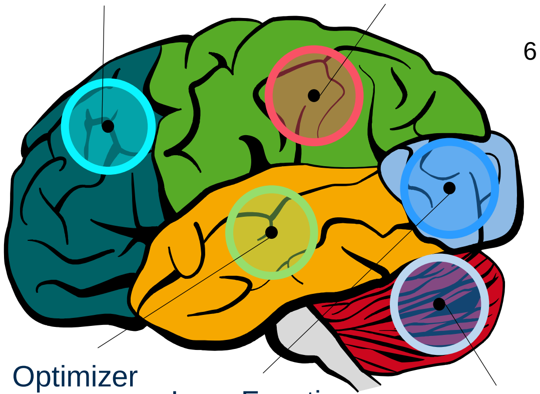
03

35:15 :50

Splitting of data set into 3 parts for train validation and test

- ratio 35:15:50

Basic Neural Network



01 Structure

- Regression Task
- Feed Forward Neural Network
- No. of hidden layers - 3

02 Architecture

- No. of input – **1 (time)**
- No. of output – **1 (phi)**
- No. of Neurons per layer - **40**

03 Activation Function

- Mathematical function applied to the output of a neuron in a neural network to determine its output or activation.
- **Our used function: Tanh**

04 Optimizer

- Method used to adjust the parameters of a neural network during training, in order to minimize the error or loss function
- **Adaptive Moment Estimation (ADAM)**

05 Loss Function

- Function to measure the difference between the predicted and actual output, to be minimized
- MSE = $\frac{\Sigma(X_i - X)^2}{N}$
- **Mean Squared Error – MSE**

Activation functions[1]

Sigmoid

$$f(x) = \frac{1}{1 + e^{-x}}$$

Range: $(0, 1)$

Exponential Linear Unit (ELU)

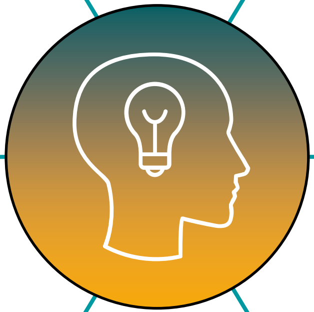
$$f(x) = x, x > 0$$
$$f(x) = \alpha(e^{x} - 1), x <= 0$$

Range: $(-\alpha, \infty)$

Rectified Linear Unit (ReLU)

$$f(x) = max(0, x)$$

Range: $(0, \infty)$



Softmax

$$f(x) = \frac{e^{x_i}}{\Sigma e^{x}}$$

Range: (0, 1)

Swish

$$f(x) = x * sigmoid(x)$$

Range: $(-\infty, \infty)$

tanh

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Range: $(-1, 1)$

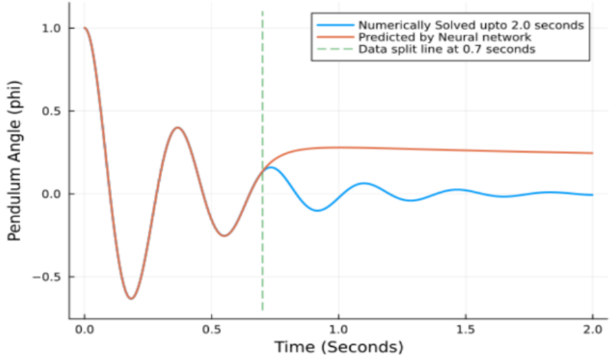
The activation function tanh() is best suited for our case because

- The range of the function is within the range of our generated data
- It is not a linear graph and hence tanh() can work better than some linear activation functions

The same is later confirmed by several iterations that tanh() gives the best output.

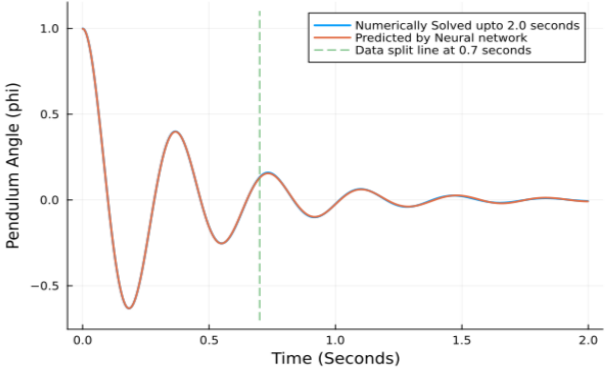
1-Deep learning

Output of Basic Neural Network & PINN



$$\ddot{\phi} + c\dot{\phi} + \omega^2\phi = \mathbf{R}$$

Total Loss = MSE + α * Physics loss



After the training set the neural network is unable to predict the results because the data provided is not sufficient to predict further.

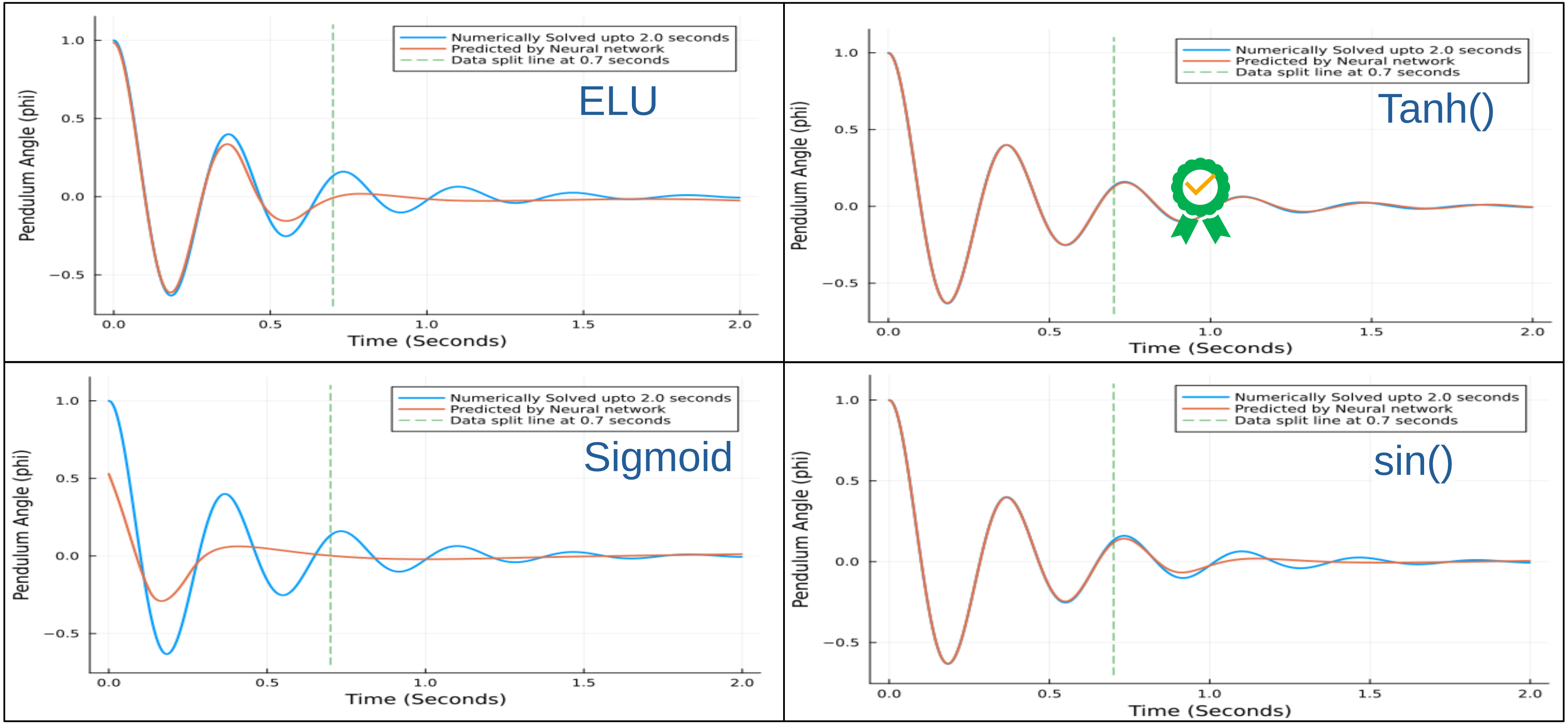
The output of the predicted values must be fed in the original equation and the deviation from required value (0) must be calculated.

This leads to a new term in the loss function and the neural network is trained accordingly to minimize the total loss. - **PINN**

Total Loss = MSE + α*Physics loss

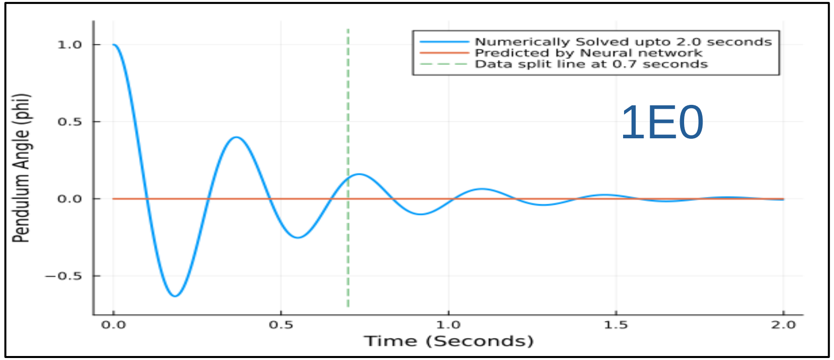
$$\frac{\sum\left(O_{NN} - Y(x_i)\right)^2}{N} + \alpha * \frac{\sum R\left(O_{NN}(x_i)\right)^2}{M}$$

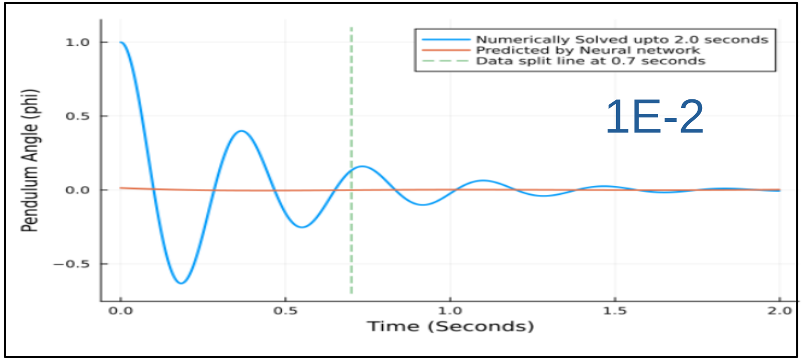
Result Optimization – Activation Function

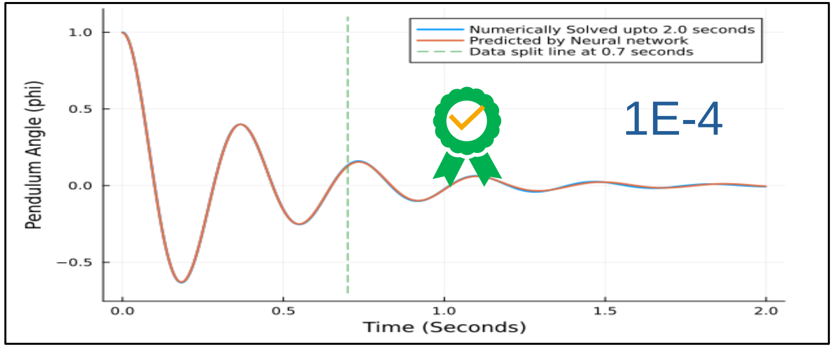


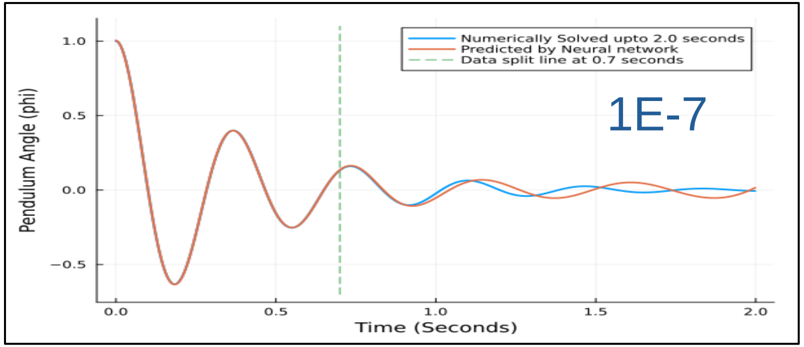
Result Optimization – % Physics Loss

Total Loss = MSE + α * Physics loss

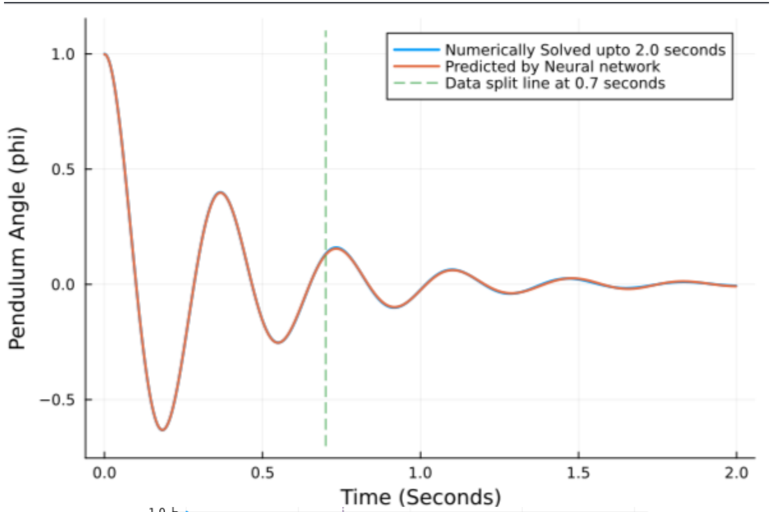


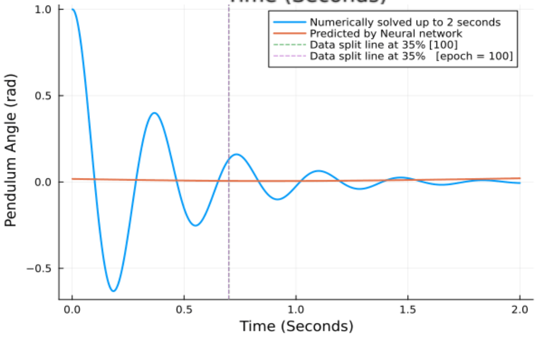






Results





Structure
- 5 layers with 40 neurons
- tanh()
- 1 input and 1 output

Optimizer
- Adam()

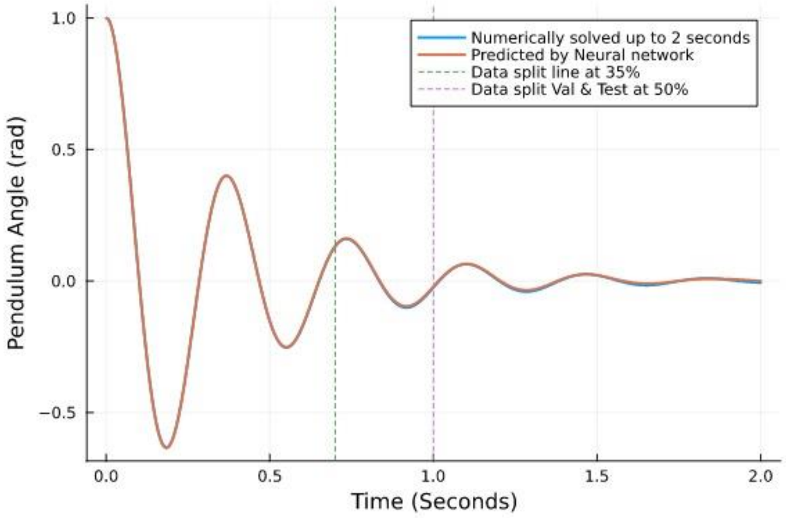
Physics Loss
- 1E-4

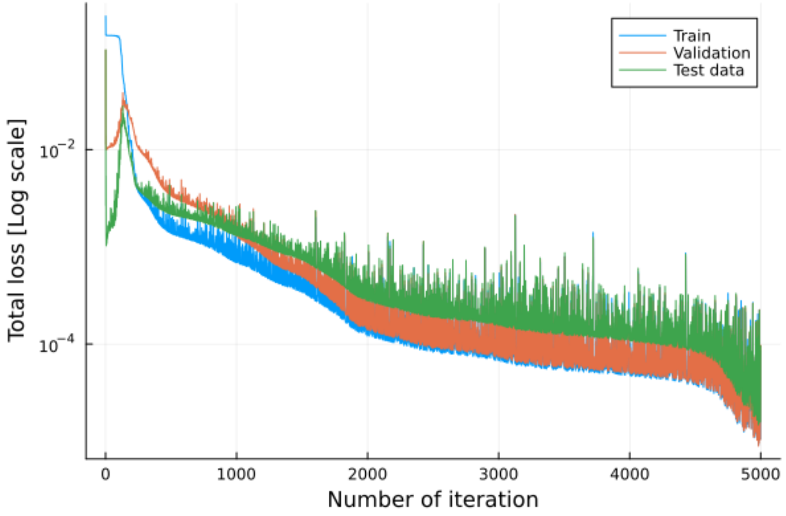
EPOCH
- 5,000

Loss
- 8.4675E-5

Simulation Time
- 01:50 => Data Generation
- 08:15 – 08:30 => Training

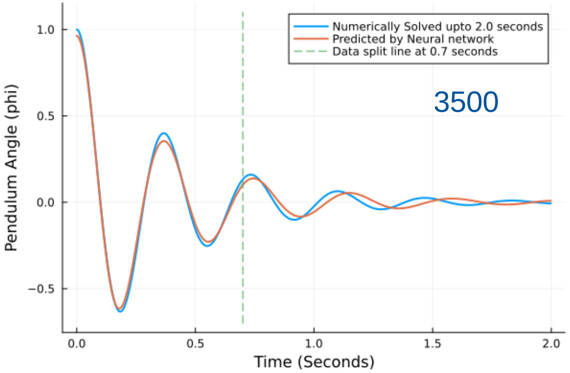
Validation and Loss

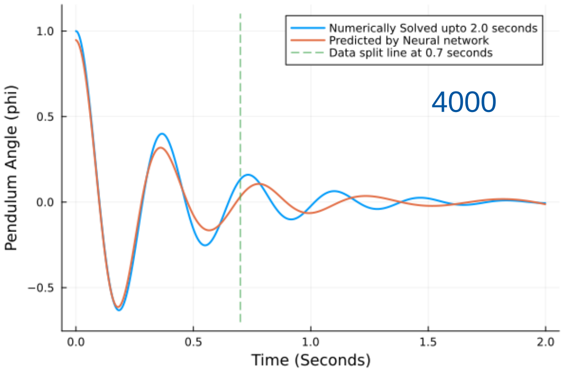


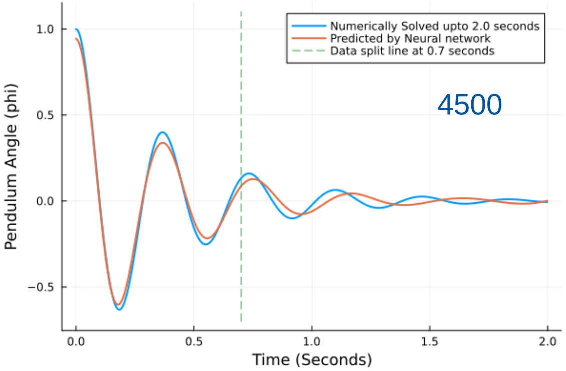


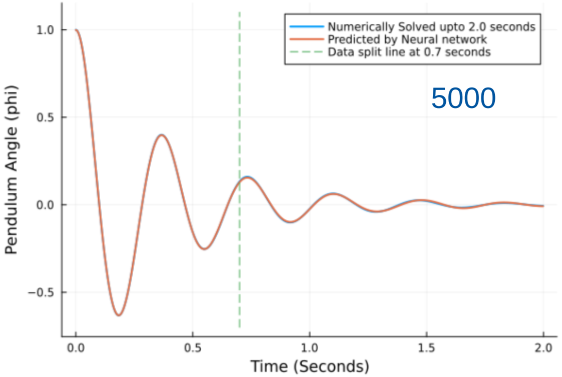
- Splitting dataset in the ratio of 35:15:50 for including validation set
- Plotting the Total loss v/s Epochs on a logarithmic scale

Findings









Observations

- We observed that training with Epochs had a cyclic pattern
- The results improved till 3500 Epochs and then decreased after that till 4500
- Then again they improved and were accurate at 5000 Epochs

Probable Reason

- Tendency to overfitting when running for several Epochs
- Then it recognises it and finally improves to get the result.

Thank you for your attention!

Resources

Books:
1- Deep learning book by Iyan Goodfellow, Yeshwa Bengio, and Aaron couville.
2- Pattern Recognition and machine learing by christopher M.Bishop
3- An Introduction to statistical learning- second edition by Garith James et all

Lectures:
Lectures on Computational Intelligence in Engineering-Institute for general mechanics at RWTH Aachen university. Semester WS 22/23.

Youtube:
4- Steve Brunton channel on Machine learning and physics informed neural network:
https://www.youtube.com/watch?v=7n7xaviepKM&list=PLMrJAkhIeNNQ0BaKuBKY43k4xMo6NSbBa

Websites:
5- The Asimov institut- The neural network zoo:
https://www.asimovinstitute.org/neural-network-zoo/

Presentation:
6- I9 presentations by Joe Smith
7- PPT Template by Computational Intelligence in Engineering-Institute for general mechanics at RWTH Aachen university