GITHUB

https://github.com/apurvmishra22/PINN_ODE_PDE

Artificial Neural Network Approach for Solving Ordinary and Partial Differential equations

In this project a method to solve initial and boundary value problems using artificial neural network is implemented. The aim is to make use of Neural Nets to improve the solution of Differential Equations to greater accuracy and reduce computation times compared to other methods such as, implicit, explicit, semi-implicit, Runge- Kutta, etc. The network is trained to satisfy the differential equation along with initial/boundary conditions. The applicability of this approach ranges from primary single order ODEs to Higher order ODEs and even extending to linear/ non-linear higher order PDEs.

The neural network approach is used to solve 3 ordinary differential equations (2 first order and 1 second order) and 2 second order partial differential equations(1 linear and 1 non-linear). In the neural network, one hidden layer with one node and one output layer for all cases is used. For the hidden layer sigmoid function as activation function

(sigmoid function: σ(x) = $\frac{1}{1+\exp(-x)}$ Is used. We use linear output layer. To perform error or loss minimization, gradient descent is used. For each test problem analytic solution was known in advance. So, from analytic solution and solution from neural network, mean square for a dataset is calculated.

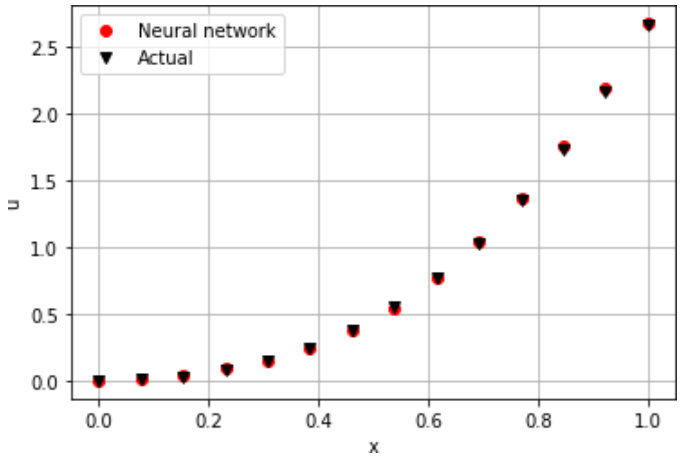
The physics-informed neural network is able to predict the solution far away from the experimental data points, and thus performs much better than the naive network. One could argue that this network does indeed have some concept of our prior physical principles.
The naive network is performing poorly because we are “throwing away” our existing scientific knowledge; with only the data at hand, it is like trying to understand all of the data generated by a particle collider, without having been to a physics class.
Back-propagation is an automatic differentiation algorithm for calculating gradients for the weights in a neural network graph structure. **Gradient descent (GD)** is an iterative first-order optimisation algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning(DL) to minimise a cost/loss function (e.g. in a linear regression). Using this algorithm I have made the project of Physics informed neural network.

Result:

Problem 1:

- Weights: (3.30615780380631, 4.75394647637002)
- Bias: (-2.97884561000171, -0.0880776078141419)
- Mean square error: 0.0002

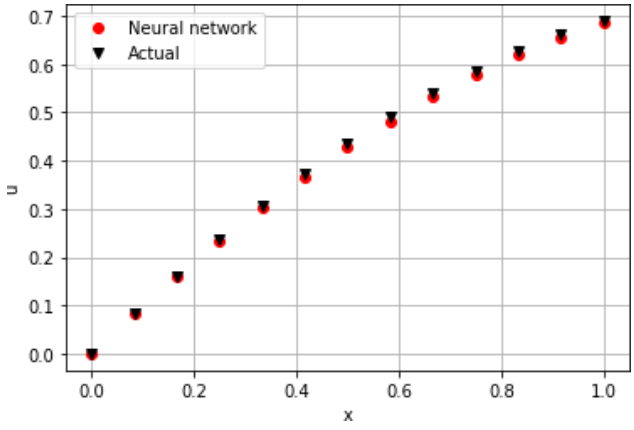
Plot of Actual Result and result obtained from neural network



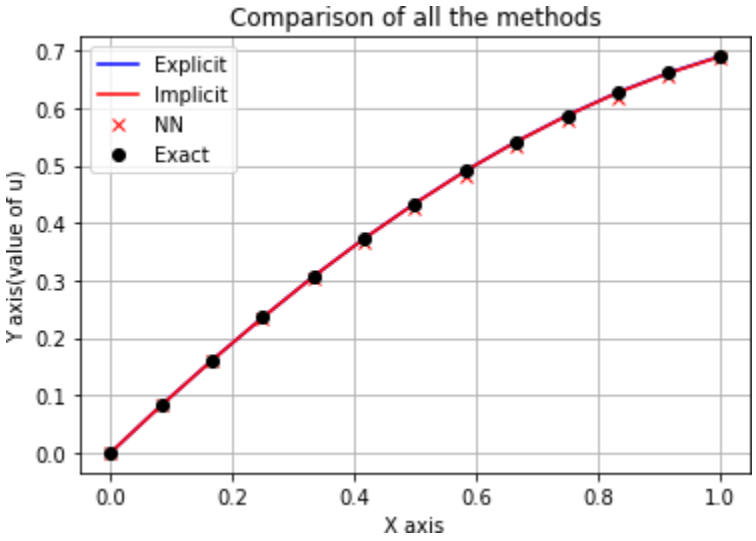
Problem 2:

- Weights: (-1.04285571443650, 1.28345356424041)
- Bias: (0.709525474405509, 0.151656425869594)
- Mean square error: 0.000037

Plot of Actual Result and result obtained from neural network



Comparison of existing methods (Numerical) vs Neural Net vs Exact solution-

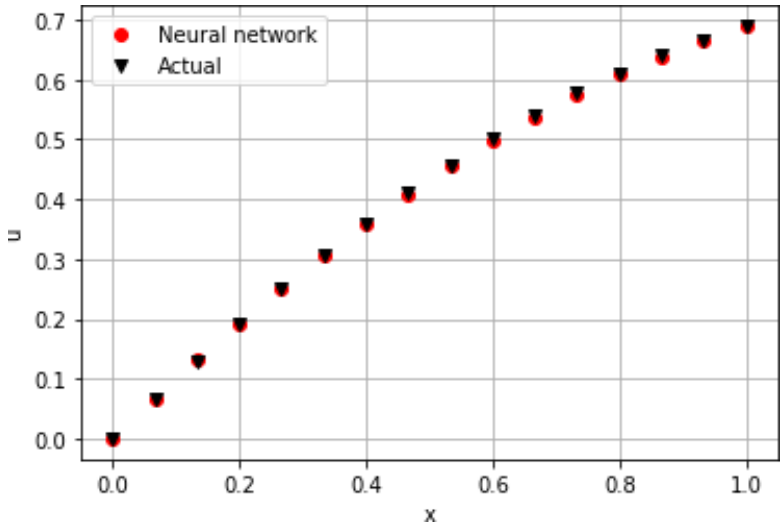


Problem 3:

- Weights: (0.124171042670933, 0.622017541230735)
- Bias: (0.708725124575733, -0.0751307199391313)

- Mean square error: 0.0000047

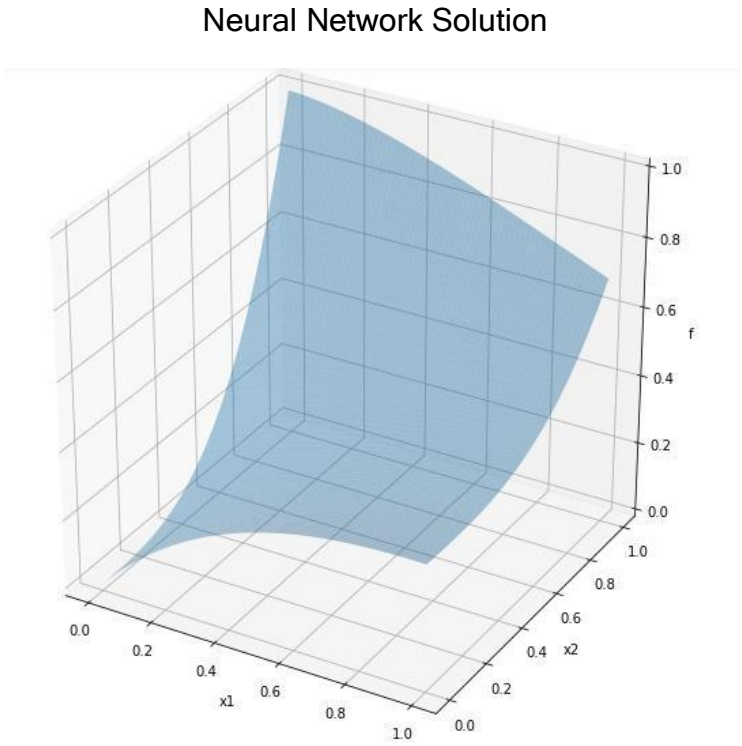
Plot of Actual Result and result obtained from neural network



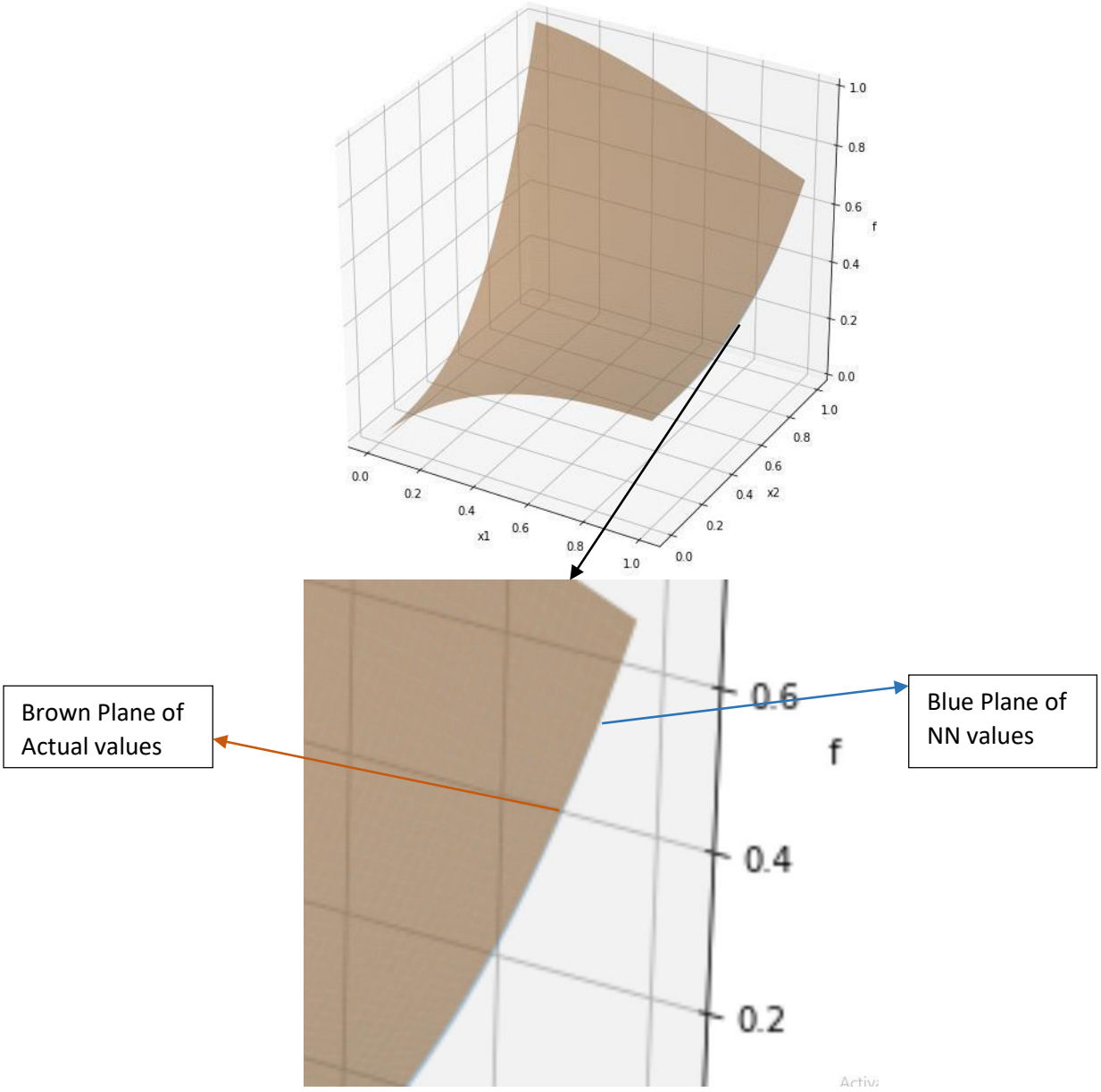
Problem 4:

- Weights: (0.100117108363215, 0.835623668247043, 0.564883550462209)
- Bias: (0.206146353119067, 0.177866882546691)

- Mean square error: 0.0000097



Comparison between NN and actual values

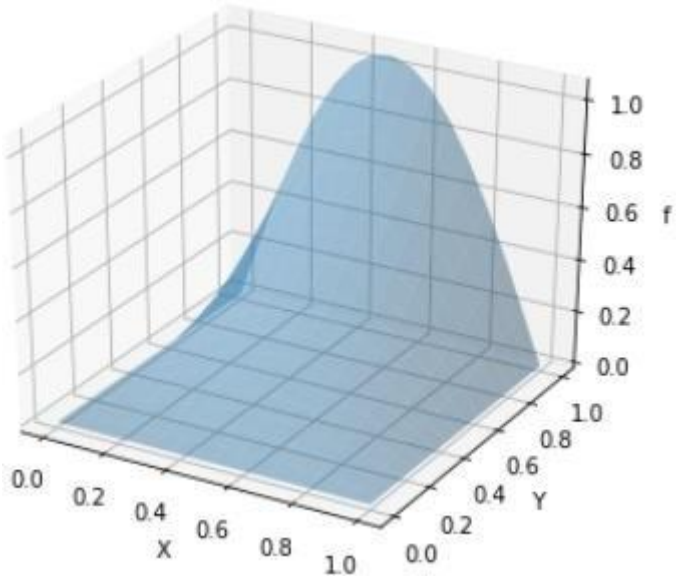


Problem 5:

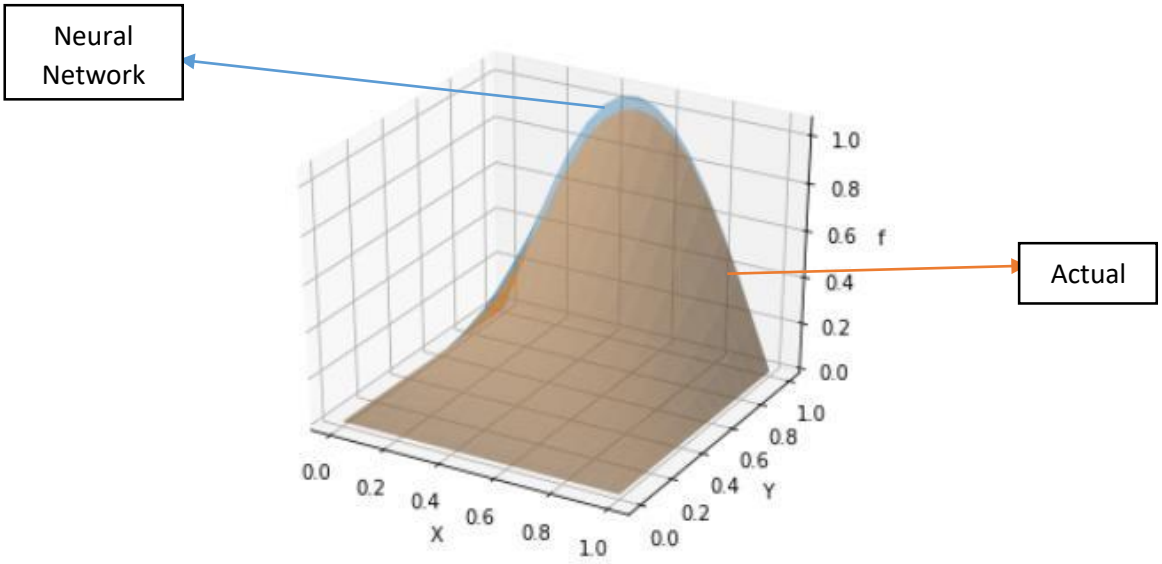
- Weights: (-2.05347124629913, 3.28229338165642, 4.63066783342696)
- Bias: (-2.13368418787738, 0.495435087091941)
- Mean square error: 0.00182

203109004 SELF PROJECT Physics Informed Neural Network

Neural Network Solution



Comparison between NN and actual values



Conclusion

In this project, a method based on Artificial Neural Network (ANN) with one hidden layer and sigmoid function as activation function is used to compute first and second order linear and non-linear ordinary and partial differential equations. This method is general and can be applied for any order of ODEs and PDEs, but with increasing the order of differential equations test set size will be increased that will lead to increase in computational time and we have to increase the no of nodes in the hidden the layer to get better accuracy. It is observed that ANN solutions are satisfactorily agreed with the exact solutions. This algorithm allows us to get good approximation results for complex differential equations without taking much computational time. This project involves comparison of result from neural network with explicit, implicit, and Runge-kutta method aswell.