COMP535: Approximating PDEs Using Neural Networks – Applied to Navier-Stokes Equations

Mohamed Abdelrahman

Problem Definition

Some Partial Differential Equations (PDEs) are hard to solve analytically and usually takes long time to reach such solution (if any).

Related Work

Neural networks were used to approximate PDEs [1] with a known exact solution. The elliptic Laplace's equation were approximated [2] using Artificial Neural Networks (ANN). A trial to approximate Stokes equations using ANN [3] with five hidden units and one linear output unit, and it was compared to a linear programming solution [4] to the problem.

Project Overview

Using Neural Networks to approximate Navier-Stokes equations. These equations play a key role in Computational Fluid Dynamics (CFD). It has two equations for the velocity components u,v and one equation for pressure p:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\nu\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\nu\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)$$

$$\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}=-\rho\left(\frac{\partial u}{\partial x}\frac{\partial u}{\partial x}+2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}+\frac{\partial v}{\partial y}\frac{\partial v}{\partial y}\right)$$

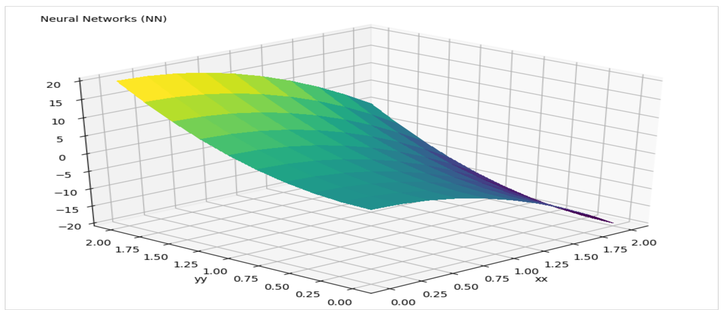
Project Steps

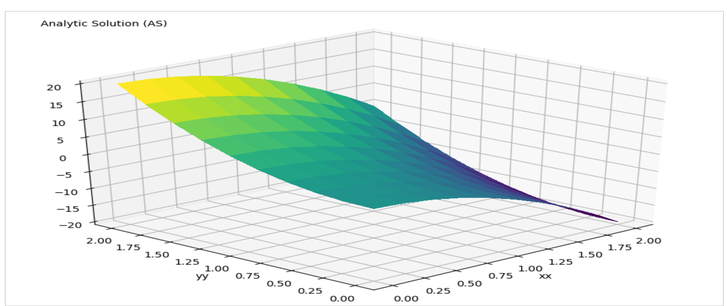
- Implemented the analytic/exact solution suggested by [3] in Python.
- Implemented the ANN with reference to the code developed by [5] to solve partial differential equations.
- Tried different ANN with different hyperparameters (# of layers, # of hidden units, Learning rate) for # of Points (100,400,2500)

Running Environment

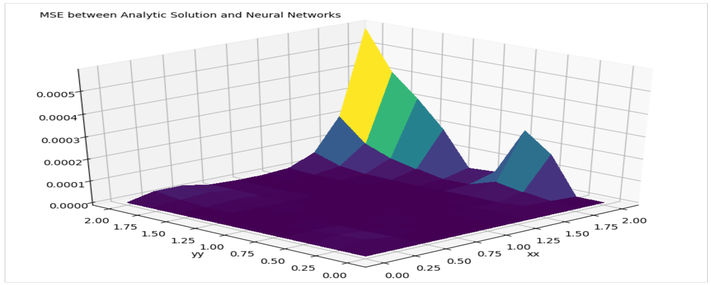
System	**Dell XPS-15-9550 Ubuntu 17.10 4x Intel Core i7 2.6 GHz**
Python Version	**2.7.14**
Number of Layers	**3 (Input, hidden, output)**
Number of Hidden Units	**10**
Learning Rate	**0.001**

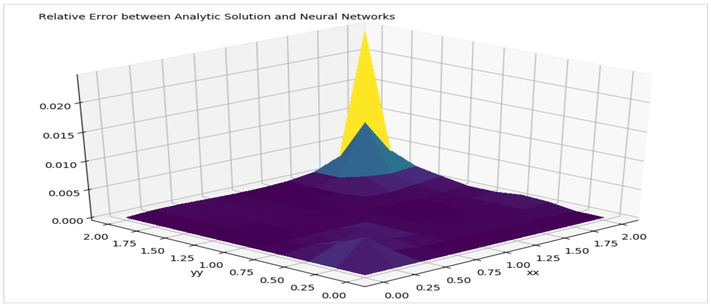
Training Results





Evaluation





Future Work

- Compare the results with a numerical solution. For example, [6] which is based on finite differences.
- Extend the work done to replace the numerical components of the OpenIFS model.

References

[1]Lagaris, I.E., Likas, A. and Fotiadis, D.I., 1998. Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks, 9(5), pp.987-1000

[2]Chiaramonte, M.M. and Kiener, M., Solving differential equations using neural networks. Machine Learning Project.

[3]Baymani, M., Kerayechian, A. and Effati, S., 2010. Artificial neural networks approach for solving stokes problem. Applied Mathematics, 1(04), p.288.

[4]Aman, M. and Kerayechian, A., 2004. Solving the Stokes Problem by Linear Programming Methods. International Mathematical Journal, 5(1), pp.9-22.

[5]https://becominghuman.ai/neural-networks-for-solving-differential-equations-fa230ac5e04c

[6]http://lorenabarba.com/blog/cfd-python-12-steps-to-navier-stokes/