Proceedings of 13th IOE Graduate Conference
Peer Reviewed
ISSN: 2350-8914 (Online), 2350-8906 (Print)
Year: 2023 Month: April Volume: 13

Development of a Neural Network to Predict Path of an Object in a Two-Dimensional Potential Flow

Rijan Niraula [a], Rajendra Shrestha [b], Laxman Poudel [c]

[a, b, c] *Department of Mechanical and Aerospace Engineering, Pulchowk Campus, IOE, Tribhuvan University, Nepal*
[a] 076msmde014.rijan@pcampus.edu.np, [b] rsfluid@hotmail.com, [c] laxman@ioe.edu.np

Abstract
Neural networks have been widely used in various fields, including fluid dynamics, to predict complex phenomena that are difficult to model analytically. In this research, a neural network is developed to predict the path taken by a circular body in a two-dimensional fluidic domain. The study involves simulating the potential flow over a rectangular domain inside which a circular body is placed. Fluctuations in different parameters such as pressure, forces, and velocity field during the motion of the body are studied. The Laplace equation is solved at each time step by applying the techniques of finite element method (FEM) to obtain accurate data, which is fed into the neural network. The neural network comprises of three layers input , middle and output layer. The study is carrried out using computational methods that relies on open-source software Python and its modules like NumPy. The results of the neural network's predictions are compared with accurate data to analyze the error. Fluctutaion of error with respect to different hyperparameters of the network is calculated and accordingly suitable hyperparameters of the network are determined.

Keywords
Neural Network, FEM, Potential Flow, Trajectory

1. Introduction

Fluid dynamics field involves solving set of Partial Differential Equations (PDE) for understanding the behavior of fluids. Most of the problems in fluid dynamics are complicated and difficult to solve. In most cases, these equations cannot be solved analytically. While experimental methods can be used for accurate results, they are expensive and require specialized facilities. As a result, computational methods have become popular, such as Finite Element Method (FEM) and Finite Difference Method (FDM). These methods transform complex differential equations into algebraic equations, which can be solved with the help of computers [1]. A newer method that combines modern numerical techniques and high-speed digital computers is the use of neural networks to solve PDEs. [2]

In this research, the focus is on development and use of neural networks to predict the path of a circular body in a fluidic domain by solving the Laplace equation at each time step and feeding the obtained data into a neural network . The two dimensional fluid domain is assumed to be inviscid, incompressible and irrotational. The neural network is based on gradient descent algorithm and is able to predict the path of the body given its initial location in two dimensional domain [3] . Data is generated by numerically solving laplace equation [4] using the techniques of finite element method (FEM), thus simulating a potential flow over the rectangular domain and analyzing fluctuations in different parameters during motion of the body. Evaluation of the error between the actual and predicted path by the network is also carried out and finally, hyperparameters of the network are tweaked accordingly to minimise error in the network.

2. Research Methodology

The study was divided into two parts: Calculation of field information using finite element method (FEM) and development of the neural network. Both analysis are carried out in Python [5] using python module like Numpy [6]. In FEM, strong form and weak form of the governing equations are developed and are applied to each cell of the mesh tranforming the PDE into the set of algebraic linear equations. For incompressible, inviscid, and steady state irrotational flow, governing equations [7] for fluid flow are given by equation 1 and equation 2.

$$\nabla^2\phi = 0 \text{ with } \vec{V} = \nabla\phi \tag{1}$$

$$\rho(\vec{V}.\nabla)\vec{V} = -\nabla \mathrm{p} + \rho\mathbf{g} \tag{2}$$

Once the input parameter of the problem are specified in the program, the program creates a rectangular flow domain, and a mesh is generated over it. After the users input all the required information, like boundary and wall condition, the program prepares the mesh structure, applies the wall and boundary condition to solve the problem and obtain solution of the potential function $\phi(x,y)$. The technique of finite element method (FEM) is used to obtain this solution [8]. Such solution of the potential fucntion ϕ(x, y) is calculated by varying the location of the object throughout the domain. After obtaining the solution of the Laplace equation, the gradient ∇ is applied to potential fucntion $\phi(x,y)$ to obtain the velocity field information. The techniques of the Finite Difference Method (FDM) (Second Order) method are used to calculate the involved derivatives [9]. From the velocity field, the distribution of pressure is calculated throughout the domain using equation 2. The obtained data are stored, which will be further used to calculate actual trajectory

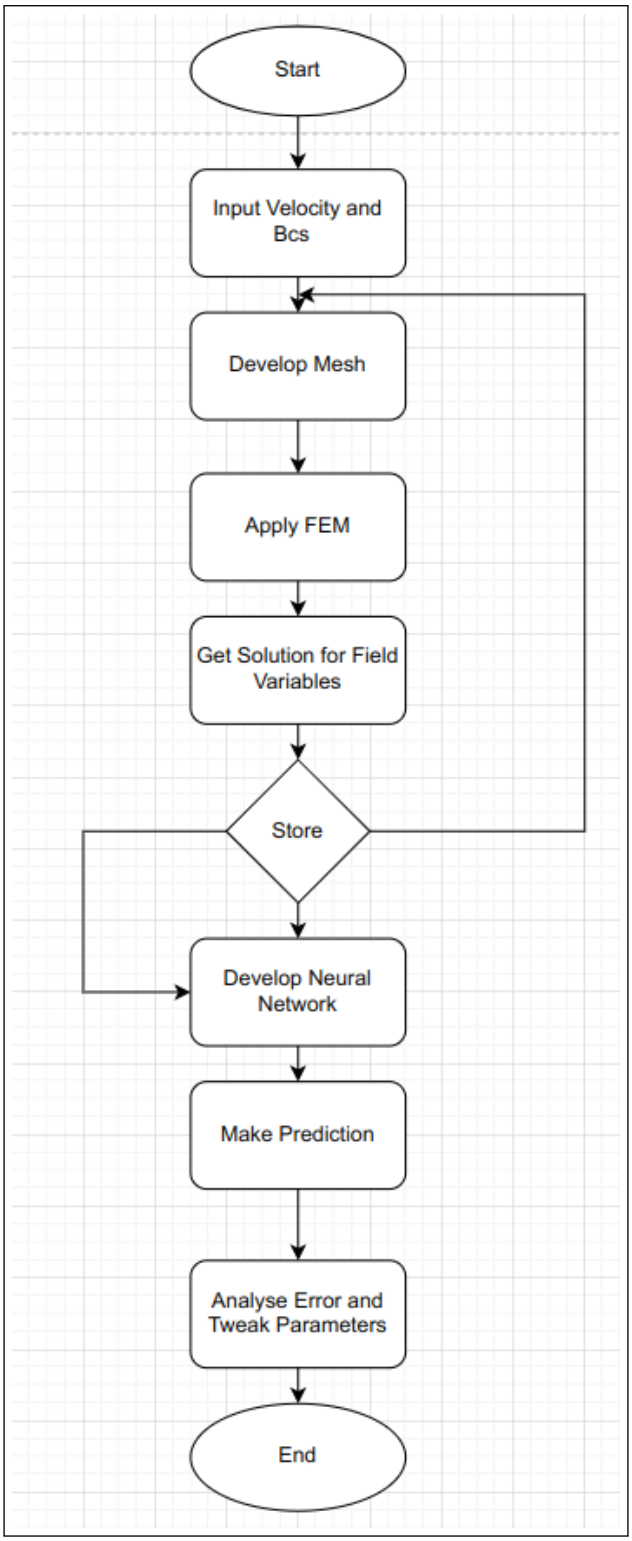

Figure 1: Methodology chart

of the body and later by a neural network for correcting its weights and biases for prediction purpose.

The next section involves the development of the three layered neural network i.e input, output and middle layer [10]. The stored data is divided into feed and test data, and the neural network is constructed by using the gradient descent algorithm. The predicted result for a given set of test data by the trained neural network is computed. Finally, the predicted path is compared against the actual path, and the observed errors are analyzed by varying various parameters of the neural network like learning rate, number of neurons in middle layer and number of data in the dataset.

3. Results

3.1 Development of the mesh

The region between the circle and the rectangular domain are meshed using triangular meshes, where each mesh cell can be represented in Python by a list of three tuples [11]. In addition, the object within the domain is placed at different locations, and meshes are created for each of these as well. This is necessary in order to accurately calculate the trajectory of each object, taking into account its position and movement within the meshed domain. The length and breadth of the rectangular domain is taken to be 1m each and radius of the body is taken as 0.1m. Such meshes are shown in figure 2.

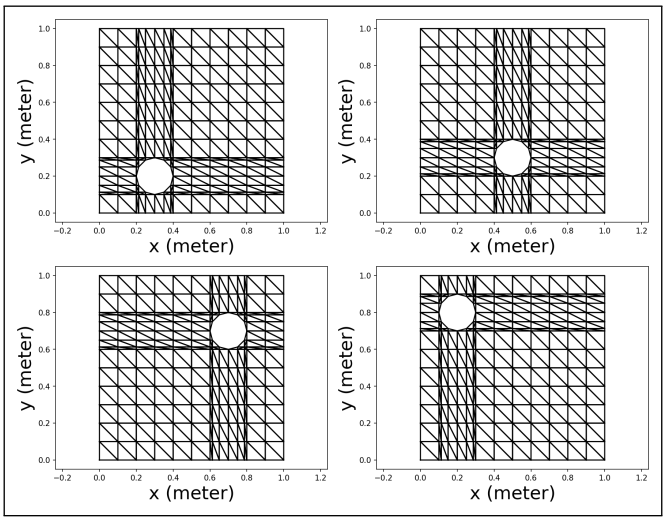

Figure 2: Mesh for different locations

3.2 Velocity Field

After applying the weak form of the governing equation to each cell in the mesh, one obtains the algebraic linear equation for the potential $\phi(x,y)$ function. Solving the linear equation gives the solution of potential function $\phi(x,y)$ over the whole domain. Taking its gradient, the velocity field information is obtained. One such velocity field graph is shown for a specific location.

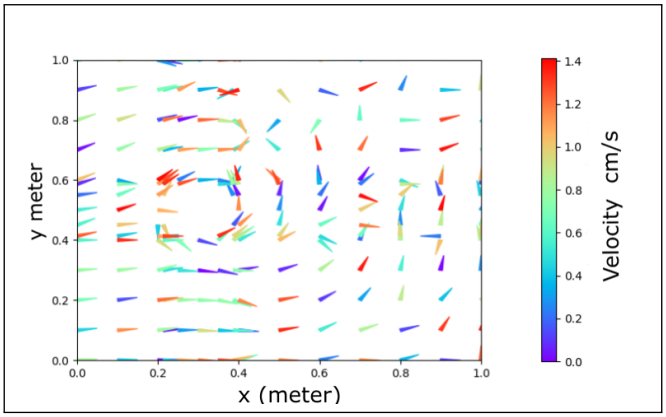

Figure 3: Velocity Field

Table 1: Path Co-ordinate table

S.N	Start Position	Path Co-ordinates
1	(0.2,0.2)	(0.25,0.28),(0.31,0.5),(0.41,0.68)
2	(0.5,0.3)	(0.56,0.72),(0.83,0.53),(0.94,0.64)
3	(0.6,0.4)	(0.61,0.43),(0.64,0.50),(0.67,0.58)

3.3 Force and Path Trajectory

The pressure around the perimeter of the circular body is calculated via equation 2. The initial condition of the pressure is assumed to be at atmospheric pressure at the outlet. Integrating this pressure values along the perimeter of the body gives the force and acceleration experienced by the body at a specific location. Basic kinematic equations are used recursively to obtain the path co-ordinates of the body.

3.4 Mean Square Error (MSE)

In neural networks, Mean Squared Error (MSE) [12] is a commonly used metric to evaluate the performance of the model. It measures the average squared difference between the predicted output and the actual output over all the samples in the dataset.

3.4.1 MSE vs Learning rate

The key step in optimizing the performance of the network is by seeing how MSE varies with different learning rate for the network. Learning rate is a hyperparameter of the network that determines the step size taken in the direction of the negative gradient during neural network training. By monitoring the MSE during training with different learning rates, one can determine an optimal learning rate that leads to the best performance on the given dataset.

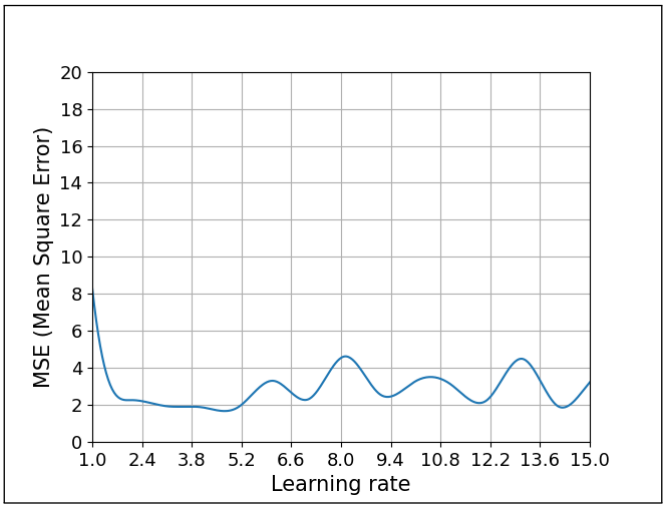

Figure 4: MSE vs Learning Rate

Figure 4 shows that MSE is minimum and stable when learning rate is in the range of 1.6-3.5. Thus learning rate of 2 is employed in the network. Learning rate fluctuates beyond the value of 3.5 which implies selection of learning rate from this region has chances of incurring high MSE for a random sample dataset.

3.4.2 MSE vs Number of Neurons

To understand the impact of the number of neurons present in the middle layer on the network performance, an examination of how the MSE varies with different numbers of neurons is carried out. A random shuffle of the dataset is done by selecting 80 percent of the data and calculating MSE vs number of neurons in middle layer for various dataset.

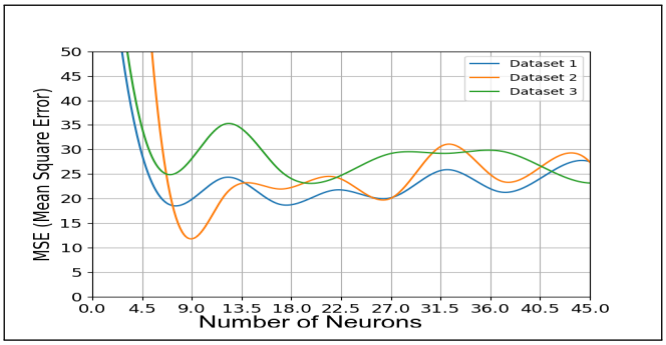

Figure 5: MSE vs Number of Neurons

Figure 5 shows that MSE is minimum, when number of neurons in middle layer is in the region of 10. Minimum of MSE also occurs at other values but higher number of neuron will imply higher computational time by the network. Thus, number of neurons in the middle layer is selected to be in the range of 8-12.

3.4.3 MSE vs Number of Dataset

An analysis on the impact of dataset size on model performance is calculated by noting how MSE varies for different numbers of data samples through which one can determine the minimum amount of data required for the model to achieve acceptable performance.

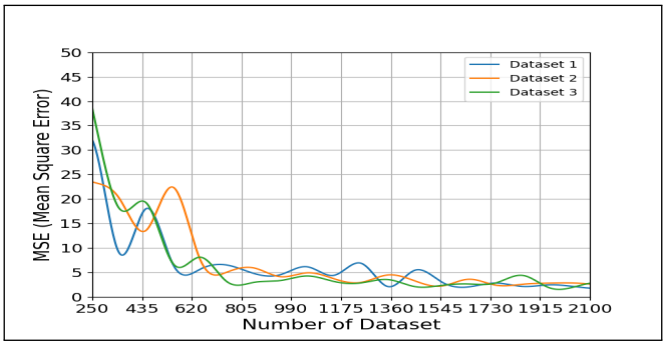

Figure 6: MSE vs Number of Data samples

Figure 6 shows that this number is in range of 2000 for the network.

4. Conclusion

Based on the numerical calculation of the research, a neural network capable of predicting path of an object in potential flow has been developed. Additionally, the optimal values for hyperparameters of the network are calculated. This hyperparameters range can guide guide the design of future neural network models for similar problems. The development

of the neural network required numerical computation of the velocity and pressure fields, and the resulting model provides faster results compared to numerically solving the governing equations. These findings suggest that neural networks can be a valuable tool for accelerating the design and optimization of fluid dynamic systems. Overall, this research has demonstrated the potential of neural networks in predicting the path of an object in a potential flow, and provides a foundation for future research in this area.

References

[1] Yue Liu and Xiangdong Gao. Numerical simulation of fluid mechanics problems using fdm, fvm, and fem. *Procedia Engineering*, 99:2092–2100, 2015.

[2] Steven L Brunton and J Nathan Kutz. Neural networks for solving differential equations. *Annual Review of Fluid Mechanics*, 51:539–574, 2019.

[3] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA, 2015.

[4] Xianming Dai and Yunjie Yang. The laplace equation and its solutions. *Procedia Computer Science*, 130:336–343, 2018.

[5] Naveen Kumar and Sheetal Taneja. Python programming: A modular approach, 2018.

[6] Ivan Idris. *NumPy*. Packt Publishing Ltd., Birmingham, UK, 2018.

[7] Yunus Cengel and John Cimbala. *Fluid Mechanics: Fundamentals and Applications*. McGraw-Hill Education, 2006.

[8] Ioannis Koutromanos. *Fundamentals of Finite Element Analysis: Linear Finite Element Analysis*. John Wiley & Sons, 2018.

[9] L Cheng. Finite difference methods for poisson equation, 2022.

[10] Chris M Bishop. Neural networks and their applications. *Review of scientific instruments*, 65(6):1803–1832, 1994.

[11] Cody J. Friesen. *Finite Element Mesh Generation with Python*, pages 89–102. Elsevier, 2018.

[12] Wei Shi and Charles Qi. Mean squared error: A review of applications in deep learning. *arXiv preprint*, 2021.