PHYSICSAccess Research Paper
ISSN Online: 2756-3898, ISSN Print: 2714-500X https://doi.org/10.47514/phyaccess.2025.5.1.014

Physics-Informed Neural Network for Solving 2D Steady Incompressible Navier-Stokes Equations: Application to Poiseuille Flow

Peter Anthony[1], Philibus M Gyuk[2] and Isaac H Daniel[2]

[1] Department of Mathematical Sciences, Kaduna State University, Kaduna, 900211, Nigeria
[2] Department of Physics, Kaduna State University, Kaduna, 900211, Nigeria

Corresponding E-mail: p.anthony@kasu.edu.ng

Received 13-05-2025
Accepted for publication 17-06-2025
Published 18-06-2025

Abstract

This study explores the application of Physics-Informed Neural Networks (PINNs) to solve the two-dimensional steady incompressible Navier-Stokes equations, focusing on Poiseuille flow in a rectangular channel. Implemented using the DeepXDE library with a TensorFlow 1.x backend, the PINN embeds the governing partial differential equations and boundary conditions into its loss function. The neural network architecture consists of four hidden layers, each with 64 neurons. A hybrid optimization strategy, combining Adam (10,000 steps) and L-BFGS-B, ensured robust convergence. The composite loss function, comprising seven components (three for momentum and continuity equations, four for boundary conditions), decreased significantly, achieving a total test loss of 8.71×10^{-5} at step 9000. The predicted x-velocity component (u) was evaluated against the analytical Poiseuille solution, yielding an L_2 relative error of 0.0861 (8.61%). Visual comparisons confirm that the PINN accurately captures the parabolic velocity profile characteristic of Poiseuille flow. This work underscores the potential of PINNs as a data-efficient, mesh-free approach for solving fundamental fluid dynamics problems, paving the way for their application to more complex flow scenarios.

Keywords: Physics-Informed Neural Networks; Navier-Stokes Equations; Poiseuille Flow; DeepXDE; Computational Fluid Dynamics; Machine Learning.

I. INTRODUCTION

The solution of partial differential equations (PDEs) is a cornerstone of scientific and engineering disciplines, underpinning advancements in fluid dynamics, heat transfer, structural mechanics, and beyond [1]. In fluid dynamics, the Navier-Stokes equations, which describe the motion of viscous fluids, are particularly critical due to their ability to model a wide range of physical phenomena, from laminar flows in pipes to turbulent flows in aerospace applications [2]. Traditionally, numerical methods such as Finite Difference, Finite Element, and Finite Volume techniques have been employed to solve these equations [1]. While effective, these methods often require extensive computational resources, high-quality mesh generation, and significant preprocessing, especially for problems involving complex geometries or high-dimensional domains [1]. In recent years, the advent of machine learning, particularly deep learning, has opened new avenues for solving PDEs [3]. Physics-Informed Neural Networks (PINNs) represent a paradigm shift by integrating the governing physical laws directly into the neural network's loss function, combining the universal approximation capabilities of neural networks with the rigour of physical constraints [4]. Unlike traditional methods, PINNs are mesh-

free, allowing for greater flexibility in handling irregular domains, and can incorporate sparse observational data, making them suitable for data-scarce scenarios [3]. Their ability to learn solutions directly from the PDEs and boundary conditions without requiring extensive labelled datasets has positioned PINNs as a promising tool for computational physics [3, 4]. Despite their potential, the application of PINNs to fluid dynamics problems, particularly the incompressible Navier-Stokes equations, remains an active area of research. Many studies have focused on simplified or one-dimensional problems or have relied on synthetic data to validate PINN performance [5, 6]. However, the robustness of PINNs in accurately capturing the complex, non-linear behaviour of two-dimensional (2D) steady flows, such as Poiseuille flow, under realistic boundary conditions, has not been thoroughly explored [7]. Moreover, challenges such as numerical stability at domain boundaries, optimization convergence, and computational efficiency persist, particularly when using frameworks like DeepXDE with older backends like TensorFlow 1.x [8]. These gaps highlight the need for detailed investigations into the practical implementation and performance of PINNs for benchmark fluid dynamics problems.

This study addresses these gaps by applying a PINN to solve the 2D steady incompressible Navier-Stokes equations for Poiseuille flow in a rectangular channel. Poiseuille flow, a classical laminar flow between parallel plates driven by a pressure gradient, serves as an ideal benchmark due to its well-established analytical solution [2]. The primary objective is to demonstrate the PINN's ability to accurately predict the velocity components (u, v) and pressure field (p) while satisfying the governing PDEs and boundary conditions. By leveraging the DeepXDE library and a hybrid optimization strategy, this work aims to provide a robust and reproducible framework for PINN-based fluid dynamics simulations [8, 9, 10]. The novelty of this research lies in its comprehensive evaluation of PINN performance for a 2D steady flow problem, focusing on practical implementation details and quantitative accuracy metrics [5]. Unlike prior studies that often emphasize theoretical formulations or simplified cases, this work provides a detailed analysis of loss evolution, boundary condition enforcement, and predictive accuracy using a realistic Poiseuille flow setup [2]. Additionally, it addresses practical challenges such as boundary-related numerical warnings and TensorFlow deprecation, offering insights into improving PINN implementations [8]. By achieving a low L_2 relative error accurately and capturing the parabolic velocity profile, this study underscores the potential of PINNs as a data-efficient, mesh-free alternative to traditional computational fluid dynamics (CFD) methods, paving the way for their application to more complex flow scenarios [6, 11].

II. METHODS

The PINN was implemented using the DeepXDE library [8] with a TensorFlow 1.x backend, despite noted deprecation warnings.

A. Governing Equations

The 2D steady incompressible Navier-Stokes equations govern the conservation of mass and momentum for a Newtonian fluid. For velocity components u (x-direction), v (y-direction), pressure p, and kinematic viscosity ν, the equations are:

1. Continuity Equation (Conservation of Mass):
$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \quad (1)$$
2. X-momentum Equation:
$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \mu\frac{\partial^2 u}{\partial x^2} + \mu\frac{\partial^2 u}{\partial y^2} \quad (2)$$
3. Y-momentum Equation:
$$u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \mu\frac{\partial^2 v}{\partial x^2} + \mu\frac{\partial^2 v}{\partial y^2} \quad (3)$$

Here, $\rho = 1$ for simplicity, and $\mu = 0.01$.

B. Domain and Boundary Conditions

The computational domain is a rectangular channel defined by $x \in [-L, L],\ y \in [-1, 1], with\ L = 5.0$. The boundary conditions are:

- Inflow at x = −L: A parabolic velocity profile for u:

$$u(-L, y) = 4(1 - y^2) \quad (4)$$

$$v(-L, y) = 0 \quad (5)$$

- No-penetration at Top/Bottom Walls (y = ±1):

$$v(x, \pm 1) = 0 \quad (6)$$

- Outflow at x = L: Zero pressure gradient:

$$\frac{\partial p}{\partial x}(L, y) = 0 \quad (7)$$

Warnings about Rectangle boundary normal called on vertices were observed, indicating potential numerical issues at domain corners. Future work could use PDE(..., exclusions=...) to address this [8].

C. Neural Network Architecture and Training

The PINN uses a feed-forward neural network with:
- 2 input neurons (x, y).
- Fur hidden layers, each with 64 neurons, using tanh activation.
- 3 output neurons (u, v, p).

Weights were initialized using the Glorot normal distribution [12]. The loss function combines residuals of the Navier-Stokes equations and boundary conditions, using 4000 collocation points in the domain, 400 boundary points, and 1000 test points (though 1065 were sampled) [8]. Training followed a two-stage strategy:

1. **Adam Optimizer**: 10,000 epochs with a learning rate of 1×10^{-3} for initial convergence [9].
2. **L-BFGS-B Optimizer**: Fine-tuning for higher precision [8].

III. RESULTS AND DISCUSSION

The training process was evaluated by monitoring the composite loss function, comprising residuals of the three PDEs and four boundary conditions [4].

A. Loss Evolution During Training

Table I shows the training and test losses at selected steps**.** The initial high loss at step 0 (9.49) was dominated by the inflow boundary condition [8]. Losses decreased consistently with the Adam optimizer, reaching a total test loss of 8.71×10^{-5} at step 9000 [9]. The test loss closely tracked the training loss, indicating no significant overfitting [4]. The Adam phase took approximately 10,177.5 seconds (≈ 2 hours 50 minutes), with L-BFGS-B fine-tuning adding 37.15 seconds, yielding minimal further improvement [10].

Table I. Training and test losses at selected steps.

Step	Train Loss Components							Total Train loss
0	1.21×10^{-1}	3.72×10^{-3}	2.28×10^{-2}	9.15	1.93×10^{-2}	1.20×10^{-1}	1.72×10^{-4}	9.49
1000	2.26×10^{-4}	2.98×10^{-4}	7.37×10^{-5}	9.64×10^{-4}	8.31×10^{-6}	3.67×10^{-5}	4.46×10^{-6}	1.61×10^{-3}
5000	2.98×10^{-5}	7.10×10^{-5}	6.47×10^{-6}	4.43×10^{-5}	1.07×10^{-5}	4.92×10^{-6}	4.72×10^{-7}	1.67×10^{-1}
9000	1.23×10^{-5}	4.45×10^{-5}	4.30×10^{-6}	3.32×10^{-5}	9.01×10^{-6}	2.89×10^{-6}	3.30×10^{-7}	1.07×10^{-4}
10000	1.91×10^{-5}	1.16×10^{-4}	6.15×10^{-6}	5.51×10^{-5}	1.70×10^{-5}	3.92×10^{-6}	1.80×10^{-7}	2.17×10^{-4}
Step	Test Loss Components							Total Test Loss
0	1.26×10^{-1}	3.80×10^{-3}	2.25×10^{-2}	9.15	1.93×10^{-2}	1.20×10^{-1}	1.72×10^{-4}	9.49
1000	2.09×10^{-4}	2.43×10^{-4}	6.54×10^{-5}	9.64×10^{-4}	8.31×10^{-6}	3.67×10^{-5}	4.46×10^{-6}	1.53×10^{-3}
5000	2.76×10^{-5}	5.35×10^{-5}	5.77×10^{-6}	4.43×10^{-5}	1.07×10^{-5}	4.92×10^{-6}	4.72×10^{-7}	1.47×10^{-4}
9000	9.81×10^{-6}	2.81×10^{-5}	3.72×10^{-6}	3.32×10^{-5}	9.01×10^{-6}	2.89×10^{-6}	3.30×10^{-7}	8.71×10^{-5}
10000	1.71×10^{-5}	9.77×10^{-5}	5.65×10^{-6}	5.51×10^{-5}	1.70×10^{-5}	3.92×10^{-6}	1.80×10^{-6}	1.45×10^{-4}

B. Predictive Accuracy

The L_2 relative error for the x-velocity component (u) compared to the analytical Poiseuille solution was 0.0861 (8.61%) [4], indicating strong agreement [2].

C. Velocity Profile Comparison

Fig. 1 compares the predicted and analytical velocity profiles at x = 0.

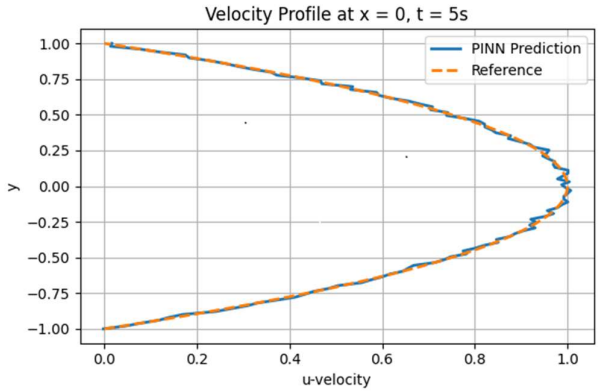

Fig. 1. [PINN-predicted velocity profile (solid blue) versus analytical Poiseuille profile (dashed black) at x = 0].

The PINN accurately reproduces the parabolic velocity profile, with maximum velocity at y = 0 and zero at y = ±1 [2]. Minor boundary deviations align with Rectangle boundary normal warnings, suggesting improved boundary handling as a future enhancement [8].

The PINN effectively solved the Navier-Stokes equations, with all loss components converging to low values [4]. The hybrid optimization strategy (Adam followed by L-BFGS-B) ensured robust convergence [9, 10]. The 8.61% L_2 error and visual agreement in Fig. 1 confirm the PINN's accuracy for Poiseuille flow [2]. TensorFlow 1.x deprecation and boundary condition warnings suggest migrating to TensorFlow 2.x and refining corner point handling for improved performance [8]. The mesh-free nature of PINNs makes them advantageous for complex geometries and sparse data scenarios [6, 11].

IV. CONCLUSION

This study successfully demonstrated the application of a Physics-Informed Neural Network to solve the 2D steady incompressible Navier-Stokes equations for Poiseuille flow in a rectangular channel. The PINN achieved a low total test loss of 8.71×10^{-5} after 10,000 training steps and an L_2 relative error of 8.61% for the x-velocity field. The predicted velocity profile closely matched the analytical parabolic profile, validating the approach. These results highlight PINNs as a powerful, mesh-free tool for fluid dynamics, with potential applications in complex flow scenarios such as turbulent or multiphase flows.

ACKNOWLEDGEMENT

This research was fully funded by Institutional Based Research (IBR) of the Tertiary Education Trust Fund (TETFUND) of Nigeria through Kaduna State University.

References

[1] J. H. Ferziger and M. Peri´c, Computational Methods for Fluid Dynamics, 3rd ed. Berlin: Springer, 2002.
[2] F. M. White, Fluid Mechanics, 7th ed. New York: McGraw-Hill, 2011.
[3] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P.

Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nature Rev. Phy., vol. 3, pp. 422–440, 2021.
[4] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys., vol. 378, pp. 686–707, 2019.
[5] S. Cai, Z. Wang, S. Fuest, Y. J. Jeon, C. Gray, and G. E. Karniadakis, "Flow over an espresso cup: Inferring 3D velocity and pressure fields from tomographic background oriented schlieren via physics-informed neural networks," J. Fluid Mech., vol. 915, A102, 2021.
[6] X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations," J. Fluid Mech., vol. 915, A91, 2021.
[7] L. Sun, H. Gao, S. Pan, and J. X. Wang, "Surrogate modelling for fluid flows based on physics-constrained deep learning without simulation data," Comput. Methods Appl. Mech. Eng., vol. 361, p. 112732, 2020.
[8] L. Wang, X. Zhou, X. Liu, J. Yang, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," SIAM Rev., vol. 63, no. 4, pp. 733–757, 2021.
[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Proceedings of the 3rd International Conference on Learning Representations (ICLR), Banff, 14-16 April 2014.
[10] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," SIAM J. Sci. Comput., vol. 16, no. 5, pp. 1190–1208, 1995.
[11] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," Comput. Methods Appl. Mech. Eng., vol. 360, p. 112789, 2020.
[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 249–256, 2010.