LETTER | DECEMBER 04 2023

Three-dimensional laminar flow using physics informed deep neural networks

Saykat Kumar Biswas ; N. K. Anand (एन. के. आनंद)

Physics of Fluids 35, 121703 (2023)
https://doi.org/10.1063/5.0180834

Three-dimensional laminar flow using physics informed deep neural networks

Cite as: Phys. Fluids 35, 121703 (2023); doi: 10.1063/5.0180834
Submitted: 12 October 2023 · Accepted: 8 November 2023 ·
Published Online: 4 December 2023

Saykat Kumar Biswas and N. K. Anand (एन. के. आनंद)[a)]

AFFILIATIONS

Department of Mechanical Engineering, Texas A&M University, College Station, Texas 77843, USA

[a)]Author to whom correspondence should be addressed: nkanand@tamu.edu

ABSTRACT

Physics informed neural networks (PINNs) have demonstrated their effectiveness in solving partial differential equations (PDEs). By incorporating the governing equations and boundary conditions directly into the neural network architecture with the help of automatic differentiation, PINNs can approximate the solution of a system of PDEs with good accuracy. Here, an application of PINNs in solving three-dimensional (3D) Navier–Stokes equations for laminar, steady, and incompressible flow is presented. Notably, our approach involves deploying PINNs using feed-forward deep neural networks (DNNs) without depending on any simulation or experimental data. This investigation focuses on 3D square channel flow and 3D lid-driven cavity flow. For each case, one deep neural network was trained using only the governing equations and boundary conditions. Finally, the PINNs' results were compared with the computational fluid dynamics results. The goal was to assess the ability of PINNs (with DNN architectures) to predict the solution of Navier–Stokes equations in the 3D domain without any simulation or experimental data (unsupervised learning).

Computational fluid dynamics (CFD) approaches, such as finite volume method (FVM), finite element method (FEM), spectral method, and meshless methods have made tremendous strides in solving partial differential equations (PDEs). Albeit their effectiveness in solving numerous problems, these approaches have their limitations. For instance, mesh generation remains a time-consuming and subjective process and inverse problems require excessive computational power.[1] Neural network models, with the potential of solving PDEs,[2] offer fast evaluation and versatility, making them suitable PDE solvers for situations requiring numerous simulations, like inverse problems.[3,4] One such class, physics informed neural networks (PINNs),[5] has seen increasing popularity in recent years. PINNs employ automatic differentiation (AD)[6] in order to represent all the differential operators in the system of PDEs, which eliminates the need for discretization (meshing). In this approach, a comprehensive loss function is formulated that encompasses all the PDEs in the system, together with the initial condition and all the boundary conditions. Subsequently, the parameters of the neural network are optimized in order to minimize the loss.[5] Consequently, this method mitigates the aforementioned inconveniences associated with CFD techniques. Currently, researchers are actively exploring ways to enhance PINNs' performance. For example, Wight and Zhao[7] introduced the concept of time adaptive PINNs, which train multiple neural networks across multiple segments of the time domain of an unsteady problem; Mattey and Ghosh[8] introduced backward compatible PINNs (bcPINNs) demonstrating how training a single neural network for an unsteady case can be done efficiently in a segment-wise manner instead of training across the entire time domain in one go; McClenny and Braga-Neto[9] used a method called self-adaptive PINNs (SA-PINNs) that involves attaching a trainable weight to the loss associated with each point in the domain and maximizing the loss with respect to these weights, while minimizing the loss with respect to the neural network parameters at the same time; and in Causal PINNs,[10] a weighted loss function was defined where the effect of each loss term in the loss function is controlled by a Causality Parameter. When tackling equations with extremely nonlinear terms, such as the Allen Cahn equation and Cahn Hilliard equation, these versions of PINNs prevail over vanilla PINNs/baseline PINNs.[5] Obviously, there are other iterations of PINNs, such as Bayesian PINNs[11] that outperform vanilla PINNs in scenarios with substantial noisy data by avoiding overfitting, Fractional PINNs[12] that can solve problems of fractional calculus, coupled-automatic-numerical PINN (CAN-PINN)[13] and local structure approximation PINN (LSA-PINN)[14] which use numerical differentiation to achieve enhanced accuracy. Moreover, conservative PINN (cPINN),[15] extended PINN (XPINN),[16] and parallel PINNs[17] use domain decomposition approaches ensuring conservation along the interfaces, separable PINNs[18] combat the "curse of dimensionality" by executing per-axis operations rather than point-wise processing as in vanilla PINNs,

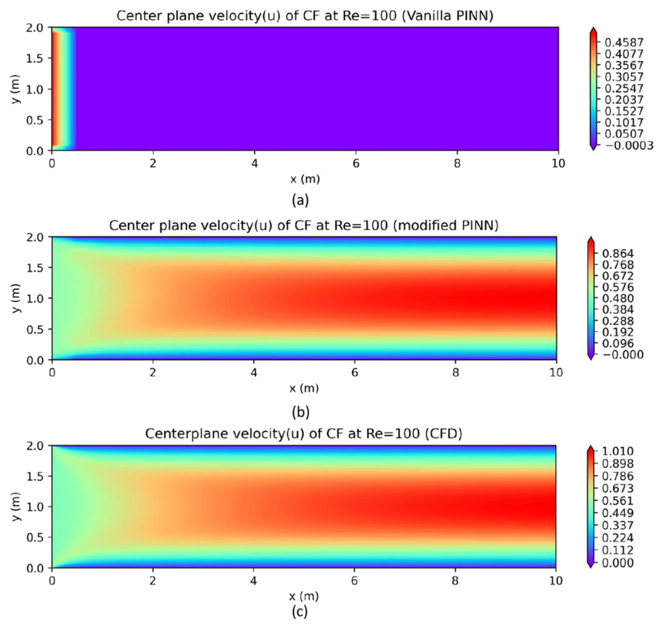
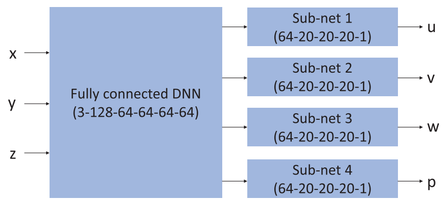


FIG. 1. Comparison of (a) Vanilla PINNs and (b) Vanilla PINNs with proposed network architecture and collocation points with (c) CFD results at Re = 100.



x
y
z
Fully connected DNN (3-128-64-64-64-64)
Sub-net 1 (64-20-20-20-1)
Sub-net 2 (64-20-20-20-1)
Sub-net 3 (64-20-20-20-1)
Sub-net 4 (64-20-20-20-1)
u
v
w
p

FIG. 2. Schematic of the considered neural network architecture.

PINNs with sine activation functions (sf-PINNs)[19] trains the neural network in sinusoidal spaces rescuing PINNs from falling into the local minimum trap.

Moreover, there have been numerous implications of PINNs in solving highly nonlinear PDE systems such as Navier–Stokes equations in 2D[20–27] as well as in 3D[28–32] domains. However, the potential of PINNs with feed-forward deep neural networks (DNN) for solving entirely unsupervised 3D fluid mechanics problems has not been explored much. To bridge this gap we attempted to solve the laminar, steady, and incompressible Navier–Stokes equations in 3D domains. The 3D square channel flow (using modified vanilla PINNs) for Reynolds numbers (Re) 50, 100, 200, and 400, as well as the 3D lid-driven cavity flow (using sf-PINNs) for Reynolds numbers 10, 50, and 100, were the specific cases considered in this work. Initially, we employed Vanilla PINNs with fully connected DNN architecture to model 3D Channel Flow at a Re = 100. Unfortunately, the outcome, depicted in Fig. 1(a), was not encouraging when compared against CFD results [Fig. 1(c)]. In response, we made two significant modifications: we altered the network architecture, as illustrated in Fig. 2, and we enhanced the concentration of collocation points in the vicinity of the inlet region. These adjustments ultimately enabled the algorithm to efficiently simulate the 3D case [Fig. 1(b)].

To construct the considered PINNs' architecture, the widely used open-source machine-learning framework PyTorch,[33] based on the Torch library,[34] was utilized. Two optimizers were used: (a) Stochastic gradient descent-based Adam[35] optimizer which uses Jacobian matrix (a matrix containing first-order derivatives of the loss function) and (b) quasi-Newton method-based L-BFGS[36] optimizer which uses the Hessian matrix (a matrix containing second-order derivatives of the loss function). L-BFGS optimizer is computationally intensive as it uses second-order derivatives, but it converges faster. On the other hand, the Adam optimizer converges slower, but uses less computational power. By adopting an initial implementation of the Adam optimizer and subsequently transitioning to the L-BFGS optimizer, we could effectively harness the advantages offered by both optimizers. Finally, we compared the results of our study to those obtained from the widely used CFD software, ANSYS Fluent.[37]

In what follows, we will discuss PINNs in a concise manner in regard to 3D, laminar, steady, and incompressible Navier–Stokes equations. The most important aspect of PINNs approach is the organization of the loss function. For this intent, we consider Fig. 3 where the schematic of the PINNs architecture is depicted. The composed loss function is as follows:

$$Loss = \lambda_{IC} MSE_{IC} + \lambda_{BC} MSE_{BC} + \lambda_{PDE} MSE_{PDE}, \qquad (1)$$

where MSE_{IC}, MSE_{BC}, and MSE_{PDE} are the mean squared errors with respect to the initial condition (for steady flow, we disregard this term), the boundary conditions [Eq. (2)] and the partial differential equations of the system [Eq. (3)], respectively. λ_{IC}, λ_{BC}, and λ_{PDE} are

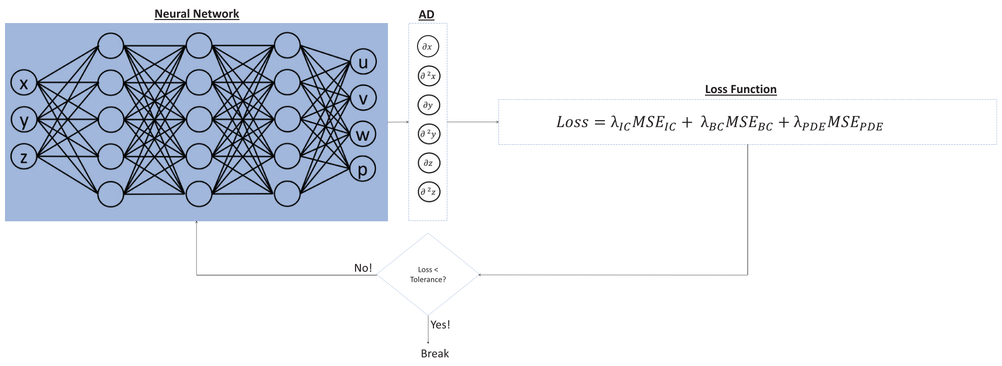


FIG. 3. Schematic of PINNs.

the weights associated with MSE_{IC}, MSE_{BC}, and MSE_{PDE}, respectively. These weights control the effect of loss terms in the loss functions (For this study, $\lambda_{IC} = \lambda_{BC} = \lambda_{PDE} = 1$). Let us, now, consider a feed-forward DNN (Fig. 2), $\mathcal{N}(\boldsymbol{x};\theta)$, parameterized by θ, where $\boldsymbol{x}$ represents the input vector (x, y, z) and θ represents the set of weights and biases of the neural network. This neural network will take the inputs x, y, z and will give the outputs u, v, w, p represented by $\mathcal{N}_u$, $\mathcal{N}_v$, $\mathcal{N}_w$, and $\mathcal{N}_p$, respectively. Now, we proceed to compose the loss function considering B as the generalized boundary condition,

$$MSE_{BC} = \frac{1}{N_{BC}}\sum_{i=1}^{N_{BC}}\left|\mathcal{N}(\boldsymbol{x}_{BC};\theta) - \mathcal{B}\right|^2, \tag{2}$$

$$MSE_{PDE} = MSE_{cont.} + MSE_{x-mom} + MSE_{y-mom} + MSE_{z-mom}, \tag{3}$$

$$MSE_{cont.} = \frac{1}{N_{PDE}}\sum_{i=1}^{N_{PDE}}\left|\frac{\partial\mathcal{N}_u}{\partial x} + \frac{\partial\mathcal{N}_v}{\partial y} + \frac{\partial\mathcal{N}_w}{\partial z}\right|^2, \tag{4}$$

$$MSE_{x-mom} = \frac{1}{N_{PDE}}\sum_{i=1}^{N_{PDE}}\bigg|\mathcal{N}_u\frac{\partial\mathcal{N}_u}{\partial x} + \mathcal{N}_v\frac{\partial\mathcal{N}_u}{\partial y} + \mathcal{N}_w\frac{\partial\mathcal{N}_u}{\partial z} + \frac{\partial\mathcal{N}_p}{\partial x} - \frac{1}{Re}\left(\frac{\partial^2\mathcal{N}_u}{\partial^2 x} + \frac{\partial^2\mathcal{N}_u}{\partial^2 y} + \frac{\partial^2\mathcal{N}_u}{\partial^2 z}\right)\bigg|^2, \tag{5}$$

$$MSE_{y-mom} = \frac{1}{N_{PDE}}\sum_{i=1}^{N_{PDE}}\bigg|\mathcal{N}_u\frac{\partial\mathcal{N}_v}{\partial x} + \mathcal{N}_v\frac{\partial\mathcal{N}_v}{\partial y} + \mathcal{N}_w\frac{\partial\mathcal{N}_v}{\partial z} + \frac{\partial\mathcal{N}_p}{\partial y} - \frac{1}{Re}\left(\frac{\partial^2\mathcal{N}_v}{\partial^2 x} + \frac{\partial^2\mathcal{N}_v}{\partial^2 y} + \frac{\partial^2\mathcal{N}_v}{\partial^2 z}\right)\bigg|^2, \tag{6}$$

$$MSE_{z-mom} = \frac{1}{N_{PDE}}\sum_{i=1}^{N_{PDE}}\bigg|\mathcal{N}_u\frac{\partial\mathcal{N}_w}{\partial x} + \mathcal{N}_v\frac{\partial\mathcal{N}_w}{\partial y} + \mathcal{N}_w\frac{\partial\mathcal{N}_w}{\partial z} + \frac{\partial\mathcal{N}_p}{\partial z} - \frac{1}{Re}\left(\frac{\partial^2\mathcal{N}_w}{\partial^2 x} + \frac{\partial^2\mathcal{N}_w}{\partial^2 y} + \frac{\partial^2\mathcal{N}_w}{\partial^2 z}\right)\bigg|^2. \tag{7}$$

Here, $MSE_{cont.}$ [Eq. (4)], MSE_{x-mom} [Eq. (5)], MSE_{y-mom} [Eq. (6)], and MSE_{z-mom} [Eq. (7)] indicate the losses associated with the continuity equation, x-momentum equation, y-momentum equation, and z-momentum equation, respectively. N_{BC} and N_{PDE} refer to the number of collocation points on the boundary and inside the domain, respectively, while $\boldsymbol{x}_{BC}$ represents the collocation points on the boundary. AD[6] has been used in order to represent and perform all the functions of the differential operators (such as $\partial/\partial x$, $\partial^2/\partial^2 x$, $\partial/\partial y$, $\partial^2/\partial^2 y$, $\partial/\partial z$, and $\partial^2/\partial^2 z$).

To demonstrate the application of PINNs method without using any experimental or simulation data, flow through a 3D square channel and flow in a 3D lid-driven cavity for various Reynolds numbers were considered. In both of these example problems, laminar, steady, and incompressible flow with constant thermo-physical properties were assumed. Figures 4 and 5 demonstrate the training history for all the channel flow cases and all the lid-driven cavity cases, respectively. After a few epochs (as the mini batching approach was not used, the epoch number was equal to the number of iterations), optimization using Adam did not show much progress, but applying LBFGS subsequently resulted in a significant decrease in loss. For comparison with the CFD results, the exact same cases were simulated using ANSYS Fluent, and the results of the grid independence study is given in Table I. Based on the grid independence study, 120 edge divisions were adopted in each direction for meshing.

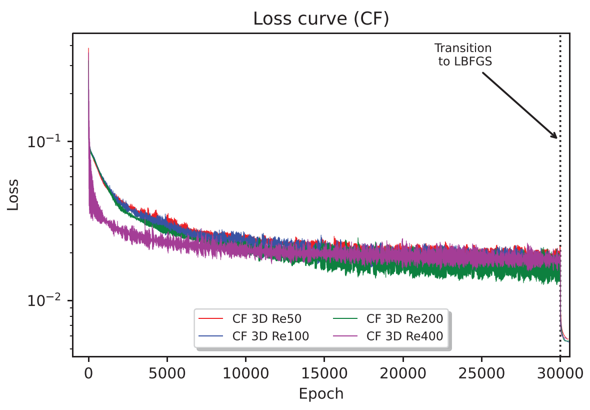


FIG. 4. Training history of PINNs for 3D channel flow.

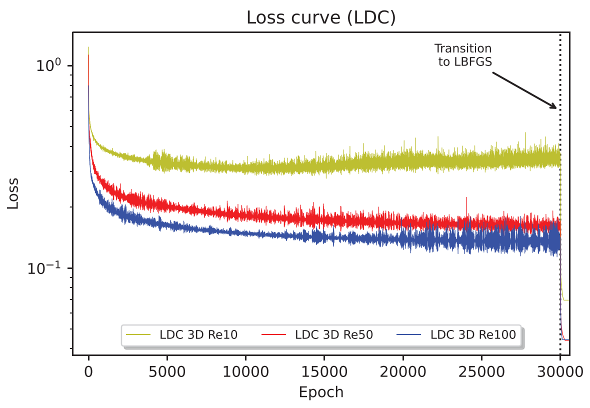


FIG. 5. Training history of PINNs for 3D lid-driven cavity flow.

TABLE I. Grid independence study of the CFD simulations.

Re	Edge divisions	Channel flow (CF) (u_{max} along central plane) (m/s)	Lid-driven cavity flow (LDC) (u_{min} along central plane) (m/s)
10	40	⋯	−0.226
10	60	⋯	−0.227
10	120	⋯	−0.227
50	40	1.040	−0.227
50	60	1.043	−0.228
50	120	1.044	−0.229
100	40	1.007	−0.227
100	60	1.010	−0.227
100	120	1.010	−0.227
200	40	0.916	⋯
200	60	0.918	⋯
200	120	0.918	⋯
400	40	0.803	⋯
400	60	0.803	⋯
400	120	0.803	⋯

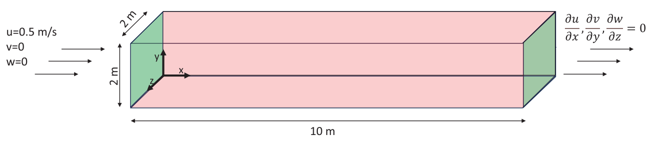

u=0.5 m/s
v=0
w=0
2 m
2 m
10 m
y
x
z
$\frac{\partial u}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial w}{\partial z} = 0$

FIG. 6. Schematic of channel flow.

The considered channel (Fig. 6) had a cross section of $2 \times 2\,\mathrm{m}^2$ and a length of 10 m. At channel entrance, uniform stream-wise velocity of 0.5 m/s was prescribed. At the channel walls, no-slip boundary conditions were imposed for velocities. At the channel exit, fully developed flow conditions were imposed. The modified vanilla PINNs algorithm was used to solve for the flow fields for $Re = 50, 100, 200$, and 400 cases. A feed-forward DNN architecture with four sub-networks (3-128-64-64-64-64-[-64-20-20-20-1,-64-20-20-20-1,-20-20-20-1,-64-20-20-20-1]) for four output variables (u, v, w, p) was used for each case. Weights and biases of the network were initialized randomly. The hyperbolic tan function was used as the activation function. For training, 187 500 collocation points were used inside the domain, 32 000 points were used near the inlet region, 20 000 points were used along inlet and outlet boundaries, and 5000 points were used along each wall. For optimization purposes, the Adam optimizer was used for 30 000 epochs, followed by the LBFGS optimizer for 600 epochs. Figures 7(a-1)–7(d-1) demonstrate the stream-wise velocity field solved by modified vanilla PINNs, while Figs. 7(a-2)–7(d-2) show the absolute error of stream-wise velocity when compared against CFD at the central plane. Similarly, Figs. 7(a-3)–7(d-3) depict stream-wise velocity field provided by vanilla PINNs, while Figs. 7(a-4)–7(d-4) exhibit the absolute error when compared against CFD at the channel exit. Furthermore, Figs. 7(a-5)–7(d-5) provide the comparison of velocity profiles at the vertical central line located at the channel exit. It is clear from these figures that the modified PINNs' solutions are in good agreement with CFD solutions for low Reynolds numbers (e.g., $Re = 50$, 100). However, the results deviate significantly for higher Reynolds numbers (e.g., $Re = 200$ and 400).

Next, a geometric cube (Fig. 8) of dimensions $1 \times 1 \times 1\,\mathrm{m}^3$ was considered for the lid-driven cavity case. In this scenario, the top lid was moving at a uniform stream-wise velocity of 1 m/s, and a no-slip boundary condition was enforced on all other walls. The flow was studied for $Re = 10, 50$, and 100. The sf-PINNs algorithm was used with a feed-forward DNN (3-128-128-128-128-128-[-20-20-20-1,-20-20-20-1,-20-20-20-1,-20-20-20-1]). Weights and biases of the network were initialized randomly. The first hidden layer used a sine activation function,[19] followed by layers with a hyperbolic tan activation function. 125 000 collocation points were used inside the domain, 22 500 points were used along the top moving boundary, and 2500 points were used along all the other walls. The Adam optimizer was used for 30 000 epochs and, subsequently, the optimization switched to LBFGS optimizer for 600 epochs. The results and validation are provided in Fig. 9. The central plane velocity fields denoted by u and v, solved using sf-PINNs, are given in Figs. 9(a-1)–9(c-1) and Figs. 9(a-3)–9(c-3), respectively. The absolute errors of u and v, when compared against CFD results, are depicted in Figs. 9(a-2)–9(c-2) and 9(a-4)–9(c-4),

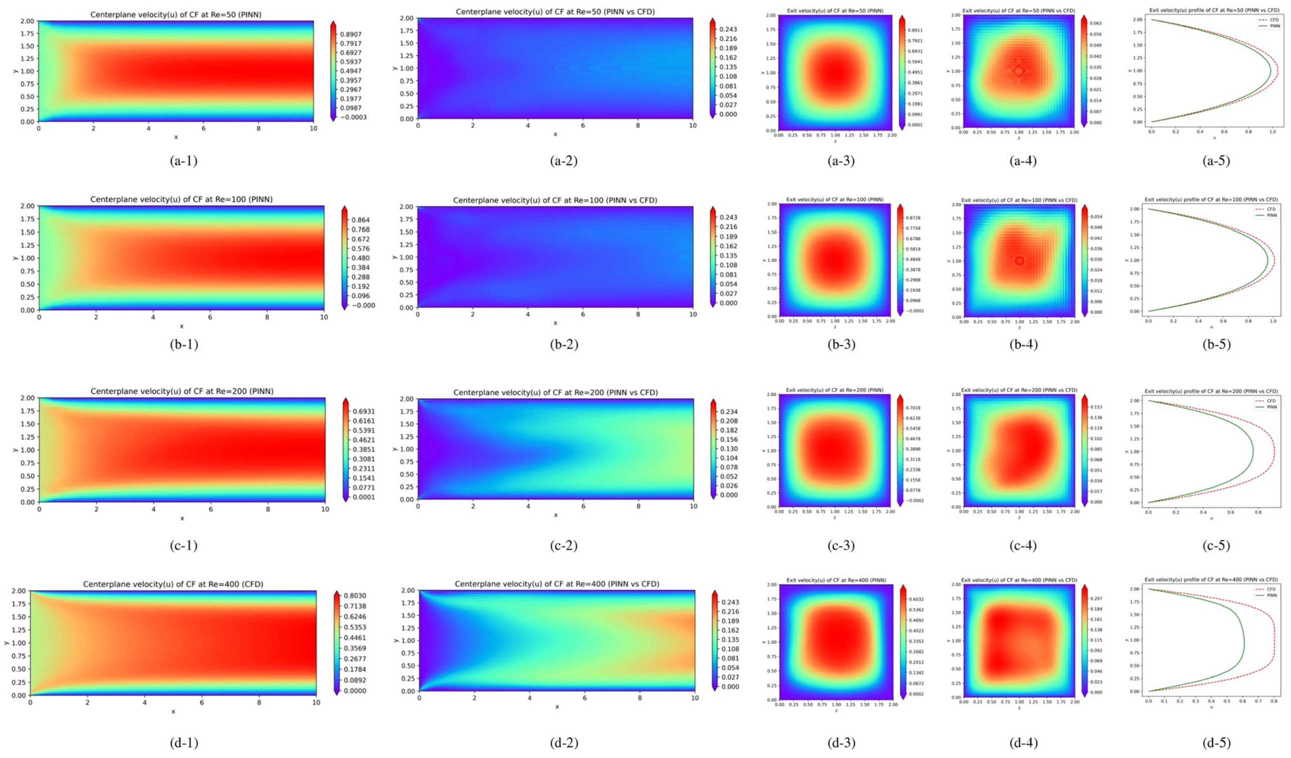


FIG. 7. 3D channel flow results from PINNs [(a-1)–(d-1) and (a-3)–(d-3)] and comparison with CFD [(a-2)–(d-2), (a-4)–(d-4), and (a-5)–(d-5)].

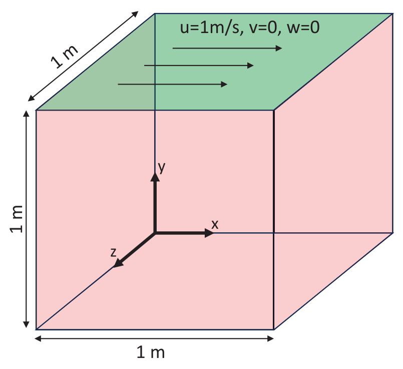


FIG. 8. Schematic of 3D lid-driven cavity flow.

respectively. Moreover, Figs. 9(a-5)–9(c-5) and Figs. 9(a-6)–9(c-6) hint at the comparisons of u and v profiles, respectively, along the vertical and horizontal central lines located at the central plane. From these figures, it is evident that for lower Reynolds numbers ($Re = 10$ and 50), the PINNs' solution compared well with the CFD solution. However, the findings for higher Reynolds number ($Re = 100$) showed noticeable discrepancies.

In summary, the authors have demonstrated the use of modified vanilla PINNs and sf-PINNs methods without using any simulation or experimental data in order to predict three-dimensional laminar, steady and incompressible flow fields in a square channel and in a lid-driven cavity. The PINNs' results were compared with CFD results. The PINNs' predictions corresponded well with the CFD predictions at low Reynolds numbers but deviated significantly at higher Reynolds numbers. The maximum relative errors of u along the vertical central line located at the channel exit were 5.30%, 5.22%, 16.83%, and 23.97% for $Re = 50$, 100, 200, and 400, respectively, and the maximum relative errors of v along the horizontal central line located at the central plane of the cavity were 9.74%, 15.10%, and 41.78% for $Re = 10$, 50, and 100, respectively. In future, advanced versions of PINNs (such as XPINN and SA-PINNs) will be employed to assess their performance in simulating 3D flows.

ACKNOWLEDGMENTS

We gratefully acknowledge the support from Texas A&M University High Performance Research Computing and the James J. Cain '51 Professor III funds.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Saykat Kumar Biswas: Conceptualization (equal); Methodology (equal); Validation (equal); Visualization (equal); Writing – original draft (equal). **N. K. Anand:** Conceptualization (equal); Funding acquisition (equal); Methodology (equal); Project administration (equal); Software (equal); Supervision (equal); Validation (equal); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

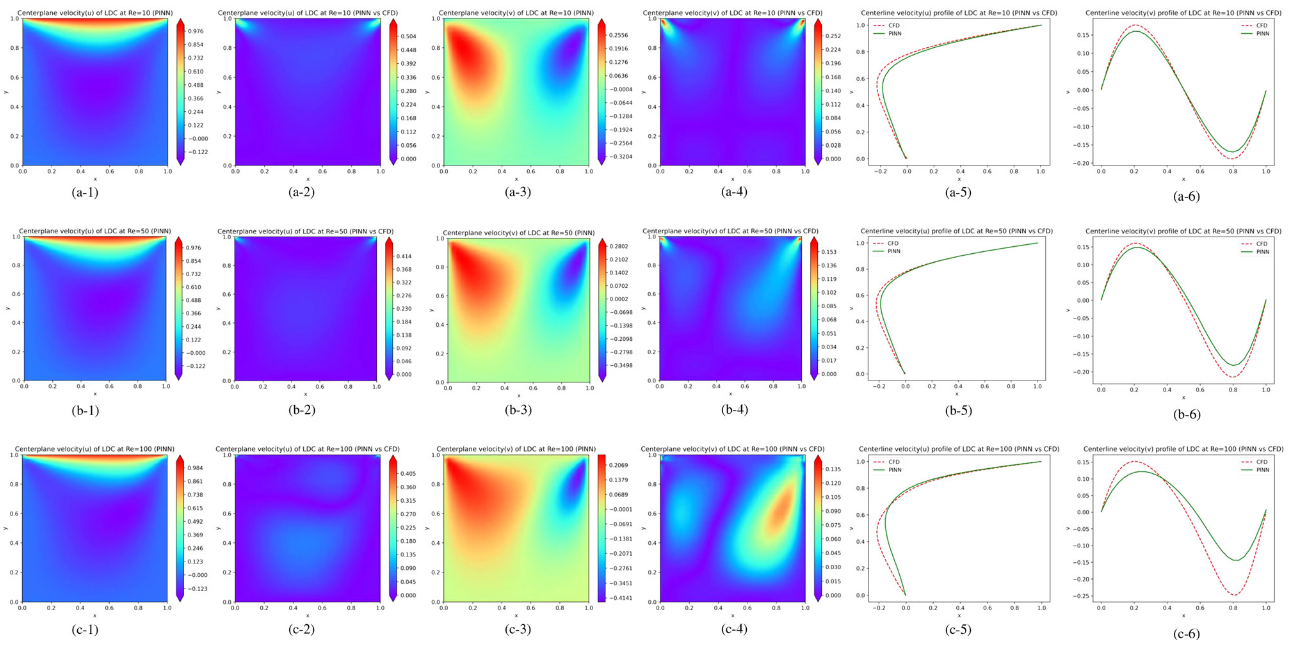


FIG. 9. 3D lid-driven cavity flow results from PINNs [(a-1)–(c-1) and (a-3)–(c-3)] and comparison with CFD [(a-2)–(c-2), (a-4)–(c-4), (a-5)–(c-5), and (a-6)–(c-6)].

REFERENCES

[1]S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNs) for fluid mechanics: A review," Acta Mech. Sin. **37**, 1727–1738 (2021).

[2]I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," IEEE Trans. Neural Networks **9**, 987–1000 (1998).

[3]Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," Comput. Methods Appl. Mech. Eng. **360**, 112789 (2020).

[4]H. Arbabi, J. E. Bunder, G. Samaey, A. J. Roberts, and I. G. Kevrekidis, "Linking machine learning with multiscale numerics: Data-driven discovery of homogenized equations," JOM **72**, 4444–4457 (2020).

[5]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. **378**, 686–707 (2019).

[6]A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS 2017 Autodiff Workshop* (OpenReview, 2017), available at https://openreview.net/pdf?id=BJJsrmfCZ.

[7]C. L. Wight and J. Zhao, "Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks," arXiv:2007.04542 (2020).

[8]R. Mattey and S. Ghosh, "A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations," Comput. Methods Appl. Mech. Eng. **390**, 114474 (2022).

[9]L. McClenny and U. Braga-Neto, "Self-adaptive physics-informed neural networks using a soft attention mechanism," arXiv:2009.04544 (2020).

[10]S. Wang, S. Sankaran, and P. Perdikaris, "Respecting causality is all you need for training physics-informed neural networks," arXiv:2203.07404 (2022).

[11]L. Yang, X. Meng, and G. E. Karniadakis, "B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data," J. Comput. Phys. **425**, 109913 (2021).

[12]G. Pang, L. Lu, and G. E. Karniadakis, "fPINNs: Fractional physics-informed neural networks," SIAM J. Sci. Comput. **41**, A2603–A2626 (2019).

[13]P.-H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, and Y.-S. Ong, "CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method," Comput. Methods Appl. Mech. Eng. **395**, 114909 (2022).

[14]J. C. Wong, P.-H. Chiu, C. Ooi, M. H. Dao, and Y.-S. Ong, "LSA-PINN: Linear boundary connectivity loss for solving PDEs on complex geometry," arXiv:2302.01518 (2023).

[15]A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis, "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems," Comput. Methods Appl. Mech. Eng. **365**, 113028 (2020).

[16]A. D. Jagtap and G. E. Karniadakis, "Extended physics-informed neural networks (XPINNs): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations," in AAAI Spring Symposium: MLPS (2021), Vol. 10.

[17]K. Shukla, A. D. Jagtap, and G. E. Karniadakis, "Parallel physics-informed neural networks via domain decomposition," J. Comput. Phys. **447**, 110683 (2021).

[18]J. Cho, S. Nam, H. Yang, S.-B. Yun, Y. Hong, and E. Park, "Separable PINN: Mitigating the curse of dimensionality in physics-informed neural networks," arXiv:2211.08761 (2022).

[19]J. C. Wong, C. Ooi, A. Gupta, and Y.-S. Ong, "Learning in sinusoidal spaces with physics-informed neural networks," in IEEE Transactions on Artificial Intelligence (2022).

[20]J. Oldenburg, F. Borowski, A. Öner, K.-P. Schmitz, and M. Stiehm, "Geometry aware physics informed neural network surrogate for solving Navier–Stokes equation (GAPINN)," Adv. Modeling Simul. Eng. Sci. **9**, 8 (2022).

[21]C. Rao, H. Sun, and Y. Liu, "Physics-informed deep learning for incompressible laminar flows," Theor. Appl. Mech. Lett. **10**, 207–212 (2020).

[22]H. Ma, Y. Zhang, N. Thuerey, X. Hu, and O. J. Haidn, "Physics-driven learning of the steady Navier-Stokes equations using deep convolutional neural networks," arXiv:2106.09301 (2021).

[23]R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (ACM, 2020), pp. 1457–1466.

[24]H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, "Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations," Phys. Fluids **34**, 075117 (2022).

[25]P. P. Mehta, G. Pang, F. Song, and G. E. Karniadakis, "Discovering a universal variable-order fractional model for turbulent Couette flow using a physics-informed neural network," Fractional Calculus Appl. Anal. **22**, 1675–1688 (2019).

[26]V. Kag, K. Seshasayanan, and V. Gopinath, "Physics-informed data based neural networks for two-dimensional turbulence," Phys. Fluids **34**, 055130 (2022).

[27]X. Yang, S. Zafar, J.-X. Wang, and H. Xiao, "Predictive large-eddy-simulation wall modeling via physics-informed neural networks," Phys. Rev. Fluids **4**, 034602 (2019).

[28]D. Lucor, A. Agrawal, and A. Sergent, "Simple computational strategies for more effective physics-informed neural networks modeling of turbulent natural convection," J. Comput. Phys. **456**, 111022 (2022).

[29]S. Cai, Z. Wang, F. Fuest, Y. J. Jeon, C. Gray, and G. E. Karniadakis, "Flow over an espresso cup: Inferring 3-D velocity and pressure fields from tomographic background oriented Schlieren via physics-informed neural networks," J. Fluid Mech. **915**, A102 (2021).

[30]H. Wang, Y. Liu, and S. Wang, "Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network," Phys. Fluids **34**, 017116 (2022).

[31]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. Comput. Phys. **426**, 109951 (2021).

[32]N. Wandel, M. Weinmann, and R. Klein, "Teaching the incompressible Navier–Stokes equations to fast neural surrogate models in three dimensions," Phys. Fluids **33**, 047117 (2021).

[33]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems (2019), Vol. 32.

[34]R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: A modular machine learning software library," Report No. IDIAP-RR 02-46, 2002.

[35]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980 (2014).

[36]D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," Math. Program. **45**, 503–528 (1989).

[37]J. E. Matsson, *An Introduction to Ansys Fluent 2023* (SDC Publications, 2023).