Application of Physics Informed Neural Network for an Unsteady Rayleigh-Bénard Flow

Sunil V. Bharadwaj
ABB Corporate Research Centre
Bengaluru, Karnataka, India.
sunil.bharadwaj@in.abb.com

Chandrika K R
ABB Corporate Research Centre
Bengaluru, Karnataka, India.
chandrika.k-r@in.abb.com

Divyasheel Sharma
ABB Corporate Research Centre
Bengaluru, Karnataka, India.
divya.sheel@in.abb.com

Subhashish Dasgupta
ABB Corporate Research Centre
Bengaluru, Karnataka, India.
subhashish.dasgupta@in.abb.com

Corresponding author: sunil.bharadwaj@in.abb.com

Abstract

Scientific machine learning, particularly in fluid dynamics, is advancing through the adoption of promising physics-informed neural networks (PINNs). This study demonstrates the applicability of PINNs in an unsteady flow configuration where boundary conditions are not explicitly defined during training. In real-world applications, fluids and convective heat transfer cannot be measured with ease – resulting in sparse data. However, PINNs overcome this lack of data by embedding domain knowledge directly into the training process through partial differential equations (PDEs) as additional loss terms. We apply a PINN-based model to the 2D Rayleigh-Bénard (RB) convective flow and show its effectiveness compared to model-free simulations, highlighting its utility in control and optimisation applications.

1 Introduction

Physics-Informed Neural Networks (PINNs) have recently gained significant attention, particularly for applications in fluid dynamics (e.g., see Cai et al. [2021]). PINNs overcome the limitations of using solely data-driven approaches by incorporating domain knowledge through a weighted loss function derived from partial differential equations (PDEs) or grid-less simulations.This allows PINNs to work effectively with sparse data, as shown by Clark Di Leoni et al. [2023], who used PINN to reconstruct velocity fields usingonly temperature measurements.

1st Conference on Applied AI and Scientific Machine Learning (CASML 2024).

A significant drawback of using only data-driven models is the lack of measurable data in fluid flow or heat transfer systems, where sparse sensors provide minimal data. Take, for example, a simple case of forced convection in heat fins. Measurements are restricted to a few thermal sensors and much more rarely, anemometers, which measure the speed of the flow. These point sensors offer minimal data compared to the degrees of freedom in these flows. One may need more sophisticated methods to learn from such sparse data. PINNs bridge this gap by embedding physical laws into the network, allowing the generation of dense approximations from sparse data. Although computational fluid dynamics (CFD) simulations offer accurate modeling, they are computationally expensive and not suitable for real-time applications.

PINNs can address these limitations by learning from sparse measurements and CFD simulations, enabling faster approximations. However, a key challenge for PINNs remains in reconstructing boundary conditions from sparse data—an "inverse" problem. While some researchers (e.g., Jagtap et al. [2022], Raissi et al. [2020]) have explored this, its general applicability is still debated (Chuang and Barba [2023] and McGreivy and Hakim [2024]).

In this study, we apply PINNs to reconstruct RB flow without explicit boundary conditions and compare the results with model-less neural network simulations to demonstrate the value of embedding domain knowledge.

1.1 Methodology

PINN in essence, is a type of domain-based machine learning algorithm. Here, the "domain knowledge" is provided as a set of partial differential equations (PDEs). The solution to these PDEs either by numerical simulation or exact solutions act as an additional learning parameter for the neural network training (Raissi et al. [2019]). The numerical simulation route is often the only viable option in practical settings, as most real-world PDEs do not have closed-form solutions.

There are many types of neural networks including fully connected and recurrent networks. In this study we have used fully connected networks. While alternative architectures, such as convolutional networks, could potentially capture spatial dependencies more efficiently, fully connected networks offer greater flexibility in handling diverse types of input data and boundary conditions, making them a suitable choice for this type of problem.

Various software frameworks support the implementation of PINNs, including Nvidia Modulus Hennigh et al. [2020] and DeepXDE Lu et al. [2021b]. We have used DeepXDE (Lu et al. [2021b])because of its ease of use, granular control, versatility and support for solving forward and inverse PDE problems that make it suitable for performing a proof-of-concept study.

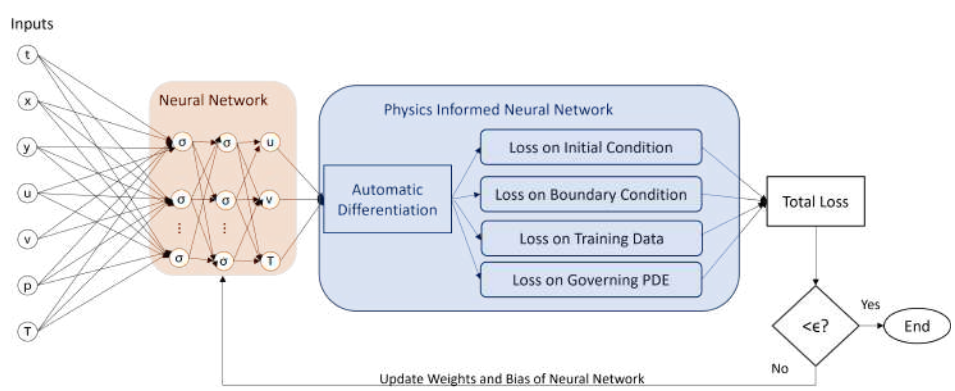


Figure 1: The schematic of PINN depicting the layers and the loss function.

Consider a generalised unsteady PDE system represented by:

$$f(x, t, U, \partial_x U, \partial_t U..., C) = 0, \quad x \in \Omega, t \in [0, T] \tag{1}$$

coupled with initial and boundary conditions

$$U(x, t = 0) = g_0(x) U(x, t) = g(x, t) \quad x \in \partial\Omega, \tag{2}$$

respectively. U is the output variable, x is the spatial coordinate and t is the time. The function f is a combination of these variables and the derivatives of U in the domain Ω - note that $\partial\Omega$ represents the boundary of the domain. These PDEs are solved at each points corresponding to the training data, enabling the network to capture the underlying physics.

The architecture of the neural network with hidden layers is represented by:

$$N^0 = (x,t)N^j = \sigma(W^j N^{j-1} + b^j), \quad 1 \leq j \leq (L-1)N^L = U = W^{L-1} + b^L \tag{3}$$

Here, W^i is the weight of the ith layer of the neural network. Similarly, b is bias of each layer. N is the neural network layers with L layers. For a detailed mechanism of this implementation of PINN, refer to Lu et al. [2021a].

The total loss function used to train the neural network is then obtained by adding the individual losses from data training, PDE, initial conditions and boundary conditions.

$$L_{total} = w_1 L_{data} + w_2 L_{PDE} + w_3 L_{IC} + w_4 L_{BC} \tag{4}$$

where L is the loss function and w_i are the respective weights. Note that IC and BC stand for initial conditions and boundary conditions, respectively. The weights are typically determined through empirical methods or cross-validation, ensuring that the model places appropriate emphasis on the physical laws and data-driven aspects of the problem.

There are two methods to applying PINN for any fluid flow - one with soft boundary conditions constraints and other with hard boundary condition constraints. Soft boundary conditions offer greater flexibility by embedding the constraints into the loss function, while hard constraints strictly enforce the conditions at each step of the training process. We employed the soft boundary conditions constraint method due to its unique flexibility and to demonstrate the "inverse" capabilities of PINNs.

2 Rayleigh-Bénard flow

Rayleigh-Bénard (RB) flow is a canonical convective case where a fluid in a closed channel experiences a temperature gradient. The bottom plate is heated whilst the top plate is cooled, and the heavier, cooler fluid sinks whilst the lighter, hotter fluid rises, forming convective rolls that depend on the channel's aspect ratio. RB flow is important for modeling natural and industrial processes like the ocean conveyor belt where temperature differences cause the movement of ocean water (see Bodenschatz et al. [2000] for details).

The equations defining the velocity in an RB flow are:

$$\frac{\partial U}{\partial t} + U.\nabla U = -\frac{1}{\rho}\nabla P + \mu\nabla^2(U) + g. \tag{5}$$

The continuity equation for an incompressible flow then dictates the mass-balance:

$$\nabla.U = 0 \tag{6}$$

The temperature equation is given by:

$$\frac{\partial T}{\partial t} + U.\nabla T = \alpha\nabla^2 T. \tag{7}$$

Note that U is the velocity, ρ is the density, P is the pressure, T is the temperature and g is the acceleration due to gravity. α is the thermal diffusivity.

Rayleigh-benard flow is typically associated with Rayleigh number:

$$Ra = \frac{\rho\beta\Delta T l^3 g}{\mu\alpha} \tag{8}$$

where β is the thermal expansion coefficient, l is the typical length scale, which in this case is the gap distance between the hot and the cold plates. Flow configurations are associated with a critical Ra number below which conduction dominates the heat transfer but above the critical Ra number, convection dominates. The critical number for RB flow is 1708. Typically in nature the Ra number is greater than 10^6, which suggests that convection dominates over conduction when involving fluid flows.

PINNs are used to model RB flow without explicit boundary conditions losses, showing their ability to handle complex domains where boundaries are hard to express.

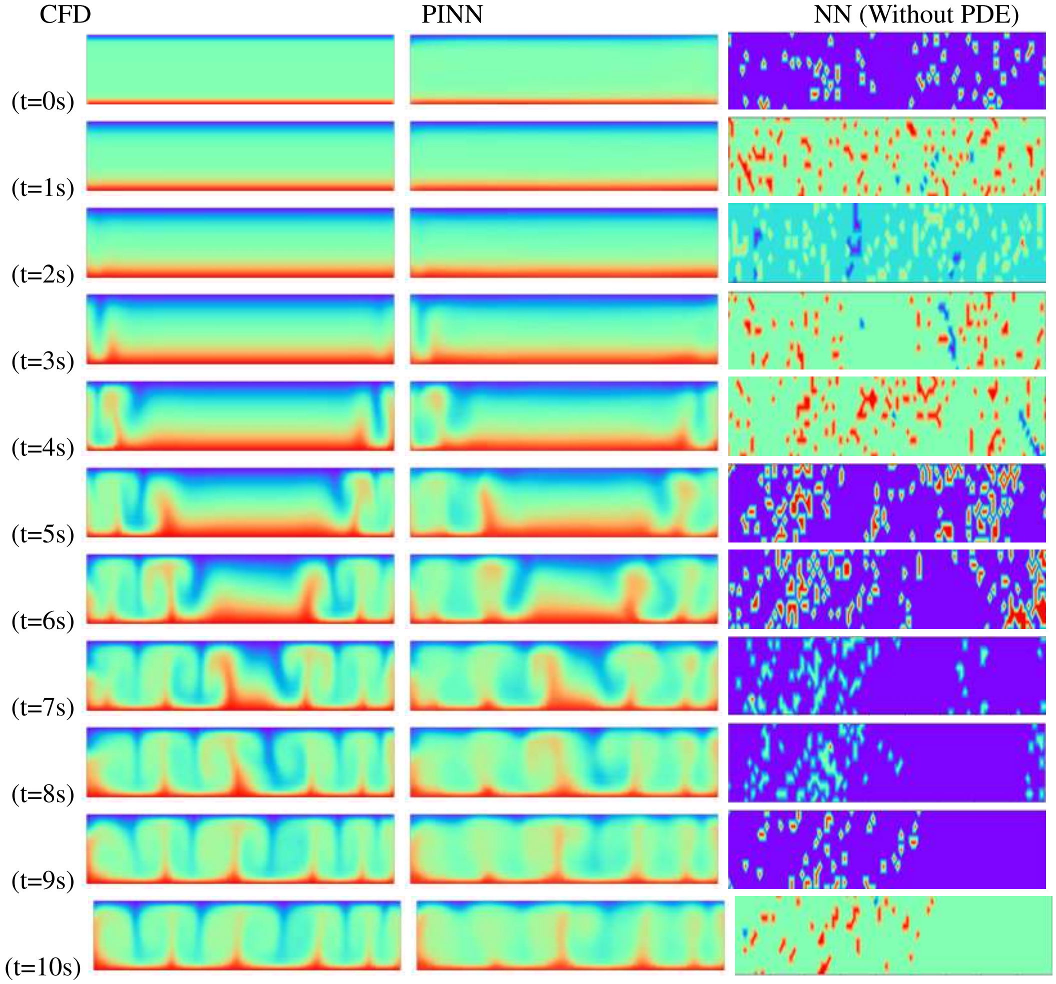
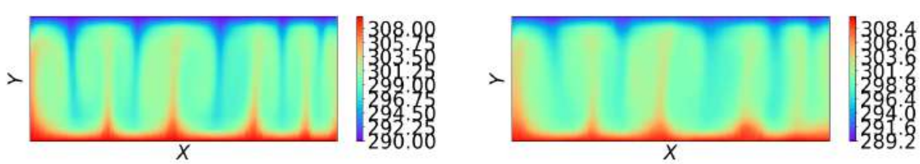


Figure 2: A comparison of temperature field as output from PINN with 7000 points, with CFD simulation and neural network (without PDE) of the Rayleigh-Benard flow. In the left are the CFD simulations whereas the middle panels are the PINN reconstructions and right are the Neural Networks (without PDE). For brevity, in the figures we have chosen to not show any labels.



)

Figure 3: A comparison plot of CFD and PINN simulations for the RB flow with 4000 training points corresponding to the temperature field. The labels and contour limits also correspond to Fig. 2.

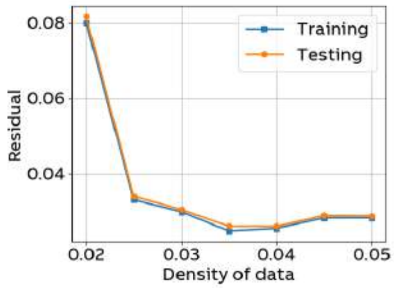


Figure 4: A plot of residual error between CFD and PINN simulations as a function of sparseness

2.1 Simulations

For the CFD simulations, the open-source package OpenFOAM (version 8) was used (Weller et al. [1998]). OpenFOAM is a CFD software that allows typical simulations and is widely used in industry and academia. In this simulation, a 2-D grid was used with temperatures of 290K at the top and 310K at the bottom. A total of 2000 points/ time step was used for our simulations. The aspect ration of geometry was 9:1. Periodic boundary conditions were applied in the lateral direction (Y-direction).

2.2 Results

In Fig. 2, we present the results where CFD (100 time steps, 10 seconds) and $\Delta T = 20K$ between the top and the bottom plate is used. In comparison, the PINN was input with PDE as a loss function in training with as few as 7000 points - these were randomly selected points across the entire simulation consisting of 200,000 points. The PINN is able to learn the dynamics of the system as evidenced. The L_2 error for different numbers of training data is shown in Fig 4. (Note that in Fig. 4, the *density of data* is defined as the ratio of training data and total data available for training). The residual reduces as you increase the density of data and stabilises at around 0.03 (that is 3 percent of the data). The CFD simulations show a typical RB flow where the flow exhibits convective rolls due to the thermal instability as shown in the velocity contours. These convective rolls form the basis of RB flow and the number of such convective rolls is a function of the aspect ratio in x and y directions. The aspect ratio in this case was 9:1. Refer to Fig. 3 for the colour map and the respective labels. The temperature field from CFD and PINN model suggests that PINN is able to learn to capture the convective rolls of the Rayleigh-Benard flow effectively. The unsteady nature is captured in the neural network model. However, it should be noted that PINN does not implicitly extrapolate the data outside the time-domain of the flow. To extrapolate the data outside the time range of the CFD simulation, one may need to modify the algorithm to account for it.
We also conducted experiments that kept the physics loss at none, which means the neural network was trained without PDE. The last column of results in Fig. 2 represents the prediction from the neural network where it was trained without PDE with 7000 samples. We also experimented by increasing the samples from 7000 to 200000 samples. Fig. 5 represents the temperature field learned by neural network without PDE with 200000 samples. However, we found that the neural network could not model the physical system even with large data samples. This might be due to the fact that when the physics loss was set to zero, the hyperparameters—originally tuned for PINN training—were not re-optimized for non-physics-based training. A fairer comparison would require optimizing the neural network specifically for data-driven training, which we could explore in the future work.

(a) (b)

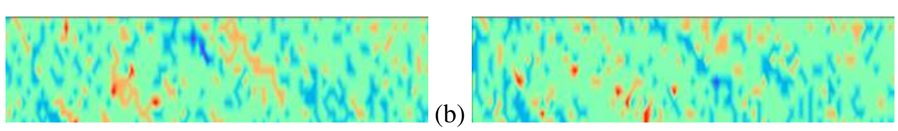


Figure 5: Temperature Field predicted by neural network without PDE trained on 200000 samples for at time= (a) 9 and (b) 10 seconds.

3 Discussion

PINN is a domain-based machine learning method with applications in many physical systems. In this study, we demonstrated that PINN can extract dense information from sparse data in fluid flows and convective heat transfer, where the governing equations are non-linear and lack analytical solutions. This paves way for complex applications including heat management and gas leak detection, where traditional measurements fall short.

Using Rayleigh-Bénard flow as a test case, we validated the method by reproducing unsteady convective rolls with soft boundary constraints, showing PINN's potential for laminar flows even with unknown boundary information. While this study highlights PINN's effectiveness in heat transfer modeling, open questions remain, particularly regarding the challenges of turbulent flow. Given the chaotic nature of turbulence, future work could explore incorporating turbulence models into the training loss functions to extend PINN's real-world applicability.

References

Eberhard Bodenschatz, Werner Pesch, and Guenter Ahlers. Recent developments in rayleigh-bénard convection. *Annual Review of Fluid Mechanics*, 32(1):709–778, 2000. doi: 10.1146/annurev.fluid.32.1.709. URL `https://doi.org/10.1146/annurev.fluid.32.1.709`.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

Pi-Yueh Chuang and Lorena A Barba. Predictive limitations of physics-informed neural networks in vortex shedding. *arXiv preprint arXiv:2306.00230*, 2023.

Patricio Clark Di Leoni, Lokahith Agasthya, Michele Buzzicotti, and Luca Biferale. Reconstructing rayleigh–bénard flows out of temperature-only measurements using physics-informed neural networks. *The European Physical Journal E*, 46(3):16, 2023.

Oliver Hennigh, Susheela Narasimhan, Mohammad Amin Nabian, Akshay Subramaniam, Kaustubh Tangsali, Max Rietmann, Jose del Aguila Ferrandis, Wonmin Byeon, Zhiwei Fang, and Sanjay Choudhry. Nvidia simnet: an ai-accelerated multi-physics simulation framework, 2020.

Ameya D. Jagtap, Zhiping Mao, Nikolaus Adams, and George Em Karniadakis. Physics-informed neural networks for inverse problems in supersonic flows. *Journal of Computational Physics*, 466:111402, 2022. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2022.111402. URL `https://www.sciencedirect.com/science/article/pii/S0021999122004648`.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021a. doi: 10.1137/19M1274067. URL `https://doi.org/10.1137/19M1274067`.

Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021b. doi: 10.1137/19M1274067.

Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature Machine Intelligence*, pages 1–14, 2024.

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. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL `https://www.sciencedirect.com/science/article/pii/S0021999118307125`.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computer in Physics*, 12(6):620–631, 11 1998. ISSN 0894-1866. doi: 10.1063/1.168744. URL `https://doi.org/10.1063/1.168744`.