Modeling Pollution Spread with Obstructions using Physics-Informed Neural Networks

Yash Ranjith
Westmont High School
Campbell, California, USA
yash.ranjith@gmail.com

Abstract

Pollution modeling plays a crucial role in combating climate change and protecting public health. Scientists rely on traditional numerical methods, such as finite difference method and computational fluid dynamics, for accurate pollution simulations; however, they are resource-intensive and time-consuming, often taking days to execute. Physics-Informed Neural Networks (PINNs) present a promising alternative, capable of significantly improving speed while maintaining high accuracy. In our research, we developed a PINN to model pollution spread under laminar flow conditions in a two-dimensional environment with obstructions. Our model integrates the Navier-Stokes and advection-diffusion partial differential equations (PDEs) and enforces no-slip and zero-flux Neumann boundaries at obstruction surfaces and simulation edges. We used a hybrid learning approach, generating a dataset of 16.1 million colocation points for supervised learning and embedding physical laws into the loss function for unsupervised learning. In our experiments, the PINN was over 2520 times faster than the traditional numerical solvers for 5000 timesteps, delivering results in under 2190 milliseconds and achieving a mean squared error below 3E-5. Our findings demonstrate that PINNs not only offer a drastic reduction in computational time but also scale favorably with both the time domain and spatial resolution, making them a viable solution for real-time pollution monitoring and emergency response planning. Future work will focus on extending our model to dynamic obstructions and arbitrary grid geometries.

1 Introduction

Pollution modeling enhances our ability to combat climate change. Scientists rely on pollution models to provide critical data for emissions management [1], climate predictions [2], health protection [3], and sustainable energy planning. Traditionally, pollution simulations use numerical methods, such as the finite element method [4], finite difference method, finite volume method, smooth particle hydrodynamics [5], and computational fluid dynamics [6] to solve PDEs, which are the types of equations seen in physical laws. While these are tried and tested methods, one of the biggest downsides is that they are known to be slow, taking days to run [7]. However, with recent advances in Machine Learning, researchers potentially have found a faster alternative to traditional numerical methods [8, 9, 10]. In the past several years, researchers have successfully trained PINNs to solve PDEs [11]. The reason PINNs can solve PDEs is that, given a big enough neural network, it is theoretically capable of approximating any continuous function, no matter how complex it may be [12]. Thus, neural networks are a promising technology that can, one day, render traditional numerical methods obsolete [13].

In recent years, there has been a growing number of publications on the use of PINNs to model pollution spread under various conditions. AI researchers at the California Institute of Technology

Tackling Climate Change with Machine Learning: workshop at NeurIPS 2024.

were able to teach a neural network to solve the Navier-Stokes equations within an acceptable error range in comparison to traditional numerical methods. However, this research was limited to solving the Navier-Stokes equations on a continuous grid, without incorporating obstructions [14]. In a recent paper [15], researchers employed PINNs to model free-surface laminar flows described by the shallow water equations. While they used soft initial conditions, the study faced limitations due to the lack of obstructions and the reliance on open boundaries across the domain. In [16], the authors introduce a method to improve the accuracy of solving nonlinear partial differential equations by adaptively focusing on difficult regions in the solution. While this research used soft initial conditions, they did not include any obstructions in the flow domain.

Our review of recent publications on PINNs identified a focus on grids with open boundaries and without obstructions. To enable more realistic pollution simulations that account for the spread around obstacles such as buildings and other structures, our research seeks to build on these works by introducing obstructions and incorporating soft initial conditions. Specifically, we will apply no-slip and zero-flux conditions to both the obstruction surfaces and the edges of the simulation environment. Additionally, we will integrate soft initial conditions by allowing the neural network to receive the coordinates of the initial pollution source location as input. This approach will enable us to develop a dynamic model capable of simulating pollution spread from any given source location.

2 Methods

2.1 Governing Equations and Simulation Environment for Pollution Spread

The spread of pollution is governed by the Navier-Stokes and advection-diffusion PDEs as shown in (1) - (4). Numerical simulations for pollution spread consist of two general steps: solving for fluid flow using (1) - (3) and advecting the pollutant particles using (4). We model pollution in a 2D 100 x 100 Cartesian grid for timesteps in range $t \in [0, 5000]$, assuming laminar flow throughout. Figure 1, shows the simulation environment with obstructions in white denoted by the Ω symbol in the left image. We apply no-slip and zero-flux Neumann boundary conditions [17] along obstruction surfaces and the edges of the simulation environment to ensure no pollutant flux occurs across boundaries. In (5) - (8), n denotes the normal to the boundary. The initial pollutant source location and concentration, shown in red, are programmed into our spatial grid using an exponential function.

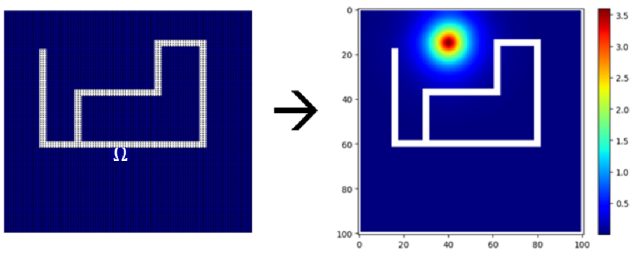

Figure 1: Simulation environment with obstructions and the initial pollution source

2.2 Solving for Pollution Spread Numerically

Our PINN training consists of both supervised and unsupervised learning. For supervised learning, we solve for pollution spread numerically to create the training dataset. We use the Navier-Stokes PDEs for 2D cavity flow to calculate all the velocity components at each timestep. The momentum equations for the 'x' direction, as shown in (1), and the 'y' direction, as shown in (2), along with the Poisson equation for pressure, as shown in (3), are solved using the finite difference method (FDM) to determine the spatially dependent velocity values throughout the grid. We first discretize these equations to obtain the momentum and pressure equations in terms of their surrounding locations and then rearrange them to arrive at the momentum equation in the u direction, as shown in (9), the momentum equation in the v direction, as shown in (10), and the pressure Poisson equation, as shown in (11). We then iteratively update the velocity components (u and v) and pressure (p) for every lattice point within the domain. Once the velocity profile is established, we pass it to the two-dimensional advection-diffusion equation (4) to simulate pollutant transport. We use FDM to solve the advection-diffusion equation, employing forward difference for time derivatives and central

difference for spatial derivatives. The discretized terms are substituted into (4) and rearranged to arrive at (12), which is used to compute the pollutant spread ϕ at the next timestep.

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

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

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -\rho\left(\frac{\partial u}{\partial x}\frac{\partial u}{\partial x} + 2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x} + \frac{\partial v}{\partial y}\frac{\partial v}{\partial y}\right) \tag{3}$$

$$\frac{\partial \phi}{\partial t} = \mu\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right) - u\frac{\partial \phi}{\partial x} - v\frac{\partial \phi}{\partial y} \tag{4}$$

2.3 Data Collection for Supervised Learning

We run each simulation for timesteps in range $t \in$ [0, 5000] with parameters: Δt = 0.001, Δx = 0.1, Δy = 0.1, ρ = 1.0, μ = 1.0, ν = 1.0. The pollution source location is allowed to vary between (10, 10) and (90, 90) outside of the obstruction. We collect random samples across multiple simulations to create a colocation dataset. We gather approximately 16.1 million colocation points to train the PINN. Additionally, it is important that the PINN receives specialized training to accurately learn the initial state. To help the model learn the initial conditions of the simulation, we create a separate dataset consisting of colocation points exclusively from timestep 0. Each colocation point includes five inputs: t (timestep), x, y (position on the simulation grid), cx, and cy (coordinates of the pollution source), and four outputs: ϕ (the concentration of the pollutant at the location specified by the input parameters), *u* (the velocity in the x-direction), *v* (the velocity in the y-direction), and *p* (the pressure at the location). The top row of Figure 2 illustrates the ground truth for a pollution source located at coordinates $cx = 20$, $cy = 20$ for timesteps 2000, 3000, 4000, and 5000.

2.4 Supervised, Unsupervised and Hybrid Learning

The objective of supervised training is to minimize the mean squared error (MSE). At a colocation point, if we assume the PINN's prediction to be y_{pred} and the actual pollution to be y_{act}, then the supervised loss function can be described as shown in (15). The same equation is also applied to the initial condition colocation dataset to create a loss for the initial condition as shown in (16).

To enhance the PINN's accuracy through unsupervised learning, we embed the governing equations directly into the loss function. This ensures that the model adheres to the physical laws governing fluid and pollutant dynamics. The unsupervised loss is derived from the Navier-Stokes equations for laminar flow and the advection-diffusion equation for pollutant transport. We introduce the loss terms in (17)-(24), and define the total unsupervised loss, $\mathcal{L}_U$, as their weighted sum, as shown in (25).

We adopt a hybrid learning approach that combines supervised and unsupervised learning, allowing the PINN to learn from both the dataset of colocation points and the governing physical equations. The hybrid loss function $\mathcal{L}_H$, as shown in (26), integrates the supervised losses $\mathcal{L}_S$ and $\mathcal{L}_I$ with the unsupervised loss $\mathcal{L}_U$. Initially, we set the weights of the unsupervised loss to a low value to prioritize supervised learning, gradually increasing them as training progresses. This strategy improves the model's generalization, ensuring it adheres to physical laws even when supervised training data is sparse.

2.5 PINN Architecture and Training Procedure

We split our colocation dataset into training and testing sets to build and validate the PINN. We train the network using the training set to predict pollutant spread for any combination of t, cx, cy, x, and y, and validate its performance using the testing set. Our network architecture includes an input layer with 5 neurons, four hidden layers with 150 neurons each, and an output layer with 4 neurons. We use the hyperbolic tangent as the activation function. We scale input features to [-1, 1] and unscale the output values after predictions using (13) and (14). We train the network with the Adam optimizer, starting at a learning rate of 0.001 and gradually decay it to 0.0005. The MSE serves as our loss function, and we train the PINN for 52 epochs with a batch size of 128. We finalized this architecture after several rounds of experimentation, using recommendations from the Auto-PINN paper [18].

3 Results

The goal of a physics-informed neural network is to replicate the results of traditional numerical methods and to do it much faster. So, the performance of a PINN can be measured using two metrics: the mean squared error (MSE) when compared to the traditional numerical method and the speed advantage of the PINN over the traditional numerical method. The PINN results for $cx = 20$, $cy = 20$, and timesteps 2000, 3000, 4000, and 5000 are shown in the bottom row of Figure 2. The PINN successfully replicated the ground truth results from the traditional solver, as shown in the top row of Figure 2 within an MSE of 0.00003.

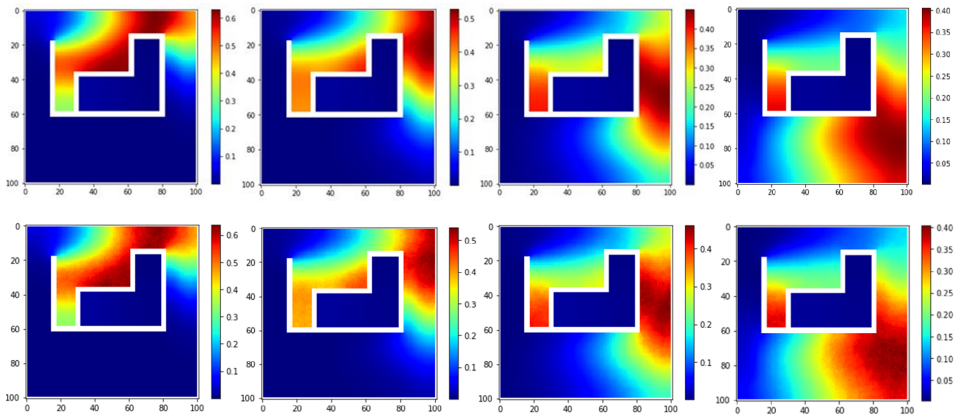

Figure 2: Ground truth (top) and PINN results (bottom) for $cx = 20$, $cy = 20$, $t = 2000, 3000, 4000, 5000$.

We executed the PINN and numerical model across 34 combinations of cx and cy values, with timesteps ranging from 1000 to 5000. The PINN consistently returned results in under 2190 milliseconds while the numerical model's runtime increased significantly: over 27 minutes for 1000 timesteps, 45 minutes for 2000, 56 minutes for 3000, 77 minutes for 4000, and 92 minutes for 5000 timesteps. For 5000 timesteps, the PINN was over 2520 times faster. As the number of timesteps increases, the time advantage of the PINN over traditional methods approaches infinity. Figure 3 illustrates the progression for a source at $cx = 20, cy = 80$, from 15 to 4500 timesteps. Figure 4 shows the progression for a source at $cx = 80$, $cy = 80$, from 15 to 2500 timesteps. In both figures, the starting points are on the left, the numerical model results are in the middle, and the PINN results are on the right. The PINN's MSE consistently remained below 3E-5.

4 Conclusion

Pollution models are crucial in combating climate change by identifying high-impact sources and informing targeted emission reduction efforts. These models also help optimize renewable energy strategies, such as the placement of wind and solar farms, to lower carbon footprints. Traditionally, scientists model pollution using complex physical simulations that rely on resource-intensive and time-consuming numerical methods. A promising alternative is the use of PINNs, which offer three key advantages: they are faster, capable of solving for any subdomain within a solution space, and resolution-invariant. As operators, PINNs can bypass sequential timestep calculations, efficiently solving time-dependent partial differential equations for any point in time. Additionally, their ability to operate on subdomains, provides greater flexibility and speed. Finally, PINNs trained on lower-resolution data can generate higher-resolution predictions, enhancing their utility in complex simulations. In our research, the PINN accurately modeled pollution spread for any source location with no-slip and zero-flux boundaries around obstructions, achieving an MSE below 3E-5 and performing 2520 times faster than numerical simulations. Our results also demonstrate that the time advantage of neural networks over numerical methods increases with both the time domain and spatial resolution. For future work, we aim to support arbitrary grid geometries and dynamic obstructions.

References

[1] Boško Josimović, Dušan Todorović, Aleksandar Jovović, and Božidar Manić. Air pollution modeling to support strategic environmental assessment: case study—national emission reduction plan for coal-fired thermal power plants in serbia. *Environment, Development and Sustainability*, 26(6):16249–16265, Jun 2024.

[2] Abdul-Lateef Balogun, Abdulwaheed Tella, Lavania Baloo, and Naheem Adebisi. A review of the inter-correlation of climate change, air pollution and urban sustainability using novel machine learning algorithms and spatial information science. *Urban Climate*, 40:100989, December 2021.

[3] Kipruto Kirwa, Adam A. Szpiro, Lianne Sheppard, Paul D. Sampson, Meng Wang, Joshua P. Keller, Michael T. Young, Sun-Young Kim, Timothy V. Larson, and Joel D. Kaufman. Fine-scale air pollution models for epidemiologic research: Insights from approaches developed in the multi-ethnic study of atherosclerosis and air pollution (mesa air). *Current Environmental Health Reports*, 8(2):113–126, Jun 2021.

[4] Muhammad Bilal Hafeez and Marek Krawczuk. A review: Applications of the spectral finite element method. *Archives of Computational Methods in Engineering*, 30(5):3453–3465, Jun 2023.

[5] Andrea Colagrossi and Maurizio Landrini. Numerical simulation of interfacial flows by smoothed particle hydrodynamics. *Journal of Computational Physics*, 191(2):448–475, 2003.

[6] Ali Reza Miroliaei, Farhad Shahraki, and Hossein Atashi. Computational fluid dynamics simulations of pressure drop and heat transfer in fixed bed reactor with spherical particles. *Korean Journal of Chemical Engineering*, 28(6):1474–1479, Jun 2011.

[7] K. Ishihara, M. Iguchi, and K. Kamo. *Numerical Simulation of Lava Flows on Some Volcanoes in Japan*, pages 174–207. Springer Berlin Heidelberg, Berlin, Heidelberg, 1990.

[8] Matthew Chantry, Hannah Christensen, Peter Dueben, and Tim Palmer. Opportunities and challenges for machine learning in weather and climate modelling: hard, medium and soft ai. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200083, 2021.

[9] Giovanni Calzolari and Wei Liu. Deep learning to replace, improve, or aid cfd analysis in built environment applications: A review. *Building and Environment*, 206:108315, 2021.

[10] Ricardo Vinuesa and Steven L. Brunton. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2(6):358–366, Jun 2022.

[11] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, Jul 2022.

[12] 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.

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

[14] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[15] Raphael Leiteritz, Marcel Hurler, and Dirk Pflüger. Learning free-surface flow with physics-informed neural networks. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1668–1673, 2021.

[16] Levi D. McClenny and Ulisses M. Braga-Neto. Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722, 2023.

[17] Yang Hu, ShiTing Zhang, Qiang He, and Decai Li. Diffuse interface-lattice boltzmann modeling for heat and mass transfer with neumann boundary condition in complex and evolving geometries. *International Journal of Heat and Mass Transfer*, 215:124480, 2023.

[18] Yicheng Wang, Xiaotian Han, Chia-Yuan Chang, Daochen Zha, Ulisses Braga-Neto, and Xia Hu. Auto-pinn: Understanding and optimizing physics-informed neural architecture, 2023.

5 Supplementary Material

5.1 Equations

$$u = 0 \tag{5}$$

$$v = 0 \tag{6}$$

$$\frac{dp}{dn} = 0 \tag{7}$$

$$\frac{d\phi}{dn} = 0 \tag{8}$$

$$\begin{aligned} u_{i,j}^{n+1} = u_{i,j}^n - u_{i,j}^n \frac{\Delta t}{\Delta x}\left(u_{i,j}^n - u_{i-1,j}^n\right) - v_{i,j}^n \frac{\Delta t}{\Delta y}\left(u_{i,j}^n - u_{i,j-1}^n\right) - \frac{\Delta t}{\rho 2\Delta x}\left(p_{i+1,j}^n - p_{i-1,j}^n\right) \\ + \nu\left(\frac{\Delta t}{\Delta x^2}\left(u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n\right) + \frac{\Delta t}{\Delta y^2}\left(u_{i,j+1}^n - 2u_{i,j}^n + u_{i,j-1}^n\right)\right) \end{aligned} \tag{9}$$

$$\begin{aligned} v_{i,j}^{n+1} = v_{i,j}^n - u_{i,j}^n \frac{\Delta t}{\Delta x}\left(v_{i,j}^n - v_{i-1,j}^n\right) - v_{i,j}^n \frac{\Delta t}{\Delta y}\left(v_{i,j}^n - v_{i,j-1}^n\right) - \frac{\Delta t}{\rho 2\Delta y}\left(p_{i,j+1}^n - p_{i,j-1}^n\right) \\ + \nu\left(\frac{\Delta t}{\Delta x^2}\left(v_{i+1,j}^n - 2v_{i,j}^n + v_{i-1,j}^n\right) + \frac{\Delta t}{\Delta y^2}\left(v_{i,j+1}^n - 2v_{i,j}^n + v_{i,j-1}^n\right)\right) \end{aligned} \tag{10}$$

$$\begin{aligned} p_{i,j}^n = \frac{\left(p_{i+1,j}^n + p_{i-1,j}^n\right)\Delta y^2 + \left(p_{i,j+1}^n + p_{i,j-1}^n\right)\Delta x^2}{2\left(\Delta x^2 + \Delta y^2\right)} - \frac{\rho \Delta x^2 \Delta y^2}{2\left(\Delta x^2 + \Delta y^2\right)} \\ \times\left[\frac{1}{\Delta t}\left(\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} + \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y}\right) - \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}\frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}\right. \\ \left. -2\frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}\frac{v_{i+1,j} - v_{i-1,j}}{2\Delta x} - \frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y}\frac{v_{i,j+1} - v_{i,j-1}}{2\Delta y}\right] \end{aligned} \tag{11}$$

$$\begin{aligned} \phi_{i,j}^{k+1} = \phi_{i,j}^k + (\Delta t)\left[\mu\left(\frac{\phi_{i+1,j}^k - 2\phi_{i,j}^k + \phi_{i-1,j}^k}{(\Delta x)^2} + \frac{\phi_{i,j-1}^k - 2\phi_{i,j}^k + \phi_{i,j+1}^k}{(\Delta y)^2}\right)\right. \\ \left. -u\frac{\phi_{i+1,j}^k - \phi_{i-1,j}^k}{2\Delta x} - v\frac{\phi_{i,j+1}^k - \phi_{i,j-1}^k}{2\Delta y}\right] \end{aligned} \tag{12}$$

$$x_{mid} = \frac{x_{min} + x_{max}}{2} \tag{13}$$

$$x_{scaled} = \frac{x - x_{mid}}{x_{max} + x_{min}} \tag{14}$$

$$\mathcal{L}_S = \frac{1}{n}\sum_{i=1}^{n}(y_{pred} - y_{act})^2 \tag{15}$$

$$\mathcal{L}_I = \frac{1}{n}\sum_{i=1}^{n}(y_{pred} - y_{act})^2 \tag{16}$$

$$L_1 = \overline{\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)^2} \tag{17}$$

$$L_2 = \overline{\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{1}{\rho}\frac{\partial p}{\partial x} - \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)\right)^2} \tag{18}$$

$$L_3 = \overline{\left(\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{1}{\rho}\frac{\partial p}{\partial y} - \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)\right)^2} \tag{19}$$

$$L_4 = \overline{\left(\frac{\partial \phi}{\partial t} - \mu\left(\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2}\right) + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y}\right)^2} \tag{20}$$

$$L_5 = \overline{u^2}, \forall (X, Y) \in \Omega \tag{21}$$

$$L_6 = \overline{v^2}, \forall (X, Y) \in \Omega \tag{22}$$

$$L_7 = \overline{\left(\frac{\partial p}{\partial n}\right)^2}, \forall (X, Y) \in \partial\Omega \tag{23}$$

$$L_8 = \overline{\left(\frac{\partial \phi}{\partial n}\right)^2}, \forall (X, Y) \in \partial\Omega \tag{24}$$

$$L_U = \sum_{i=1}^{8} w_i L_i \tag{25}$$

$$L_H = L_S + L_I + w \cdot L_U \tag{26}$$

5.2 Figures

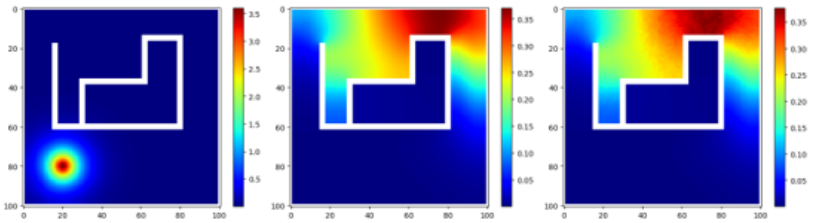

Figure 3: Ground Truth vs. PINN comparison for 4500 timesteps for pollution source (20, 80). Note. Left: Pollution Source $cx = 20, cy = 80, t = 15$. Middle: Ground Truth $t = 4500$. Right: PINN Results $t = 4500$

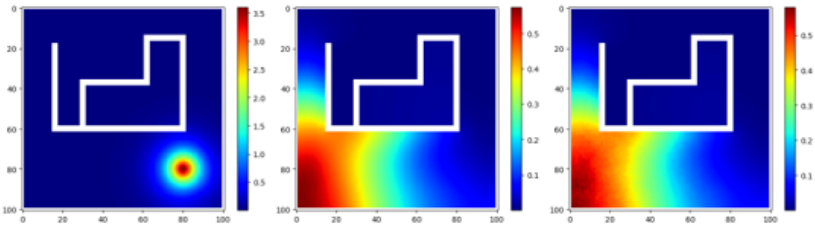

Figure 4: Ground Truth vs. PINN comparison for 2500 timesteps for pollution source (80, 80). Note. Left: Pollution Source $cx = 80, cy = 80, t = 15$. Middle: Ground Truth $t = 2500$. Right: PINN Results $t = 2500$