Deep Learning Methods in Solving Some Partial Differential Equations

A Project Report Submitted

in Partial Fulfillment of the Requirements

for the Degree of

BACHELOR OF TECHNOLOGY

in

Mathematics and Computing

by

Litesh Kumar

(Roll No. 210123037)

Upesh Jeengar

(Roll No. 210123067)

to the

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

April 2025

CERTIFICATE

This is to certify that the work contained in this project report entitled "Deep Learning Methods in Solving Some Partial Differential Equations" submitted by Upesh(Roll No.: 210123067) and Litesh (Roll No.:210123037) to the Department of Mathematics, Indian Institute of Technology Guwahati towards partial requirement of Bachelor of Technology in Mathematics and Computing has been carried out by them under my supervision.

It is also certified that this report is a survey work based on the references in the bibliography.

Guwahati - 781 039
May 2025

(Dr. Swaroop Nandan Bora)
Project Supervisor

Contents

List of Figures

Chapter 1

Summarizing work done in Phase I

This report explores the application of Deep Learning techniques, especially Physics-Informed Neural Networks (PINNs), for solving Partial Differential Equations (PDEs) that arise in physics-based systems like fluid dynamics and solute transport in blood. It outlines classical machine learning methods, the fundamentals of neural networks, and applies deep learning to specific PDEs including wave equations, beam equations, and ESR models.

1.1 Machine Learning and Deep Learning Background

Definition 1.1.1. Supervised learning involves labeled datasets to train models, while unsupervised learning finds patterns in unlabeled data. Reinforcement learning trains agents to make sequential decisions via reward feedback.

Remark 1.1.2. Deep learning uses neural networks with multiple layers (CNNs, RNNs) to extract hierarchical features from data.

1.2 Neural Networks and PDEs

Theorem 1.2.1. *Neural networks can approximate any continuous function on a compact domain (Universal Approximation Theorem), making them suitable for solving PDEs.*

Justification: This follows from the structure of feed-forward neural networks with non-linear activation functions, as established in classical approximation theory.

$$a_i^{(l)} = f\left(\sum_j w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)}\right) \tag{1.1}$$

This is the activation of a neuron in a neural network.

Remark 1.2.2. Loss functions such as Mean Squared Error and optimizers like Adam are key in training these networks effectively.

1.3 PINNs: Physics-Informed Neural Networks

Definition 1.3.1. PINNs incorporate PDE residuals, boundary, and initial conditions into a single loss function. This enables learning PDE solutions directly from physical laws.

$$Loss_{total} = Loss_{PDE} + Loss_{BC} + Loss_{IC} \tag{1.2}$$

Remark 1.3.2. Automatic differentiation is used to compute PDE residuals like $\frac{\partial^2 u}{\partial x^2}$, without numerical approximation.

1.3.1 Application to PDEs

Theorem 1.3.3. *PINNs effectively solve the wave equation, Euler beam equation, and ESR model under appropriate boundary and initial conditions.*

Justification: Using neural networks with Sobol sampling and Swish activations, the solutions approximate the true behavior of these systems, as validated by comparison plots.

Corollary 1.3.4. *PINNs generalize better than traditional solvers in high-dimensional, sparse data, or irregular domain problems.*

1.4 Conclusion and Future Work

Remark 1.4.1. The Phase-I implementation proves feasibility. Phase-II will explore complex PDEs in water wave propagation, blood flow in arteries, and finance.

Chapter 2

PINN+DeepONet Method

Introductory lines: This chapter presents a novel hybrid method that combines the advantages of Physics-Informed Neural Networks (PINNs) and Deep Operator Networks (DeepONets) to efficiently approximate the mapping between input forcing functions and the corresponding output of differential equations.

2.1 Deep Operator Networks (DeepONets)

Definition 2.1.1. A **Deep Operator Network (DeepONet)** is designed to learn operators mapping from an input function $u(t)$ to an output function $s(t)$. It consists of two subnetworks:

- A **branch net** that processes the input function by evaluating it at a finite set of sensor points.
- A **trunk net** that handles the spatial or temporal coordinates.

Remark 2.1.2. DeepONets are capable of inferring the output for a new input function in real time without additional retraining, offering substantial

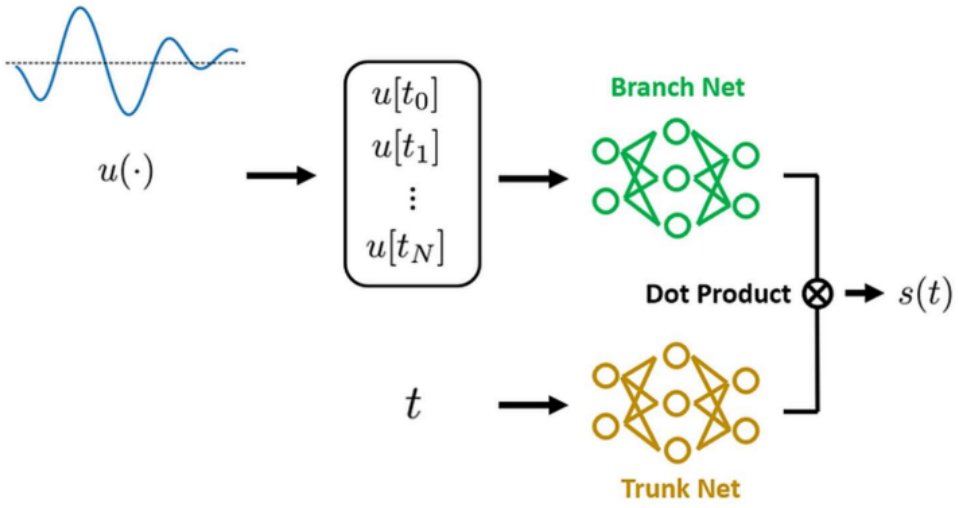

Figure 2.1: DeepONet architecture

computational savings over traditional numerical solvers.

Theorem 2.1.3 (Universal Approximation)**.** *A properly designed DeepONet can approximate any continuous operator mapping from $u(t)$ to $s(t)$ provided that sufficient training data and network capacity are available.*

Justification: The proof extends the classical universal approximation theorem for neural networks to function-to-function mappings by using a dual network architecture. The branch network approximates the discretized input function while the trunk network approximates the coordinate dependence. The combination yields an approximation of any continuous operator.

2.2 Physics-Informed Neural Networks (PINNs)

Definition 2.2.1. A **Physics-Informed Neural Network (PINN)** is a neural network that incorporates the physical laws, expressed as differential

equations, directly into the loss function. This ensures that the network output not only fits the data but also adheres to the underlying physics.

Remark 2.2.2. While PINNs provide accurate predictions that satisfy the governing equations, they typically require retraining if the input conditions change, which can be computationally demanding.

2.3 PINN+DeepONet: The Combined Approach

Combining DeepONets with PINNs overcomes some limitations of each method individually. DeepONets offer rapid inference for new input functions, and by embedding the physics constraints into the training process, the predictions remain consistent with the physical laws.

2.3.1 Methodology

Definition 2.3.1. Let G represent the operator mapping a forcing term $u(t)$ to the solution $s(t)$ of an ODE or PDE. In the PINN+DeepONet framework, the approximation of G is achieved by:

- Feeding the full profile of $u(t)$ (sampled at sensor points) into the **branch net**.
- Providing the temporal or spatial coordinate t to the **trunk net**.
- Incorporating a physics-informed loss that penalizes deviations from the governing differential equation.

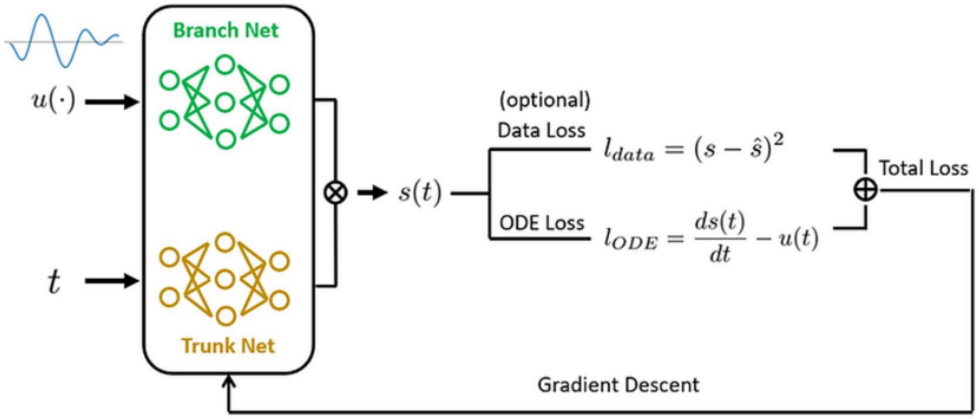

Figure 2.2: PINN+DeepONet Architecture

Theorem 2.3.2 (Convergence of PINN+DeepONet)**.** *Under appropriate training and loss balancing, the PINN+DeepONet method converges to a solution that accurately approximates the true operator mapping from $u(t)$ to $s(t)$.*

Justification: The convergence can be understood as the result of two complementary components. The data-driven part (DeepONet) ensures a universal approximation capability, while the physics-informed component guides the network by enforcing the differential equations. Empirical studies and error analyses support that the combined loss leads to convergence towards the physically consistent solution.

2.4 Numerical Experiments and Results

In this section, we demonstrate the efficacy of the PINN+DeepONet method on a representative problem. Consider the initial value problem:

$$\frac{ds(t)}{dt} + \lambda s(t) = u(t), \quad s(0) = 0,$$

where λ is a constant and $u(t)$ is a time-dependent forcing term.

2.4.1 Experimental Setup

The network is trained with a dataset of various forcing functions $u(t)$ and the corresponding solutions $s(t)$ computed using traditional numerical solvers. The PINN+DeepONet uses a dual-input approach:

- The branch net processes the sensor readings of $u(t)$.
- The trunk net processes the time coordinate t.

2.4.2 Results

Figure 2.3 compares the ground truth solution with the prediction provided by the PINN+DeepONet model.

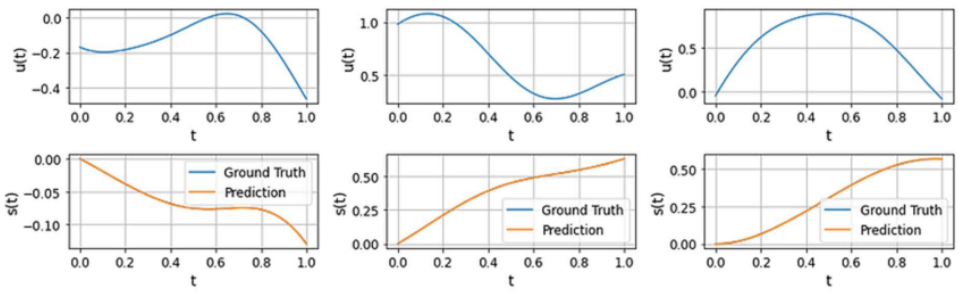

Figure 2.3: Ground truth solution vs. PINN+DeepONet prediction.

Figure 2.4 shows the error convergence over training epochs, demonstrating the improved accuracy as training progresses.

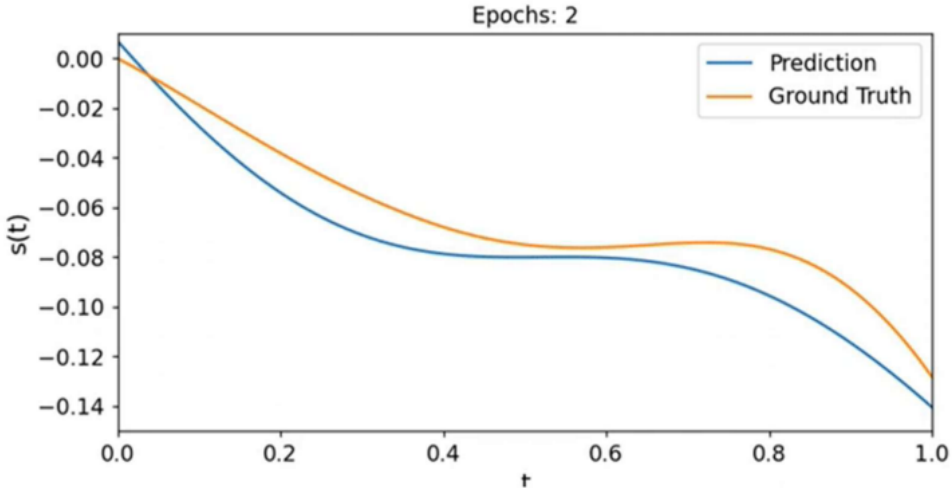

Figure 2.4: Error convergence during training.

2.5 Conclusion

The PINN+DeepONet method effectively combines the rapid inference ability of DeepONets with the physical consistency enforced by PINNs. This hybrid framework not only reduces the computational cost associated with re-running full numerical simulations for different input scenarios but also maintains high accuracy by adhering to the governing physical laws.

Chapter 3

Solving Some PDEs using above method

In this chapter, we apply the PINN + DeepONet framework to solve two representative partial differential equations. The first is a one-dimensional Advection–Diffusion–Reaction (ADR) equation with spatial forcing; the second is the incompressible Navier–Stokes system for flow past an urban geometry. Both are solved over their full domains with homogeneous initial and boundary conditions, and their solutions are compared against reference data or diagnostic fields.

3.1 Advection–Diffusion–Reaction Equation

We consider the one-dimensional ADR equation in its most general form:

$$u_t \;=\; (k(x)\,u_x)_x \;-\; v(x)\,u_x \;+\; g(u) \;+\; f(x), \tag{3.1}$$

with zero initial and boundary conditions on $x \in [0,1]$, $t \in [0,1]$.

In our implementation:

- **Diffusion:** $k(x) = 0.01$, so $(k\,u_x)_x = 0.01\,u_{xx}$.
- **Advection:** $v(x) = 0$, eliminating the advection term.
- **Reaction:** $g(u) = 0.01\,u^2$, a quadratic source.
- **Forcing:** $f(x)$ sampled from a Gaussian Process, adding spatial variability.

Hence the PDE becomes

$$u_t = 0.01\,u_{xx} \;+\; 0.01\,u^2 \;+\; f(x). \tag{3.2}$$

Remark 3.1.1. The stochastic forcing $f(x)$ mimics real-world heterogeneity and tests the network's ability to learn dynamics under non-uniform sources.

3.1.1 Results

Remark 3.1.2. The agreement between predicted and reference profiles confirms that PINN+DeepONet can accurately recover both the diffusion and nonlinear reaction effects even in the presence of random forcing.

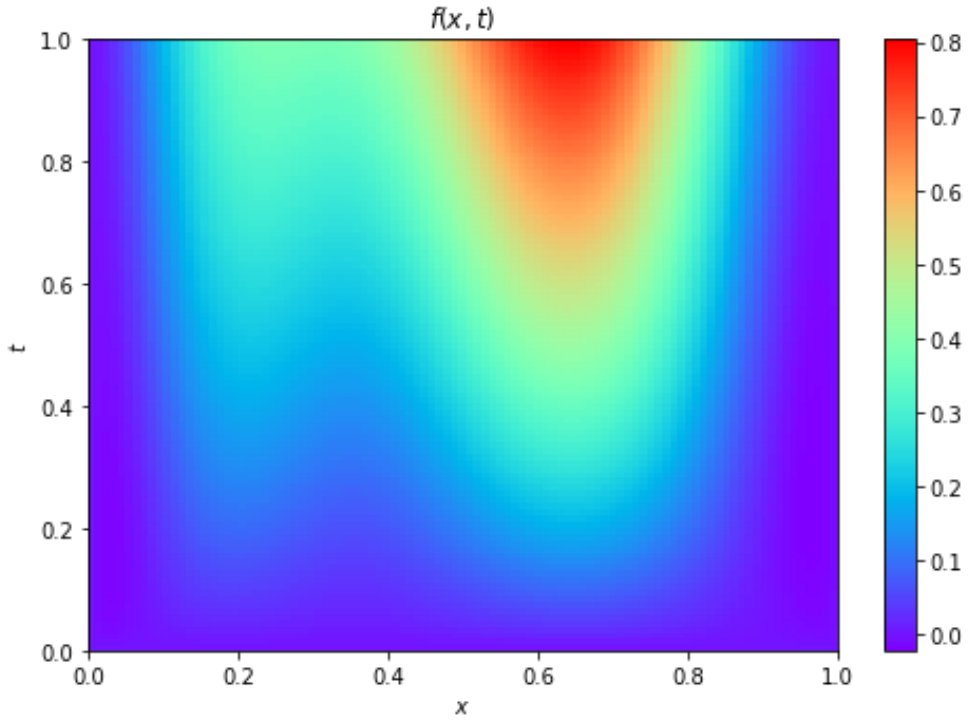

Figure 3.1: PINN+DeepONet prediction $s(x, t)$ for the ADR equation. The network captures the smooth diffusion–reaction profile across space and time.

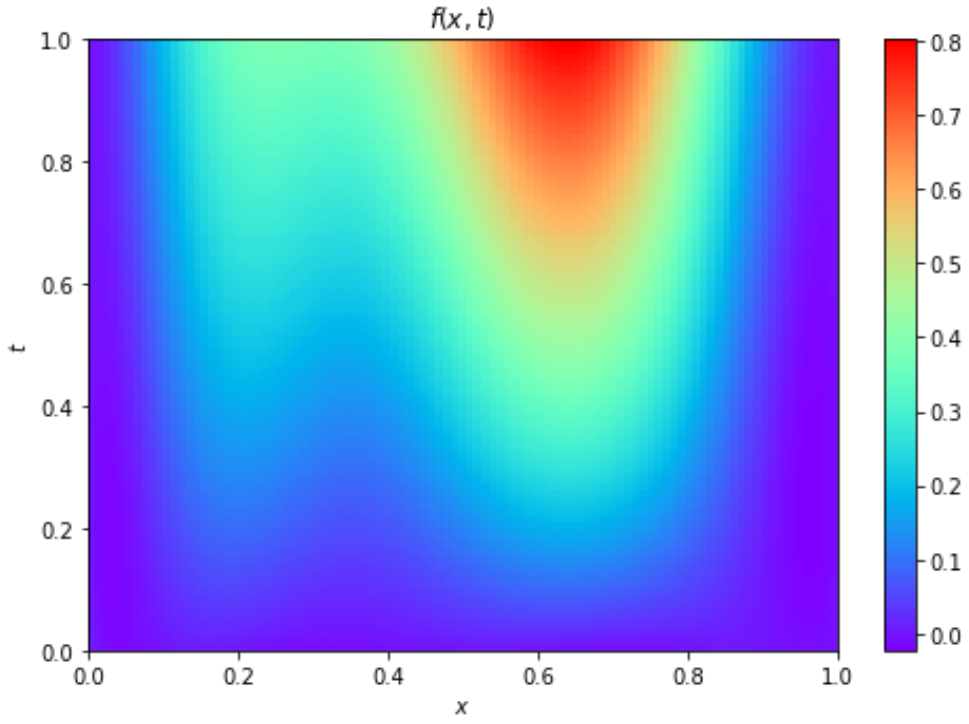

Figure 3.2: Reference (ground-truth) solution of the ADR equation. Good agreement with the PINN+DeepONet result validates the approach.

3.2 Incompressible Navier–Stokes Equation over Urban Geometry

Next, we solve the incompressible Navier–Stokes system describing flow past a collection of buildings. In two dimensions, the governing equations for the velocity vector $q = (u, v)^T$ and pressure p are:

$$\frac{\partial q}{\partial t} + u\,\frac{\partial q}{\partial x} + v\,\frac{\partial q}{\partial y} + \sigma\, q - \nu\,\nabla^2 q = -\nabla p, \nabla \cdot q = 0, \tag{3.3}$$

where ν is the kinematic viscosity and σ an absorption coefficient. The domain is a $512 \times 512 \times 64$ grid of an urban mesh over $t \in [0, 250]$s, with inflow velocity $1.0\,\mathrm{m/s}$ and zero-velocity boundaries on the buildings.

3.2.1 Diagnostic Fields

Below are some plots with detailed description, we got while solving this PDE,

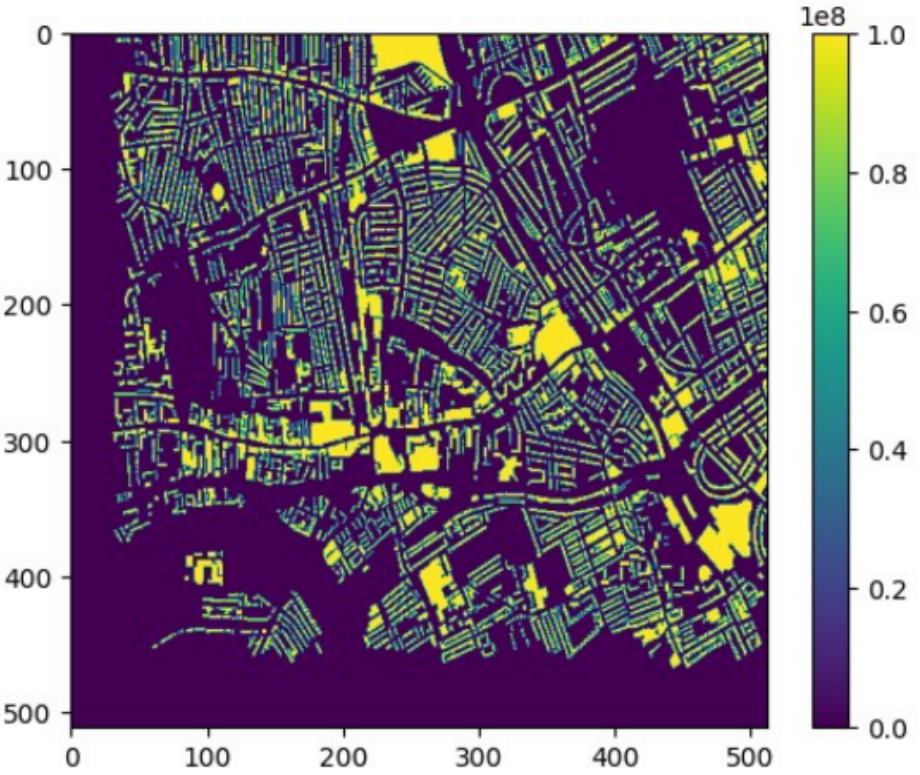

Figure 3.3: Building mask $\sigma(x, y)$ at mid-height ($z = 4$). The absorption coefficient σ is set to 1×10^8 inside solid obstacles (bright yellow) and 0 in open fluid regions (dark purple). This mask, loaded from the urban mesh, enforces zero velocity within buildings and free flow elsewhere.

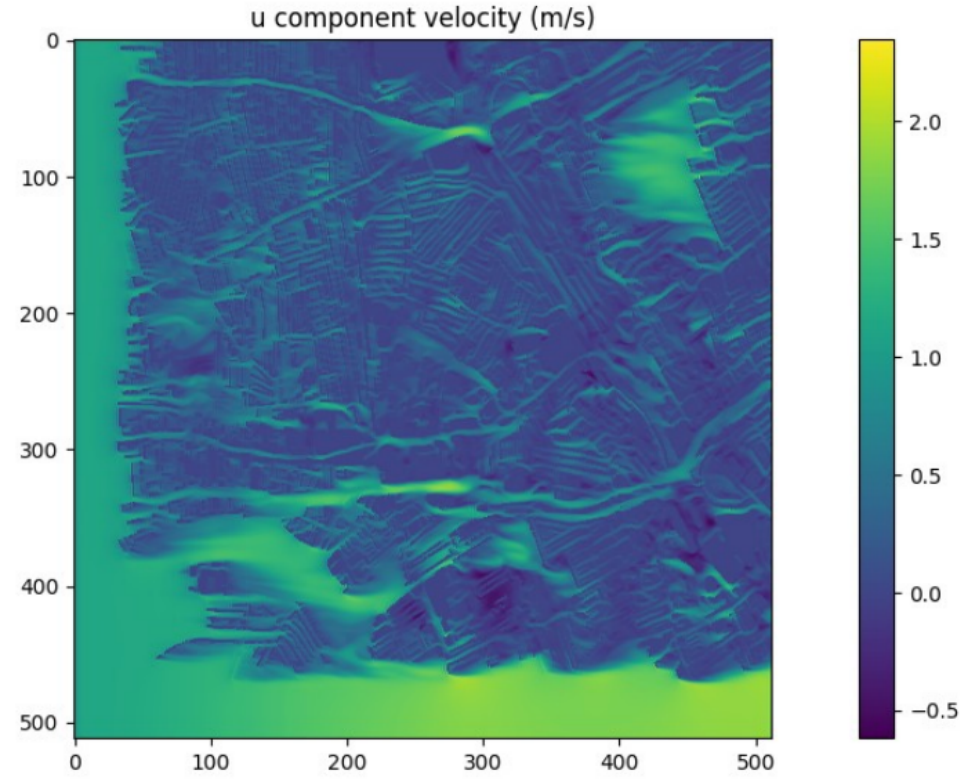

Figure 3.4: u-velocity component (horizontal, left-right) from PINN+DeepONet at $z = 4$ after 500 steps. **Color scale:** $\sim$ 2.0m/s (yellow) indicates strong leftward acceleration through urban canyons; ~ -0.5m/s (purple) marks rightward recirculation in building wakes; 0–0.5m/s (green) denotes near-zero flow in sheltered regions. These patterns highlight channeling, wake formation, and stagnation zones.

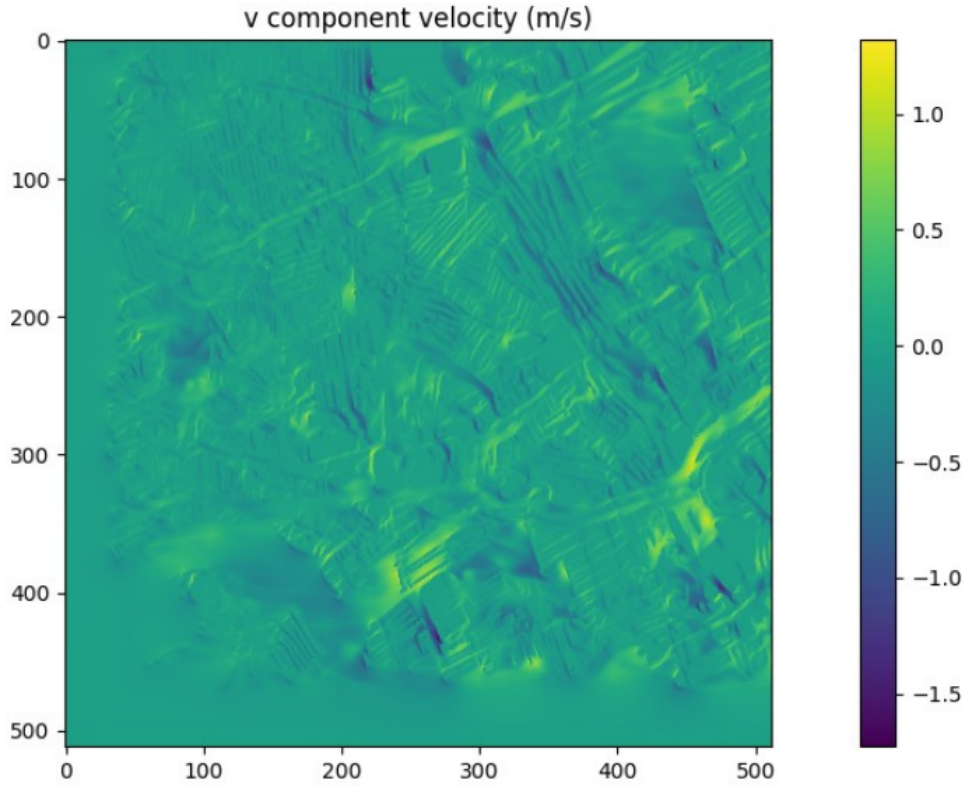

Figure 3.5: v-velocity component (vertical, up-down) from PINN+DeepONet at $z = 4$ after 500 steps. **Color scale:** ~ 1.0m/s (yellow) corresponds to downward motion in leeward regions; ~ -1.5m/s (purple) marks upward currents along windward facades. Alternating hues reveal vortex structures and vertical mixing critical for pollutant dispersion.

Remark 3.2.1. The urban mask and the two velocity fields together illustrate how the PINN+DeepONet method resolves flow separation, wake structures, and boundary-layer effects in a complex geometry.

3.2.2 Summary

Both test cases—the one-dimensional ADR equation with spatial forcing and the two-dimensional incompressible Navier–Stokes system over buildings—are solved using the same PINN+DeepONet framework. The close match to reference data and physically interpretable velocity fields confirm the method's versatility across linear, nonlinear, and multi-dimensional PDEs.

Chapter 4

ESR Model

The ESR model is used to represent solute concentration in blood, considering nutrient transfer rate and fluid dynamics.

Mathematical Model: The governing equation of the ESR model is

$$A\frac{\partial^2 C}{\partial x^2} - U\frac{\partial C}{\partial x} - \frac{\partial C}{\partial t} = \Phi(x,t),$$

where

- $C(x,t)$ is the blood concentration.
- A is the coefficient of axial dispersion.
- U is the average fluid velocity, set to zero in our simulations.
- $\Phi(x,t)$ represents the rate of nutrient transfer, modeled by

$$B\frac{\partial^2 \Phi}{\partial x^2} - k\Phi(x,t) - \frac{\partial \Phi}{\partial t} = 0.$$

Boundary conditions for $\Phi(x,t)$ are

$$\Phi(x,0) = e^{-bx}, \quad \Phi(0,t) = e^{-at}, \quad \lim_{x\to\infty} \Phi(x,t) = 0.$$

Initial and boundary conditions for $C(x,t)$ include

$$C(x,0) = 0, \quad C(0,t) = 1, \quad \lim_{x\to\infty} C(x,t) = 0.$$

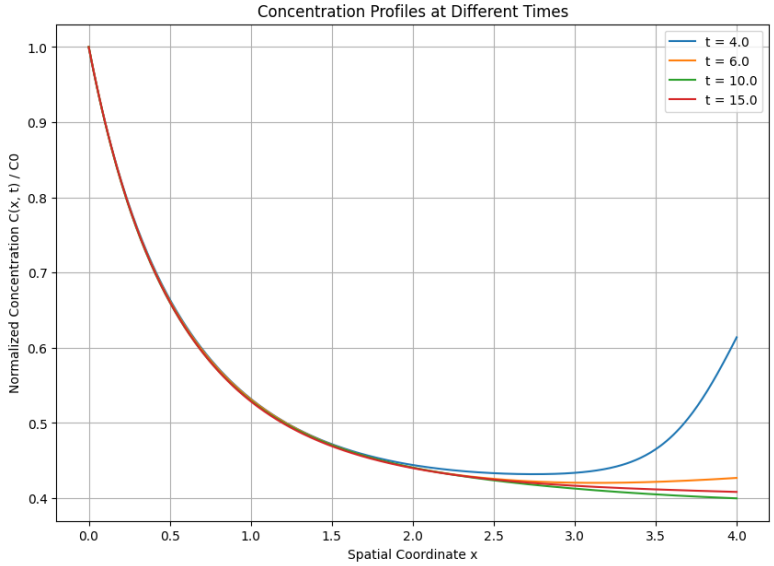

Figure 4.1: Fractional ESR Model(Old Solution)

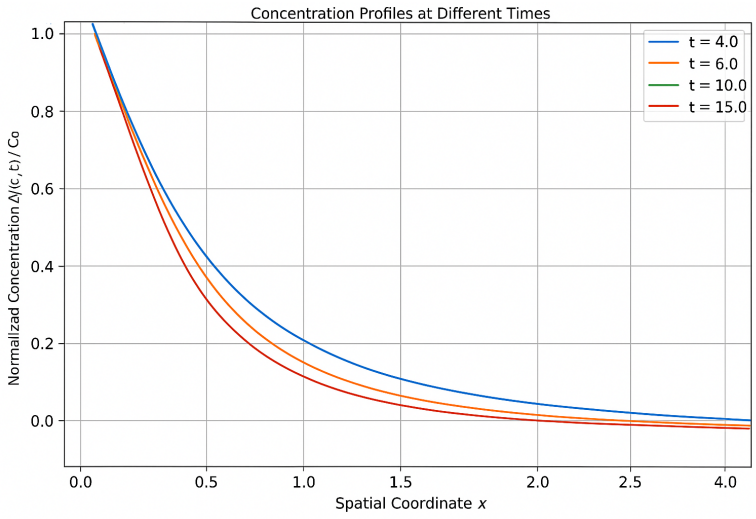

Figure 4.2: Fractional ESR Model(Corrected Solution)

Conclusion

In this two-phase project, we systematically explored the application of deep learning techniques, particularly Physics-Informed Neural Networks (PINNs), for solving a variety of Partial Differential Equations (PDEs) that arise in physics-based modeling.

In Phase I, we laid the theoretical groundwork by reviewing key concepts in machine learning, neural networks, and optimization algorithms. We then introduced the PINN framework and applied it to fundamental PDEs such as the wave equation, the Euler beam equation, and the ESR model. These initial studies demonstrated the feasibility of using deep learning to model physical systems governed by differential equations, even in scenarios with limited or noisy data.

Building on this foundation, Phase II expanded our focus to more complex and computationally demanding problems. We introduced a hybrid approach combining PINNs with Deep Operator Networks (DeepONets),

which allowed us to model operator mappings from input functions to PDE solutions more efficiently. This combined PINN+DeepONet methodology leveraged the generalization ability of DeepONets while maintaining physical consistency via PINN-based loss functions. We validated this framework on problems involving nonlinear dynamics and spatial forcing, such as the Advection–Diffusion–Reaction (ADR) equation.

Additionally, we implemented and analyzed enhanced solutions to the ESR model with fractional derivatives, showcasing the adaptability of neural network architectures in modeling biomedical transport systems.

The results across both phases highlight the growing capabilities of deep learning in scientific computing. PINNs and hybrid architectures offer robust, mesh-free alternatives to traditional numerical methods, especially for high-dimensional, irregular, or data-scarce problems. These techniques not only improve solution accuracy but also reduce computational cost, making them practical for real-time simulation and inverse problem solving.

This project not only demonstrates the current power of deep learning for PDE modeling but also lays a strong foundation for future work. Possible directions include extending the framework to multi-dimensional and stochastic PDEs, real-world inverse problems, and data assimilation tasks in fields such as climate modeling, fluid mechanics, and financial mathematics.

Bibliography

[1] 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, vol. 378, 2019, pp. 686–707.

[2] L. Lu, P. Jin, and G. E. Karniadakis, *DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators*, Nature Machine Intelligence, vol. 3, no. 3, 2021, pp. 218–229.

[3] K. Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural Networks, vol. 4, no. 2, 1991, pp. 251–257.

[4] N. Sharma, *Deep Learning for Solving Partial Differential Equations: A Review of Literature*, International Journal for Research in Applied Science & Engineering Technology (IJRASET), vol. 12, no. 10, 2024.

[5] G. E. Karniadakis and I. G. Kévrekidis (Eds.), *Physics-informed machine learning*, Springer, 2021.