INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, RAICHUR

Deep Learning Solutions for the Biharmonic Equation

Abhay Tiwari, AD23B1001
Ansh Gupta, AD23B1005
Ayush Raj Singh, AD23B1009
Deepak Kumar, AD23B1016
Rahul Dhaka, AD23B1044
December 2, 2025

1 INTRODUCTION

This report analyzes the application of deep learning techniques to solve the Biharmonic Equation:

$$\Delta^2 u = f,$$

a fourth-order Partial Differential Equation (PDE) commonly encountered in elasticity and fluid mechanics (e.g., Stokes flow).

The experiment was conducted in two distinct phases using different methodologies to solve two specific variations of the problem (Example 3.1 and Example 3.2) under Condition P1.

1.1 PROBLEM DEFINITIONS

The problems were solved on the unit square domain

$$\Omega = (0,1)^2,$$

under Condition P1, which enforces both Dirichlet and Neumann boundary conditions:

$$g_1 = u, \qquad g_2 = \frac{\partial u}{\partial n} \quad \text{on } \partial\Omega.$$

1.2 Example 3.1 (Sinusoidal)

$$u(x,y) = \frac{2}{\pi^2}\sin(\pi x_1)\sin(\pi x_2).$$

1.3 Example 3.2 (Polynomial)

$$u(x,y) = x_1^2 x_2^2 (1-x_1)^2 (1-x_2)^2.$$

2 Phase 1: Physics-Informed Neural Networks (PINN)

2.1 Methodology (Strong Form)

In Phase 1, the problem was solved using the strong form of the PDE. The neural network approximated the solution $u_\theta(x,y)$, and Automatic Differentiation was used to compute the fourth-order derivatives required to evaluate the residual $\Delta^2 u - f$.

Loss Function: $L = L_{\text{PDE}} + \lambda_{\text{BC1}} L_{\text{Dirichlet}} + \lambda_{\text{BC2}} L_{\text{Neumann}}$.

Architecture: Fully connected network (5 layers, 60 neurons per layer, Tanh activation).

Hardware: CPU.

Training Time: Approx. 12.3 hours (44,000 s).

2.2 Analysis

Problem	**Training Time (s)**	**Script Time (s)**	**Final Loss**	**Rel. L2**	**Rel. H1**	**Rel. H2**
Ex 3.1 (Sin)	44216.61	44218.29	2.4768×10^{-3}	1.4162×10^{-2}	6.9963×10^{-3}	1.0545×10^{-2}
Ex 3.2 (Poly)	45116.46	45118.04	2.4871×10^{-4}	1.8243×10^{-2}	3.4321×10^{-2}	5.5712×10^{-2}

Table 2.1: Performance metrics for Examples 3.1 and 3.2.

Phase 1 demonstrated high precision. The PINN successfully captured both the sinusoidal curvature and the specific polynomial behavior. However, calculating 4th-order derivatives via automatic differentiation on the CPU resulted in extremely high computational costs.

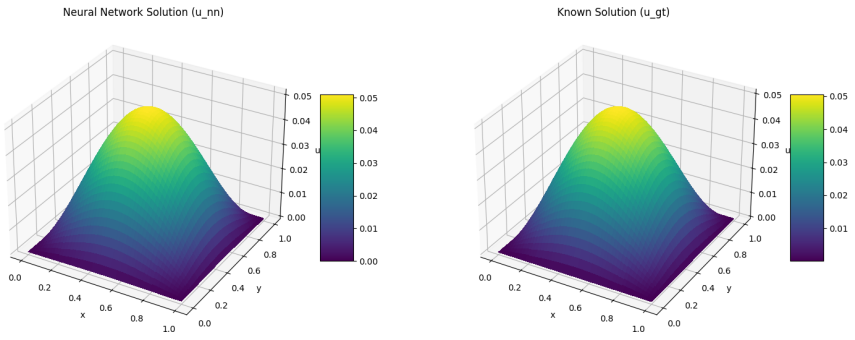


Figure 2.1: 3D Solution Comparison E1.

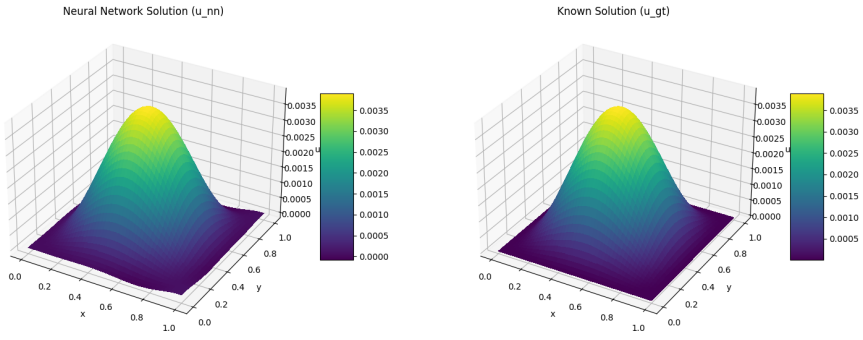


Figure 2.2: 3D Solution Comparison E2.

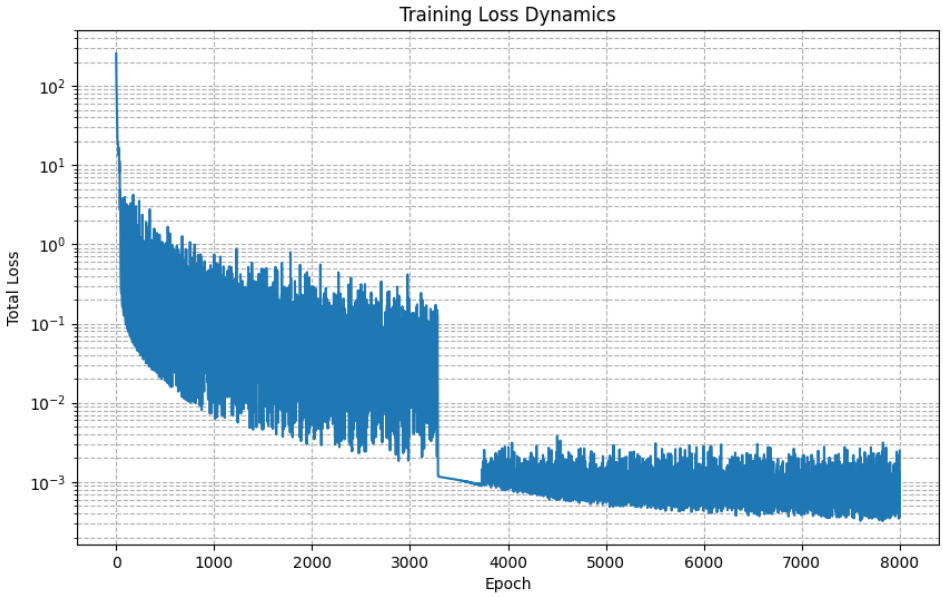


Figure 2.3: Loss History Q1.

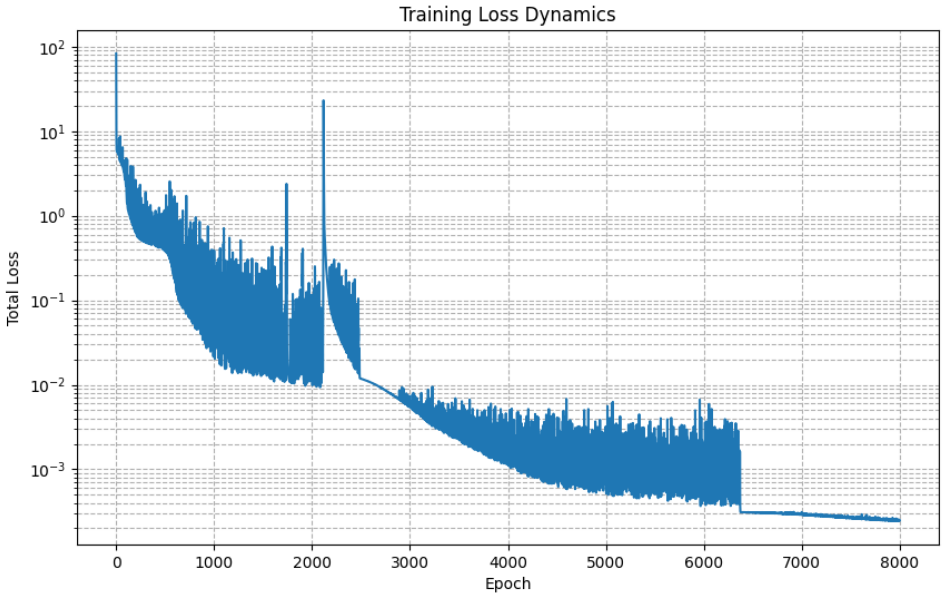


Figure 2.4: Loss History Q2.

3 Phase 2: Deep Ritz Method (DRM)

3.1 Methodology (Variational Form)

In Phase 2, the problem was solved using the weak (variational) form. Instead of minimizing the PDE residual directly, the neural network minimized the potential energy functional of the system:

$$L(\theta) = \int_\Omega \left(\frac{1}{2}(\Delta u)^2 - f u \right) dx + \beta_1 \int_{\partial\Omega} (u - g_1)^2 \, ds + \beta_2 \int_{\partial\Omega} \left(\frac{\partial u}{\partial n} - g_2 \right)^2 ds$$

Advantage: Only requires second-order derivatives (Laplacian), reducing computational complexity.

Training: 20,000 epochs (Adam) + 1,000 epochs (L-BFGS).

Hardware: GPU (CUDA).

Training Time: Approx. 1.25 hours (4,500–4,700 s).

3.2 Analysis of Phase 2

Example	L2 Error	Relative L2 Error	Energy Error (L2 + H1)	Relative Energy Error	H2 Error	Relative H2 Error
E1	1.602×10^{-3}	6.432×10^{-2}	1.338×10^{-2}	9.737×10^{-2}	6.658×10^{-2}	1.915×10^{-1}
E2	4.926×10^{-3}	3.155×10^{0}	3.333×10^{-2}	3.619×10^{0}	2.153×10^{-1}	4.167×10^{0}

Table 3.1: Error metrics for Examples E1 and E2 using L2, H1, and H2 norms.

3.3 Example 3.1 (Sinusoidal)

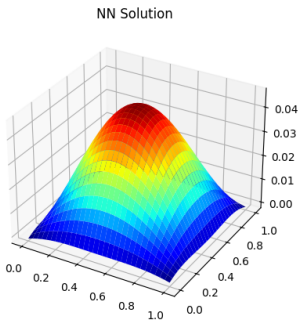
The visual results (`NNsolution1.png`) show a smooth and accurate surface reconstruction. The corresponding error plot (`Error1.png`) exhibits slightly higher values near the boundaries but remains low overall, with a maximum error of approximately 0.004.

3.4 Example 3.2 (Polynomial)

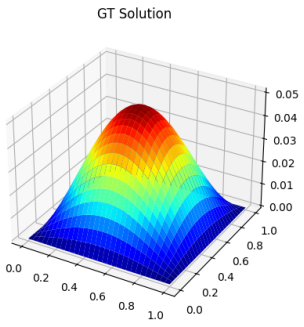
The relative error is high (315%) not because the reconstructed shape is incorrect, but because the target values are extremely small. The polynomial

$$x^2(1-x)^2 \cdots$$

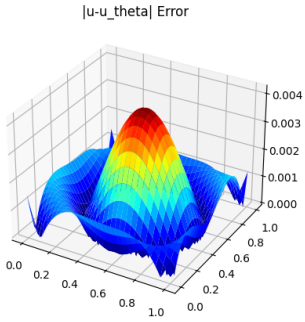
has a maximum value of approximately 0.0039. The network's absolute error (about 0.0049) is of the same order of magnitude as the signal itself, which leads to a poor relative error metric despite the 3D plot showing the correct overall shape.



(a) NN1 Solution



(b) Ground Truth (GT1)



(c) Error2

Figure 3.1: Comparison of NN, Ground Truth (GT) solutions 1.

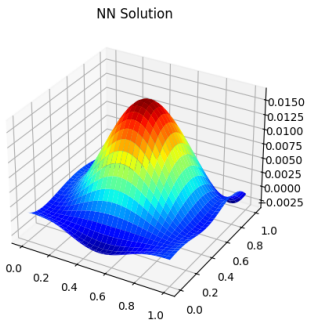
4 Comparative Conclusion

The experiment highlights a trade-off between accuracy and computational efficiency, specifically regarding Condition P1 (boundary enforcement).

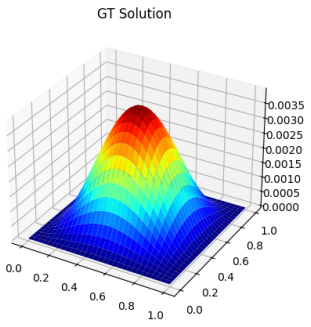
4.1 1. Accuracy

1. **PINN (Phase 1)** achieved superior accuracy (approximately 1.4% error) for both the sinusoidal and polynomial problems. It handled the P1 boundary conditions robustly through soft loss constraints.

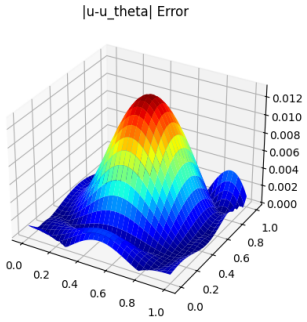
2. **DRM (Phase 2)** struggled with the polynomial problem due to its extremely small solution magnitude, where the boundary penalty terms likely dominated the energy minimization process. This resulted in scale sensitivity issues.



(a) NN Solution 2



(b) Ground Truth (GT2)



(c) Error2

Figure 3.2: Comparison of NN and Ground Truth (GT) solutions 2.

4.2 2. Efficiency

- **DRM (Phase 2)** was approximately ten times faster than PINN. By reducing the derivative order from four to two and leveraging GPU acceleration, the DRM approach proved to be far more scalable for high-order PDEs, provided that the solution magnitude is normalized or the loss weights are properly tuned.

Final Verdict. For the Biharmonic operator under Condition P1, PINN provides reliable, high-fidelity solutions at the cost of computational time, while DRM offers rapid approximations that require careful hyperparameter tuning when dealing with small-amplitude solutions.

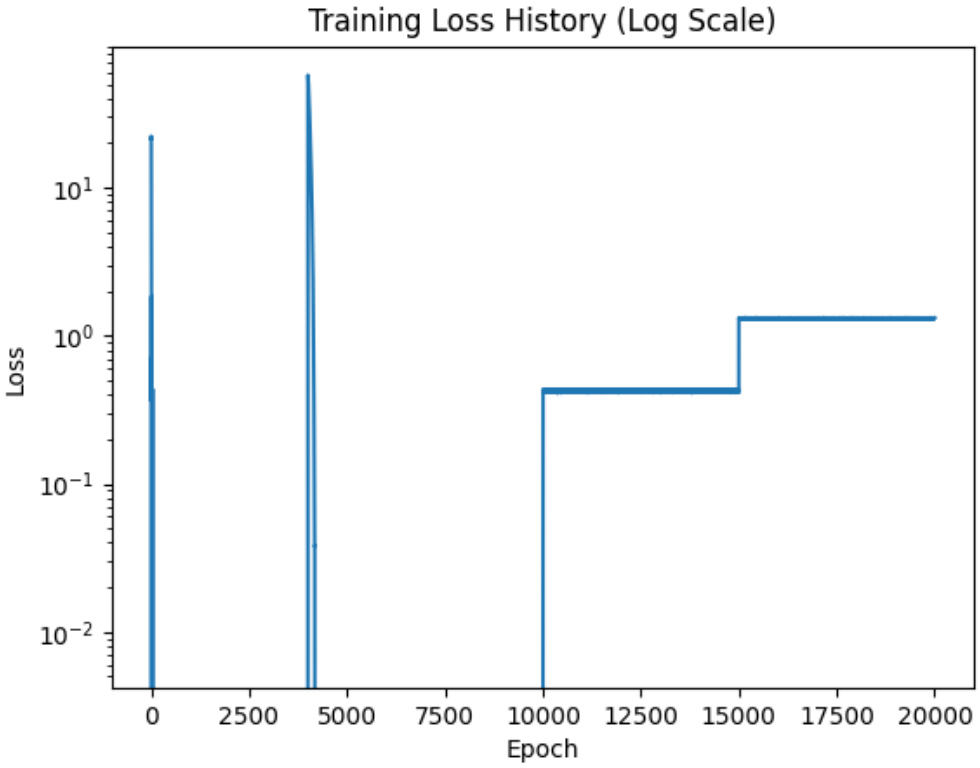


Figure 3.3: Loss History Q1 DRM.

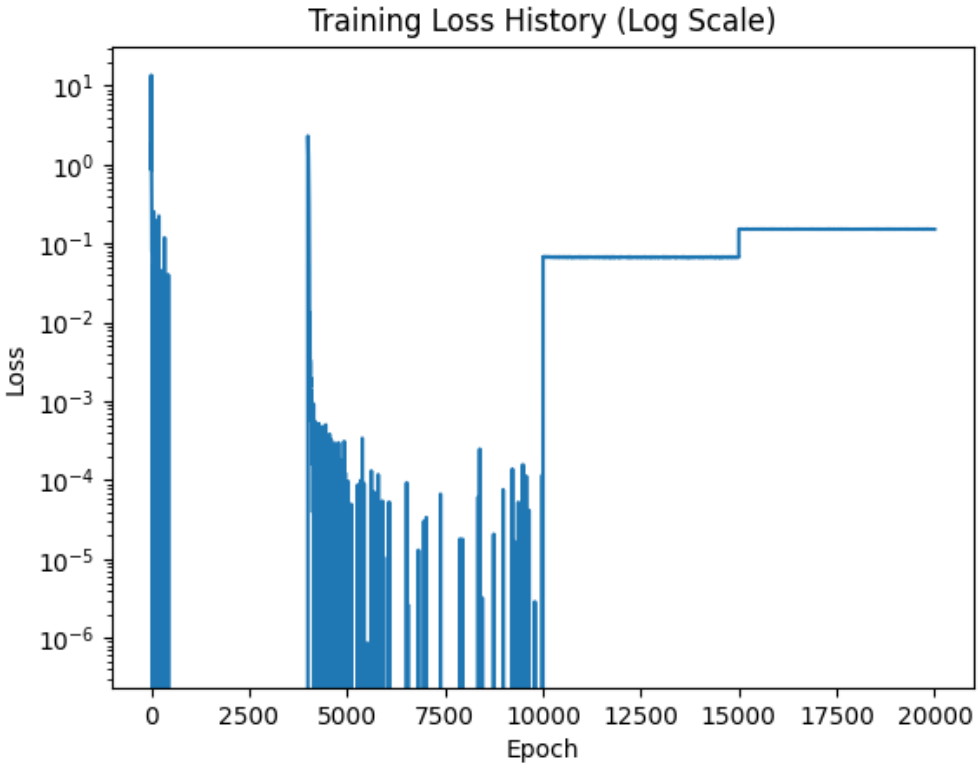


Figure 3.4: Loss History Q2 DRM.