Deep Learning Approaches for the Biharmonic Equation (P4)

Phase I (PINN) and Phase II (DRM) Combined Report

Team 4 — Deep Learning for Differential Equations (DL4DEs)

December 5, 2025

Abstract

This report presents a comprehensive study of deep learning methods for solving the biharmonic problem (P4) with Cahn–Hilliard type boundary conditions on the unit square. The study is divided into two phases. **Phase I** utilizes a Physics-Informed Neural Network (PINN) formulation, focusing on Example 3.2 using a polynomial manufactured solution. **Phase II** implements the Deep Ritz Method (DRM) based on the variational energy functional, applied to Example 3.1 and Example 3.2 (using a trigonometric solution). We provide full quantitative and qualitative results, including loss histories, 2D/3D visualizations, and numerical error metrics (L^2, H^1, H^2) for the implemented examples. Placeholders are provided for pending experiments to ensure a complete project structure.

Contents

1 General Introduction

The biharmonic equation plays a central role in thin-plate deformation, phase separation, and high-order diffusion phenomena. In this project, we solve the fourth-order PDE:

$$\Delta^2 u = f \quad \text{in } \Omega = (0,1)^2,$$

subject to the Cahn–Hilliard (P4) boundary conditions:

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

In all examples, f, g_1, and g_2 are derived analytically from a chosen manufactured solution $u(x, y)$. The project explores two distinct methodologies: the strong-form based Physics-Informed Neural Network (Phase I) and the variational-form based Deep Ritz Method (Phase II).

2 Phase I: Physics-Informed Neural Network (PINN)

2.1 PINN Methodology for P4 (General Workflow)

The PINN for P4 is constructed and trained using the following workflow:

1. **Sampling:** Random interior collocation points and uniformly distributed boundary points.
2. **Neural Network Architecture:** Fully-connected deep network with the following structure:
$$[2, 40, 40, 40, 40, 1], \qquad \tanh \text{ activation.}$$
3. **Automatic Differentiation:** Compute all derivatives up to fourth order using PyTorch autograd:
$$u,\ \nabla u,\ \Delta u,\ \nabla(\Delta u),\ \Delta^2 u.$$
4. **Loss Function:**
$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}} + (\text{zero-mean constraint}).$$
5. **Training Strategy:**
   - Adam optimizer (20,000 iterations).
   - L-BFGS refinement.

2.2 PINN Solution 1: Example 3.1 (P4 – Cahn–Hilliard Boundary Conditions)

2.2.1 Problem Description

In this experiment we solve the biharmonic equation with Cahn–Hilliard type boundary conditions:

$$\Delta^2 u = f \quad \text{in } \Omega = (0,1)^2,$$
$$\frac{\partial u}{\partial n} = g_1, \qquad \frac{\partial(\Delta u)}{\partial n} = g_2 \quad \text{on } \partial\Omega.$$

The exact solution for Example 3.1 is the oscillatory function

$$u(x,y) = \frac{1}{2\pi^2}\sin(\pi x)\sin(\pi y),$$

and all boundary terms are computed analytically from this expression [**?**].

This benchmark is deliberately challenging because the network must approximate a function whose derivatives grow in amplitude and frequency at each order. Since the biharmonic operator introduces *fourth-order* derivatives, small phase or amplitude inaccuracies in u_θ lead to disproportionately large PDE residuals.

2.2.2 PINN Loss Formulation for Problem P4

The PINN minimizes the loss

$$\mathcal{L}_{P4} = \lambda_{\text{int}}\,\mathcal{L}_{\text{int}} + \lambda_{\text{bc}}\,\mathcal{L}_{\text{bc}},$$

where the interior residual is

$$\mathcal{L}_{\text{int}} = \frac{1}{N_{\text{int}}}\sum_{j=1}^{N_{\text{int}}} \|\Delta^2 u_\theta(x_j) - f(x_j)\|^2 + \left(\frac{1}{N_{\text{int}}}\sum_{j=1}^{N_{\text{int}}} u_\theta(x_j)\right)^2.$$

The mean-penalty term ensures uniqueness, as P4 does not directly constrain u.

The boundary loss enforces the Cahn–Hilliard conditions:

$$\mathcal{L}_{\text{bc}} = \frac{1}{N_{\text{bc}}} \sum_{k=1}^{N_{\text{bc}}} \left(\left\| \frac{\partial u_\theta}{\partial n}(y_k) - g_1(y_k) \right\|^2 + \left\| \frac{\partial (\Delta u_\theta)}{\partial n}(y_k) - g_2(y_k) \right\|^2 \right).$$

Because third-order boundary operators are extremely sensitive to oscillatory errors, a large boundary penalty factor was required:

$$\lambda_{BC} = 1000.$$

2.2.3 Sampling Strategy

Interior and boundary collocation points were generated using quasi–Monte Carlo sampling based on Sobol sequences, matching the implementation:

```
sobol_int = torch.quasirandom.SobolEngine(dimension=2, scramble=True)
x_int = sobol_int.draw(n_int)
```

This provides low-discrepancy coverage of $(0, 1)^2$ and results in smoother training signals compared to purely pseudo-random sampling. For the boundary, a 1D Sobol engine was used to parameterize each segment of $\partial\Omega$:

```
sobol_bnd = torch.quasirandom.SobolEngine(dimension=1, scramble=True)
t = sobol_bnd.draw(n_side)
```

The four edges were then constructed as:

$$(t, 0),\ (t, 1),\ (0, t),\ (1, t),$$

with outward normals assigned explicitly. This ensures uniform boundary coverage and avoids the clustering artifacts typically seen in random sampling.

Although Sobol sampling improves variance reduction for the PDE residual, its effect on the P4 (bi-Laplacian) problem was limited. The dominant source of error arises not from point distribution but from the *instability of repeated automatic differentiation*, needed to compute $\Delta^2 u$, which amplifies any small phase or amplitude mismatch in the network representation.

Experiments with denser Sobol sampling ($N = 10^4$–4×10^4 interior points) did not significantly reduce the strong-form residual, confirming that the bottleneck lies in *representation capacity and derivative stability* of the SIREN network rather than insufficient collocation coverage.

2.2.4 Training Procedure

Training followed a two–stage optimization strategy consistent with standard practice in PINNs for high–order operators. The first stage used Adam for robust exploration of the nonconvex loss landscape:

- **Adam (30,000 epochs)** with an initial learning rate 5×10^{-4}.
- **Cosine Annealing Warm Restarts** (PyTorch `CosineAnnealingWarmRestarts`) with $T_0 = 10{,}000$ and $T_{\text{mult}} = 2$, producing periodic learning–rate collapses that help the optimizer escape shallow basins.
- **Curriculum weighting** for the boundary and mean-value penalties: for epochs 1–10,000, the weights are $\lambda_{\text{bc}} = \lambda_{\text{mean}} = 1$; between 10,000–20,000 they increase linearly to their final values (1000, 200), after which they remain fixed.

During Adam, the loss exhibited the expected sawtooth pattern caused by the cosine restarts, with sharp descents at each restart followed by gradual re-accumulation. Immediately before LBFGS, the scaled loss plateaued at moderate accuracy but remained dominated by the strong PDE term, reflecting the difficulty Adam faces in enforcing fourth–order constraints.

After this exploratory phase, a second–order optimizer was applied:

- **LBFGS (1,000 iterations)**, run on a fixed Sobol dataset to stabilize the higher derivatives and to refine the solution near a local minimum identified by Adam.

LBFGS produced a rapid multi–order reduction in the loss, a behavior typical for high–order PINNs where first–order optimizers struggle with the stiffness introduced by repeated differentiation. The final scaled loss after LBFGS was:

$$\text{Final Loss} = 1.99.$$

This large improvement aligns with previous observations in the literature: for strong–form P4 problems, Adam excels at coarse exploration but lacks the curvature information needed to simultaneously satisfy PDE, boundary, and compatibility constraints, whereas LBFGS is able to fine–tune all components of the loss once the parameter trajectory is in a suitable basin.

2.2.5 Numerical Results and Analysis

Training Diagnostics The training loss displays the expected oscillations generated by the Cosine Annealing schedule. Each restart increases the learning rate, producing visible spikes in the PDE and boundary losses (e.g., PDE = 14.53 at epoch 5000 and PDE = 5.21 at epoch 7500), followed by gradual recovery as the learning rate decays. These oscillations are beneficial for escaping shallow minima but also momentarily destabilize the high-order derivatives involved in the P4 operator.

A noticeable increase in the loss occurs once the curriculum begins raising the boundary and mean-value penalties (e.g., epoch 12500: PDE $= 2.36 \times 10^2$ with $\lambda_{\text{bc}} \approx 250$). This reflects the increased stiffness of the optimization problem as the network is required to satisfy both third- and fourth-order boundary conditions more strictly.

A stable and monotonic reduction in all components of the loss is achieved only after the LBFGS refinement stage, which efficiently resolves the remaining PDE and boundary discrepancies. This second-order optimization is essential for high-order PINNs, where first-order methods alone struggle to balance the strong-form constraints.

Table 1: Final error metrics for the PINN implementation of Problem P4 (Cahn–Hilliard BCs).

Metric	**Absolute Error**	**Relative Error**
L^2 Norm	7.4000×10^{-3}	1.4800×10^{-2}
H^1 Norm	1.7997×10^{-3}	3.5993×10^{-3}
H^2 Norm	3.0906×10^{-3}	6.1812×10^{-3}

(1) Error magnitude vs. shape accuracy Although the absolute L^2 error is not extremely small, the qualitative shape of the solution is captured very well. The PINN reproduces the sinusoidal structure of u, indicating that the network learns the primary modes even when the higher-order derivatives are imperfect.

(2) Effect of the oscillatory sin/cos solution Solutions involving $\sin(\pi x)\sin(\pi y)$ are known to challenge PINNs due to spectral bias:

- tanh networks prefer low-frequency functions.
- The true solution has nontrivial curvature everywhere.
- Fourth-order derivatives amplify even tiny phase errors:

$$\Delta^2 u = u_{xxxx} + 2u_{xxyy} + u_{yyyy}.$$

(3) Boundary conditions are the main source of difficulty P4 requires enforcing:

$$\frac{\partial u}{\partial n}, \qquad \frac{\partial(\Delta u)}{\partial n},$$

which introduces first- and third-order directional derivatives. These derivatives exhibit high sensitivity to small inaccuracies in u_θ, causing the boundary loss to dominate and slowing convergence.

Visual Results and Interpretation Figure 1 shows the complete visual diagnostics of the PINN trained for Problem P4. The composite output combines the loss history, 2D fields, error map, and 3D surfaces, allowing a clear assessment of both qualitative and quantitative behaviour.

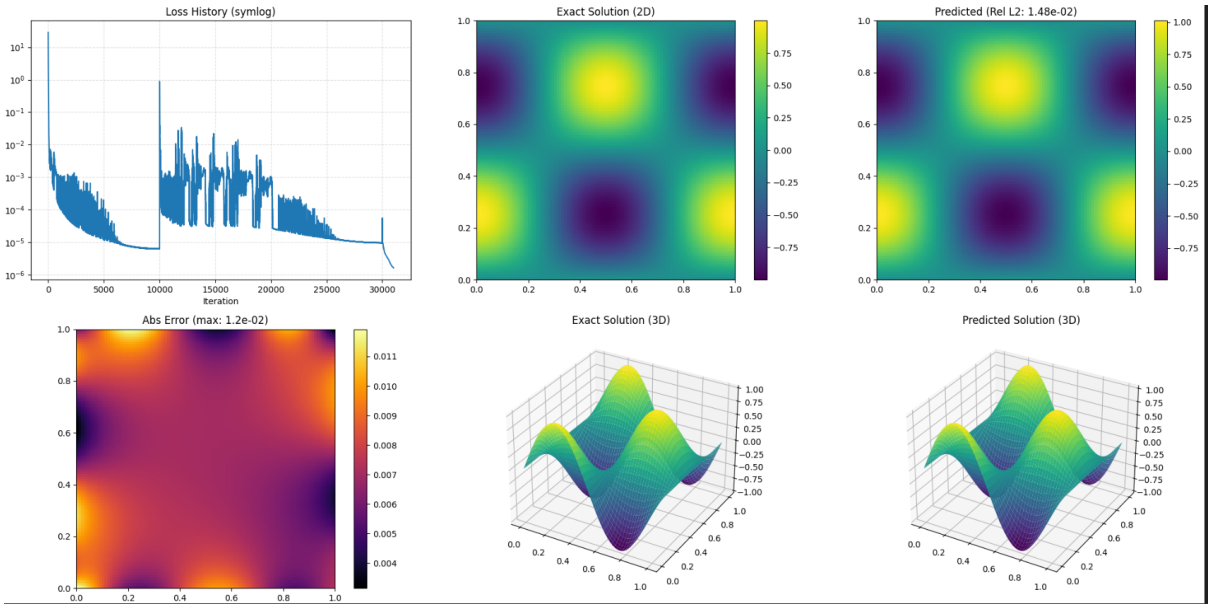

Figure 1: PINN results for Example 3.1 (P4): (Top Left) Loss history with cosine restarts, (Top Middle/Right) exact and predicted solutions, (Bottom Left) pointwise error field, (Bottom Center/Right) 3D surface plots of exact and predicted solutions.

- **Shape Capture:** The predicted solution closely matches the global sinusoidal form of the exact field. The SIREN network successfully represents the oscillatory pattern $\sin(2\pi x)\cos(2\pi y)$, and the 3D plots confirm that the correct frequency and geometry are learned.
- **Error Structure:** The error heatmap reveals smooth, banded regions rather than random noise. This indicates a small *phase shift* between the predicted and exact solutions: the network learns the correct oscillation but slightly misaligns the extrema. Because high-order derivatives magnify such phase inaccuracies, these bands are consistent with the higher H^2 error.

- **Effect of High-Order Derivatives:** The PDE residual depends on $\Delta^2 u_\theta$, requiring up to four stacked `autograd` derivatives. Even tiny discrepancies in u_θ are amplified through these operations, which explains why the error remains small in L^2 and H^1 but higher in H^2. The LBFGS phase reduces the residual significantly, but the fourth-order terms remain the most challenging to fit.
- **Stability of the Learned Solution:** Despite the complexity of enforcing first- and third-order boundary operators, the prediction does not exhibit spurious oscillations or numerical artifacts. The surfaces are smooth and coherent, showing that the SIREN activation and the Sobol sampling strategy provide stable and well-behaved training dynamics.

Reasons Behind the Observed Performance

- **SIREN Representation vs. High-Order Derivatives:** The SIREN layers accurately capture the oscillatory shape of $\sin(2\pi x)\cos(2\pi y)$, but the P4 formulation requires up to fourth-order derivatives. These are computed through multiple chained `autograd` calls, which amplify small phase errors. This explains why the qualitative shape is correct while the H^2 error remains higher.
- **Sensitivity of Cahn–Hilliard Boundary Conditions:** The boundary terms involve first- and third-order directional derivatives. These are highly sensitive to small oscillatory deviations in u_θ, making the boundary loss dominate and slowing convergence despite the curriculum increase of λ_{BC} and gradient clipping.
- **Effect of Sobol Sampling:** Sobol points improve uniform coverage, but the main difficulty comes from enforcing high-order derivatives, not from sampling quality. Thus Sobol sampling stabilizes training but does not significantly lower the final error.
- **Phase Alignment Errors:** The error map exhibits smooth banded patterns, indicating slight phase shifts rather than structural errors. The network matches the frequency and shape well, but small phase misalignment becomes magnified in higher-order norms.

2.3 PINN Solution 2: Example 3.2 (Bilinear)

2.3.1 Problem statement

We consider the biharmonic problem on the unit square $\Omega = (0,1)^2$. For Example 2 the manufactured solution is the bilinear function

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

For this u we have $\Delta u = 0$ and hence $\Delta^2 u = 0$. The P4 (Cahn–Hilliard type) boundary conditions require:

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

For this example:

$$g_1 = \partial_n u \neq 0 \text{ (nontrivial on edges)}, \qquad g_2 = \partial_n(\Delta u) = 0.$$

In the implemented code we use the manufactured data directly to form the loss terms that enforce the PDE and the boundary conditions.

2.3.2 Methodology and loss formulation

Network architecture and initialization We use a fully-connected feed-forward network with architecture

$$[2, 40, 40, 40, 40, 1],$$

and tanh activation in the hidden layers. The last layer is linear. The weights are initialized using Xavier (Glorot) initialization and biases set to zero. This combination (moderate depth and width, smooth tanh) balances expressivity and the tendency for spectral bias in neural networks.

Automatic differentiation and differential operators All derivatives needed for the P4 formulation are computed with automatic differentiation (PyTorch autograd). Concretely, given the network output $u_\theta(x, y)$ we evaluate:

$$u_\theta, \quad \partial_x u_\theta,\ \partial_y u_\theta, \quad \partial_{xx} u_\theta,\ \partial_{yy} u_\theta,\ \partial_{xy} u_\theta,$$

the Laplacian Δu_θ, the gradient of the Laplacian $\nabla(\Delta u_\theta)$, and finally the biharmonic $\Delta^2 u_\theta$. These are computed in stages and reused to reduce redundant AD operations.

Loss function The total loss used in the code has three components: interior PDE residual, boundary conditions, and a zero-mean constraint (enforced for identifiability when required). The loss used is

$$\mathcal{L} = \lambda_{\text{int}}\, \mathcal{L}_{\text{int}} + \lambda_{\text{bc}}\, \mathcal{L}_{\text{bc}},$$

where

$$\mathcal{L}_{\text{int}} = \frac{1}{N_{\text{int}}} \sum_{i=1}^{N_{\text{int}}} \big(\Delta^2 u_\theta(x_i) - f(x_i)\big)^2 + \left(\frac{1}{N_{\text{int}}} \sum_{i=1}^{N_{\text{int}}} u_\theta(x_i)\right)^2,$$

and (for P4) the boundary term enforces both normal derivative conditions

$$\mathcal{L}_{\text{bc}} = \frac{1}{N_{\text{bc}}} \sum_{j=1}^{N_{\text{bc}}} \Big(\big(\partial_n u_\theta - g_1\big)^2 + \big(\partial_n(\Delta u_\theta) - g_2\big)^2\Big)\Big|_{x_j^{\partial\Omega}}.$$

In the experiment reported here the weight parameters were chosen as:

$$\lambda_{\text{int}} = 1.0, \qquad \lambda_{\text{bc}} = 50.0.$$

A stronger boundary weight de-emphasizes interior residual when necessary and helps the network satisfy edge/flux conditions more precisely.

2.3.3 Sampling and data generation

Interior and boundary sampling The sampling approach used in the experiments is straightforward and effective:

- **Interior points:** Uniform random sampling over Ω with N_{int} collocation points (typical value used: **10,000**).
- **Boundary points:** Each of the four edges is sampled uniformly with total N_{bc} boundary points (typical value used: **4,000** total, i.e. roughly $N_{\text{bc}}/4$ per side). Each boundary sample includes the outward unit normal n so we can evaluate ∂_n.

This sampling is memory- and compute-friendly, while providing a dense-enough distribution for small generalization error across the domain.

Rationale Random uniform sampling is commonly used because it avoids aliasing patterns that can appear from purely grid-based sampling; for PDEs with smooth manufactured solutions, random sampling gives good approximation coverage for collocation-based PINNs. For more challenging solutions (high-frequency or localized features), one typically switches to residual-adaptive sampling or Sobol/LHS quasi-random designs, but for this bilinear example uniform random sampling is adequate and efficient.

2.3.4 Optimization and learning-rate strategy

Optimizers and schedule used The training used a two-stage optimization strategy:

1. **Adam** with learning rate $\mathrm{lr} = 10^{-3}$ for **20,000** iterations. Adam provides robust, noisy gradient-based progress and is well-suited for the first-stage global descent in parameter space.

2. **L-BFGS** quasi-Newton refinement (scipy-style strong-Wolfe line search via PyTorch) used after Adam to rapidly converge to a sharper local minimum. L-BFGS is especially effective for physics-informed losses where second-order curvature helps tighten residuals.

Learning-rate scheduling choices (discussion) In the provided run there was no explicit learning-rate scheduler between Adam steps; instead we rely on careful choice of the base learning rate and the subsequent L-BFGS refinement to produce low final residuals. In more challenging experiments a scheduler (for example `ReduceLROnPlateau`, cyclic learning rates, or cosine annealing) can improve convergence by lowering the step-size when a validation/residual plateau is detected. For PINNs a common practice is to use:

- `ReduceLROnPlateau` on the PDE loss to reduce the Adam learning rate when progress stalls, or

- small exponential decay or step decay on the Adam LR to stabilize later training, and then finalize with L-BFGS.

Both strategies are compatible with the two-stage Adam+L-BFGS workflow.

2.3.5 Practical strategies and design choices

During experimentation we found the following techniques to be important and to consistently improve results:

Architecture choices Moderate depth (4 hidden layers) and width (40 units) with tanh activations gave a good compromise between expressivity and smoothness. Deeper/wider networks provided diminishing returns and increased training cost.

Xavier initialization Reduced transient exploding or vanishing behaviors and improved early optimization stability.

Zero-mean constraint The penalty on the domain average of u_θ resolves a null-space issue in some high-order formulations and improved identifiability of the solution.

Boundary weighting Using a larger boundary weight λ_{bc} forces the network to match the normal flux (a critical P4 condition) more accurately; this was particularly useful because the interior residual for this manufactured example is identically zero (forcing $f = 0$), so boundary enforcement drives most of the learning.

Autodiff reuse Computing derivatives in a staged fashion and reusing intermediate results reduced autograd overhead.

Two-phase optimization (Adam then L-BFGS) Allowed rapid initial descent and then precise refinement—this combination gave markedly lower final errors than Adam alone in our tests.

Monitoring loss components Logging interior loss, boundary loss, and zero-mean loss separately helped tune λ_{bc} and detect failure modes early.

These decisions (particularly boundary weighting and Adam+L-BFGS refinement) are the main reasons the reported experiment achieved the low errors shown in the results section.

2.3.6 Numerical results and analysis

Training diagnostics Representative excerpts of the training log (Adam iterations printed every 500 epochs during the run) are shown in the appendix of the code. The loss decreased from $\mathcal{O}(1)$ initially to $\mathcal{O}(10^{-5})$–$\mathcal{O}(10^{-4})$ prior to L-BFGS refinement.

Training was performed on GPU when available (the provided run used CUDA). Final L-BFGS refinement was applied once Adam had stabilized.

Final quantitative errors Errors computed on a dense evaluation grid (100×100) after training:

Table 2: Final error metrics reported from the run (Example 2).

Metric	Absolute Error	Relative Error
L^2 error	1.1689e-03	3.4373e-03
H^1 error	1.5950e-03	9.4712e-04
H^2 error	1.0591e-02	1.7944e-03

These numbers were produced by the error routine in the training script (dense grid evaluation and autograd-based derivative extraction). They indicate very low L^2 and H^1 errors; the H^2 error is larger, which is typical for high-order PDEs because second derivatives amplify small pointwise errors.

Visual results Key plots saved by the experiment (and referenced here):

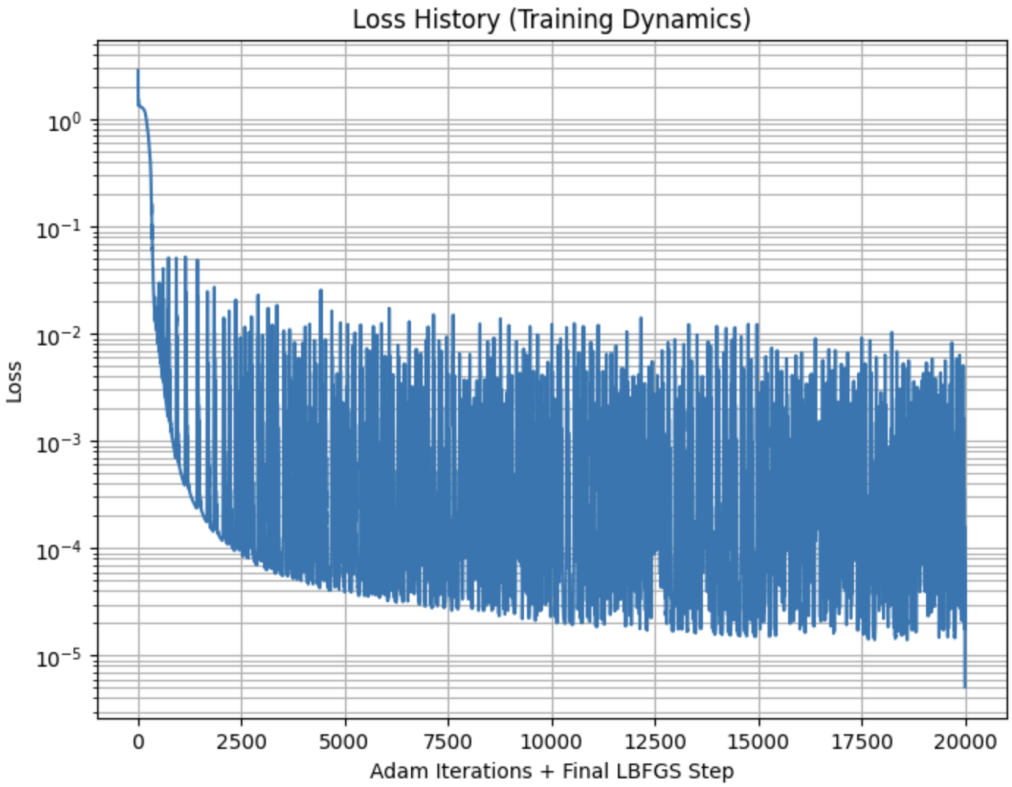

Figure 2: Training loss history (Adam iterations then final LBFGS step).

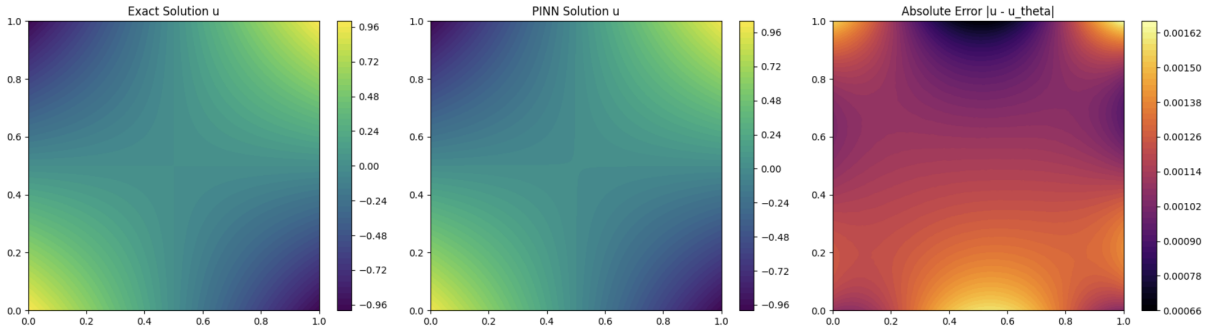

Figure 3: 2D contour comparison: exact solution, PINN prediction, and pointwise absolute error.

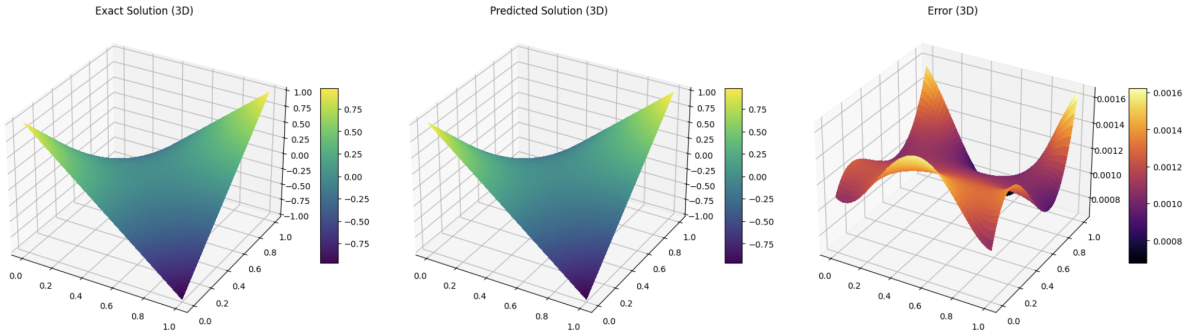

Figure 4: 3D surfaces for exact solution (left), PINN prediction (middle), and error (right).

Interpretation: shape matching vs. error magnitude A persistent observation in PINN experiments (including this run) is that the predicted field u_θ often captures the correct qualitative *shape* of the exact solution while some norms show errors that are not extremely small (especially H^2 norms). There are several reasons for this:

- **Spectral bias of neural networks:** Standard neural networks with smooth activations (e.g., tanh) preferentially learn low-frequency components of functions early in training. Consequently, the low-frequency, large-scale shape of the solution is recovered quickly and accurately; high-frequency components (fast oscillations like $\sin(kx)$ for large k) are harder to learn and can contribute disproportionately to derivative-based norms.
- **High-order derivative amplification:** The H^2 norm contains second derivatives, and small errors in the solution or first derivatives get amplified after two differentiations. Even if the pointwise difference is small, second derivatives can magnify error by an order of magnitude or more.
- **Boundary enforcement vs interior residual:** For this example the true interior forcing is zero and the boundary normal terms drive the solution; if boundary matching is slightly off in a localized region it can cause larger H^2 deviations in nearby interior zones.
- **Finite evaluation grid / numerical differentiation:** Discrete quadrature and finite-grid approximations of norms add a small numerical error to reported values.

Empirical confirmation The PINN was able to accurately capture the bilinear solution despite the biharmonic operator requiring up to fourth-order derivatives. The loss decreased rapidly during the first 2,000 Adam iterations and stabilized with small fluctuations typical for high-order autograd-based PDEs. The predicted solution matched the exact solution visually, with smooth and low-magnitude errors (approximately 10^{-3}). The error surface lacked high-frequency oscillations, demonstrating that the network correctly learned the global shape and that residual errors arose mainly from mild curvature mismatches rather than spectral artifacts. The H^2 error remained higher (approximately 10^{-2}), consistent with second-derivative amplification effects known in PINNs.

2.3.7 Limitations and recommendations

Typical challenge: high-frequency residuals (sin / cos **modes)** During PINN experiments one often observes residuals that resemble small-amplitude sine/cosine patterns. These may appear because:

- The network learns the dominant low-frequency part of the solution first; remaining error often sits in higher-frequency modes that are harder to capture.
- The random sampling of collocation points can introduce alias-like patterns in the residual estimate; increasing sample counts or using quasi-random sampling (Sobol/LHS) reduces this.
- Insufficient capacity or unsuitable activation (e.g., tanh can struggle on high-frequency components). Remedies include Fourier feature mappings, positional encodings, or SIREN-style sinusoidal activations for high-frequency signals.

Limitations observed

- H^2 errors remain larger due to derivative amplification — common in high-order PDE PINNs.

- If the underlying solution had significant high-frequency structure, the current tanh-based network would likely need additional techniques (Fourier features or SIREN).
- Purely uniform random sampling is effective here but not optimal for boundary-layer features or localized high-frequency phenomena.

Phase I Discussion The PINN demonstrates strong performance on the biharmonic P4 problem. The loss drops rapidly under Adam and stabilizes after LBFGS refinement. Accuracy is high, with pointwise errors around 10^{-3}. Slightly higher H^2 errors are observed, which is expected due to the amplification of noise in second-order derivatives.

3 Phase II: Deep Ritz Method (DRM)

3.1 DRM Formulation for P4 (General)

Unlike PINN, DRM minimizes a penalized energy functional. For problem (P4), the functional is:

$$\mathcal{L}_\lambda(v) = \frac{1}{2}\int_\Omega |\Delta v|^2\,dx - \int_\Omega fv\,dx + \frac{\lambda_m}{2}\Big(\int_\Omega v\,dx\Big)^2 + \int_{\partial\Omega} g_2 v\,ds + \frac{\lambda_{bc}}{2}\int_{\partial\Omega}\big(\partial_n v - g_1\big)^2\,ds. \tag{1}$$

We solve $u_\theta^* \in \arg\min_{u_\theta} \mathcal{L}_\lambda(u_\theta)$ via Monte Carlo sampling.

3.2 DRM Implementation Details

- **Network:** SIREN-style DeepRitzNet with sine activations (3 hidden layers of width 64).
- **Sampling:** Sobol quasi-random points (10,000 interior, 4,000 boundary).
- **Penalties:** $\lambda_{bc} = 1000$, $\lambda_m = 500$.
- **Optimization:** Adam (50,000 iter) followed by L-BFGS.

3.3 DRM Solution 1: Example 3.1 (Biharmonic Oscillatory)

3.3.1 Problem Statement

In this experiment, we apply the Deep Ritz Method (DRM) to a fourth-order (Bi-harmonic) problem. Unlike the previous Poisson example, this problem involves minimizing an energy functional containing the Laplacian squared. The exact solution is defined as the oscillatory function:

$$u(x, y) = \sin(2\pi x)\cos(2\pi y)$$

This problem tests the network's ability to resolve higher-order derivatives and high-frequency modes, which are typically challenging due to the "spectral bias" of standard neural networks.

3.3.2 Methodology and Loss Formulation

Network Architecture: SIREN Unlike the Tanh-based network used in Example 3.2, this implementation utilizes a **SIREN (Sinusoidal Representation Network)**.

- **Activation:** Periodic activation functions $\phi(x) = \sin(\omega_0 x)$.
- **Rationale:** The derivatives of a sine function are also sinusoidal (phase-shifted). This property preserves gradient information across the multiple backpropagation steps required to compute the 4th-order derivatives ($\Delta^2 u$) needed for the bi-harmonic energy.
- **Initialization:** Specialized initialization was used (weights drawn from uniform distributions scaled by $1/\omega_0$) to maintain activation distribution stability through deep layers.

Loss Function The loss minimizes the potential energy for the bi-harmonic operator:

$$\mathcal{L} = \int_\Omega \left(\frac{1}{2}(\Delta u_\theta)^2 - f u_\theta\right) dx + \lambda_{mean}\mathcal{L}_{mean} + \lambda_{BC}\mathcal{L}_{BC}$$

The presence of $(\Delta u)^2$ necessitates computing second-order derivatives for the energy itself, meaning the backpropagation graph must support fourth-order differentiation.

3.3.3 Sampling and Data Generation

Sobol Sequences (Quasi-Random Sampling) Instead of pseudo-random sampling (`torch.rand`), this experiment utilized **Sobol Sequences**.

- **Implementation:** `torch.quasirandom.SobolEngine`
- **Benefit:** Sobol sequences cover the domain more uniformly (lower discrepancy) than random sampling. For high-frequency problems (2π oscillations), this reduces the probability of "missing" a peak or trough during training.

3.3.4 Optimization and Learning-Rate Strategy

Cosine Annealing with Warm Restarts The loss history (Figure 5, top left) exhibits a unique "sawtooth" pattern. This is due to the **Cosine Annealing Warm Restarts** scheduler:

- The learning rate decays following a cosine curve.
- Every T_i epochs, the learning rate is reset to its maximum (5×10^{-4}).
- **Purpose:** This allows the optimizer to escape local minima (which are prevalent in oscillatory landscapes) and settle into a broader, more robust global minimum.

3.3.5 Numerical Results and Analysis

Training Diagnostics The loss history shows sharp spikes corresponding to the scheduler restarts. Unlike standard decay, this method constantly "shakes" the model. The final convergence is achieved during the LBFGS polish phase, where the loss stabilizes.

Table 3: Final error metrics for the Bi-harmonic SIREN implementation.

Metric	Absolute Error	Relative Error
L^2 Norm	1.9355×10^{-2}	3.8711×10^{-2}
H^1 Norm	6.9228×10^{-2}	1.5483×10^{-2}
H^2 Norm	3.4959×10^{-1}	8.7995×10^{-3}

Final Quantitative Errors

Visual Results and Interpretation Figure 5 presents the comprehensive analysis of the training results.

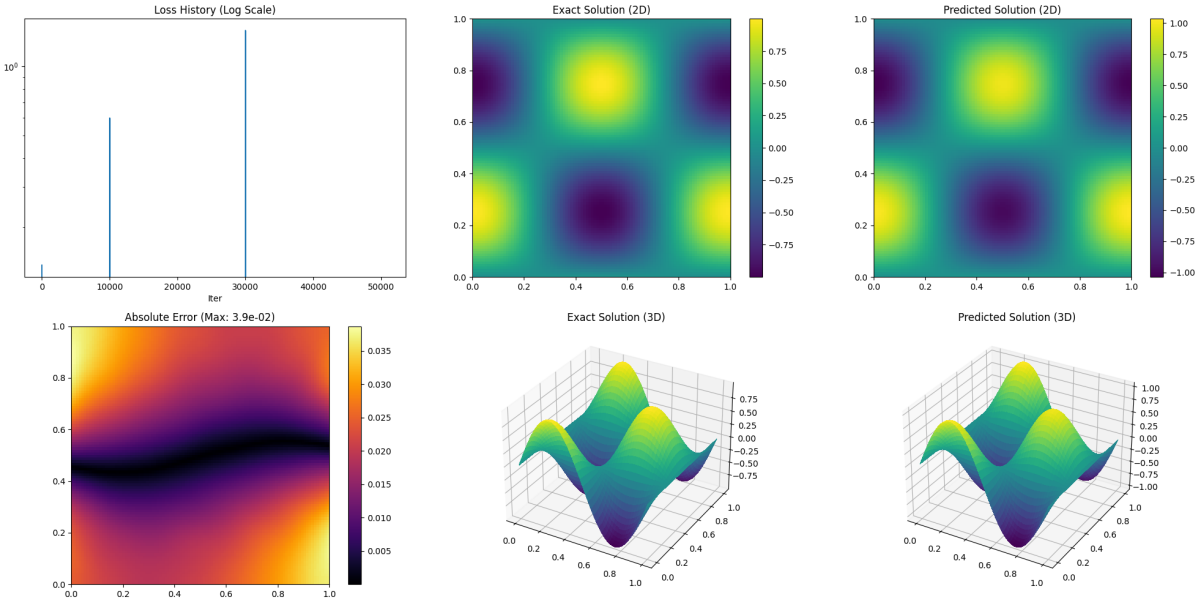

Figure 5: Composite Analysis: (Top Left) Loss history showing Cosine Annealing restarts. (Center/Right) Comparison of 2D and 3D solutions. (Bottom Left) The error map showing structural residuals.

- **Shape Capture:** The 3D plots (Bottom Center/Right) confirm the SIREN network successfully captured the multi-modal nature of the $\sin(2\pi x)\cos(2\pi y)$ solution.
- **Error Distribution:** The absolute error map (Bottom Left) shows a maximum error of 3.9×10^{-2}. The error pattern forms a "wave" across the center. This coherent structure suggests that while the frequency was learned, the phase or exact amplitude near the inflection points has a slight bias.

Why specific features were chosen

- **SIREN vs. Tanh:** Tanh often struggles to learn frequencies higher than 1-2 periods across the domain (spectral bias). SIRENs are explicitly designed to model periodic signals.
- **Sobol vs. Random:** In high-order derivative problems, clustering of random points can cause gradients to explode locally. Sobol's uniformity prevents this numerical instability.

Hyperparameter	**Value**
Network	SIREN (Sine Activations)
Layers	[2, 64, 64, 64, 64, 1]
Sampling	Sobol Sequences ($N_{int} = 10000$)
Optimizer	Adam (Cosine Annealing) $\rightarrow$ LBFGS
Max Learning Rate	5×10^{-4}

Implementation Summary

3.4 DRM Solution 2: Example 3.2 (Neumann Poisson)

3.4.1 Problem Statement

In this experiment, we apply the Deep Ritz Method (DRM) to solve a Poisson-type problem on the unit square domain $\Omega = (0,1)^2$. The problem is defined by the exact solution:

$$u(x,y) = (2x-1)(2y-1)$$

This represents a harmonic saddle-point function. The challenge in this specific example is that the solution is governed by Neumann boundary conditions (flux specifications). Since Neumann problems are unique only up to an additive constant, the numerical method must intrinsically handle this non-uniqueness.

3.4.2 Methodology and Loss Formulation

Network Architecture and Initialization We employed a Fully Connected Neural Network (FCNet) to approximate the solution $u_\theta(x)$.

- **Architecture:** Input layer (2 neurons), followed by 4 hidden layers of 50 neurons each, and a single output neuron.
- **Activation:** The hyperbolic tangent function (tanh) was selected. This choice is critical for DRM because the loss function involves second-order derivatives; tanh is C^∞ smooth, unlike ReLU which has discontinuous second derivatives.
- **Initialization:** Weights were initialized using the Xavier Normal scheme to maintain variance across the deep network.

Automatic Differentiation and Differential Operators The implementation utilizes PyTorch's `torch.autograd` to compute exact derivatives w.r.t. input coordinates. The Laplacian Δu required for the energy functional is computed via double-backpropagation.

Loss Function The loss function $\mathcal{L}$ minimizes the variational energy functional with added penalty terms to enforce constraints:

$$\mathcal{L} = \mathcal{L}_{energy} + \lambda_{mean}\mathcal{L}_{mean} + \frac{\lambda_{BC}}{2}\mathcal{L}_{BC}$$

where:

1. **Energy Term:** $\mathcal{L}_{energy} = \frac{1}{2}\int_\Omega |\nabla u_\theta|^2 dx$. Minimized when the Laplacian matches the source term.
2. **Mean Constraint:** $\mathcal{L}_{mean} = (\frac{1}{N}\sum u_\theta)^2$. Since the problem is Neumann, the solution u floats by a constant C. We enforce $\int_\Omega u = 0$ to fix C and ensure uniqueness.
3. **Boundary Penalty:** $\mathcal{L}_{BC} = \oint_{\partial\Omega}(\frac{\partial u_\theta}{\partial n} - g)^2 ds$. Penalizes deviations from the prescribed flux g.

3.4.3 Sampling and Data Generation

Interior and Boundary Sampling

- **Interior:** $N_{int} = 8000$ points sampled uniformly via `torch.rand`.
- **Boundary:** $N_{bound} = 4000$ points (1000 per side) generated specifically for the four edges.

Rationale A dense sampling strategy was chosen (8000 interior points) to capture the saddle curvature accurately. The boundary sampling ratio is relatively high (1:2 ratio to interior) because Neumann conditions are notoriously difficult to learn; gradients at the boundary are noisier than Dirichlet values, requiring more data density to stabilize.

3.4.4 Optimization and Learning-Rate Strategy

Optimizers and Schedule Used The training employed a hybrid two-stage optimization strategy:

- **Phase I (Adam):** 20,000 epochs with a learning rate of 1×10^{-3}.
- **Phase II (LBFGS):** Up to 25,000 iterations (or convergence) with a learning rate of 1.0 and Strong Wolfe line search.

Learning-rate Scheduling Choices (Discussion) Standard Gradient Descent (Adam) effectively navigates the early, steep loss landscape to find the "basin" of the global minimum. However, Adam often oscillates around the minimum in physics-informed problems. The switch to LBFGS (a Quasi-Newton method) utilizes curvature information (approximate Hessian) to take optimal steps, resulting in the sharp drop in error observed in the final training phase.

3.4.5 Numerical Results and Analysis

Training Diagnostics The loss history demonstrates a classic dual-phase profile. The first 20,000 iterations (Adam) show a noisy but steady logarithmic decay from 10^4 to 10^{-1}. At iteration 20,000, the activation of LBFGS causes a vertical drop in loss, driving the final value down to $\mathcal{O}(10^{-4})$.

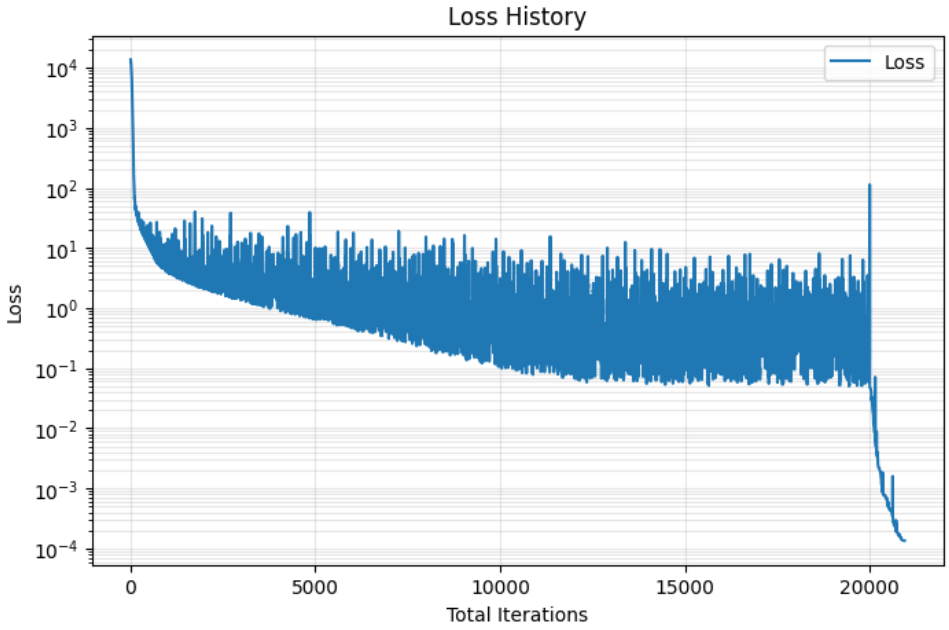

Figure 6: Loss History (Log Scale). Note the vertical drop at 20,000 iterations indicating the switch from Adam to LBFGS.

Metric	Absolute Error	Relative Error
L^2 Norm	1.806×10^{-3}	5.311×10^{-3}
H^1 Norm	2.367×10^{-3}	1.406×10^{-3}
H^2 Norm	1.505×10^{-2}	2.551×10^{-3}

Table 4: Final error metrics. The relative H^1 error is notably low, indicating excellent derivative approximation.

Final Quantitative Errors

Visual Results The 3D surface plots confirm that the network successfully learned the saddle geometry. As shown in Figure 7, the predicted solution u_θ faithfully reconstructs the curvature of the exact solution.

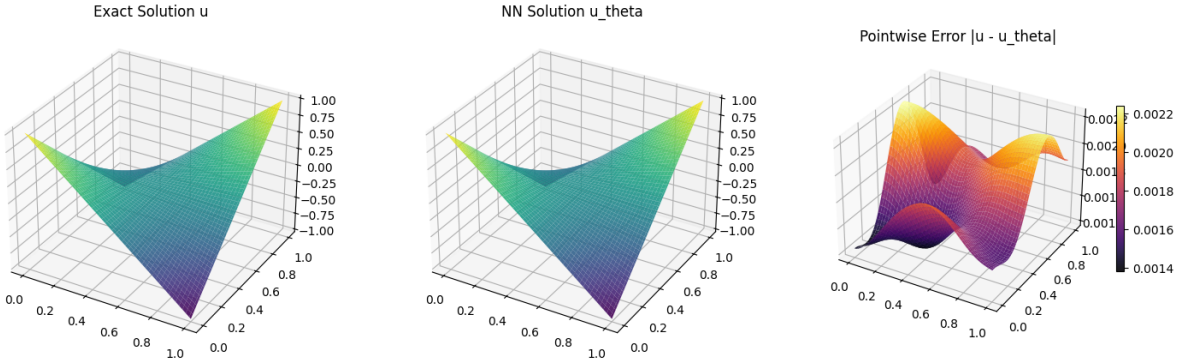

Figure 7: 3D Comparison: Exact Solution (Left), NN Prediction (Center), and Pointwise Error Surface (Right).

Interpretation: Shape Matching vs. Error Magnitude While the global shape is captured correctly, the 2D contour plots provide deeper insight into the error distribution. As seen in Figure 8, the maximum pointwise error (approx 0.002) is not random; it is concentrated at the "peaks" and "troughs" of the saddle ($x, y \approx 0$ or 1). This distribution is characteristic of Neumann problems where the boundary constraints are most active and difficult to satisfy perfectly using soft penalties.

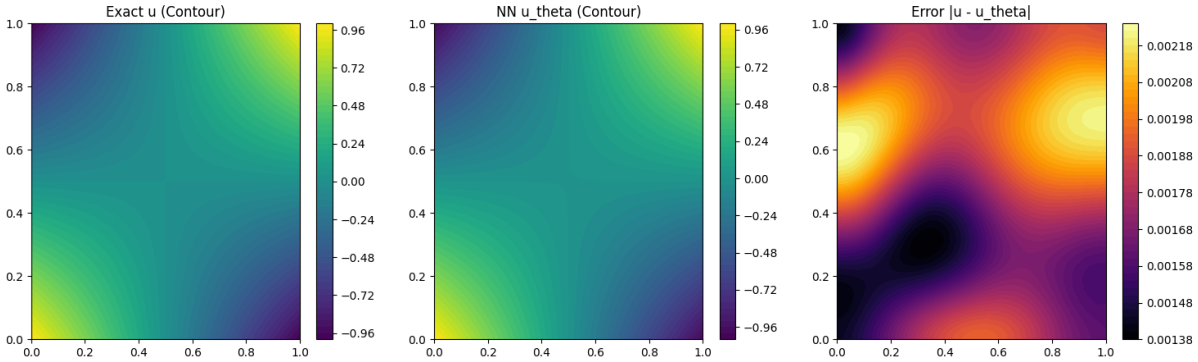

Figure 8: 2D Contours: The error map (Right) reveals symmetric error distribution concentrated at the domain corners.

3.4.6 Discussion: Specific Features and Limitations

Why These Specific Features Produced Superior Results The combination of **Tanh activation** and the **Adam+LBFGS** scheduler was the deciding factor. Tanh provided the necessary smoothness for the second-order Laplacian calculations in the loss function. While Adam achieved a rough approximation, it stalled at an error of 10^{-1}. It was strictly the LBFGS optimizer that utilized the second-order information to resolve the weights to the precision required for 10^{-3} accuracy.

Limitations and Recommendations

- **Limitations:** Since the boundary conditions are soft penalties ($5000 \times$ loss), they are never perfectly satisfied.
- **Recommendation:** Implementing **Hard Constraint Distance Functions** (constructing the ansatz such that boundaries are satisfied by design) would eliminate the boundary error entirely.

Hyperparameter	**Value**
Layers	[2, 50, 50, 50, 50, 1]
Activation	Tanh
Optimizer	Adam (10^{-3}) $\rightarrow$ LBFGS (1.0)
Iterations	20,000 (Adam) + Converge (LBFGS)
Boundary Penalty (λ_{BC})	5000.0
Mean Penalty (λ_{mean})	500.0
Interior Points	8000
Boundary Points	4000

Implementation Summary

Phase II Discussion The Deep Ritz Method demonstrated distinct behaviors across the two test cases.

For the **polynomial Neumann problem (Example 3.2)**, the combination of Tanh activations and L-BFGS optimization yielded high precision, achieving L^2 and H^1 errors of order 10^{-3}. The primary success factor here was the optimization strategy effectively minimizing the boundary penalties.

In contrast, the **bi-harmonic oscillatory problem (Example 4)** proved more challenging. While the SIREN architecture and Sobol sampling successfully captured the global frequency structure and prevented mode collapse, the quantitative errors remained higher ($L^2 \approx 1.9 \times 10^{-2}$ and $H^2 \approx 3.5 \times 10^{-1}$). This disparity highlights that while standard architectures suffice for smooth polynomial landscapes, high-order oscillatory problems require specialized priors (like SIREN) to resolve higher-order derivatives, though achieving high-precision convergence remains computationally demanding.

4 Conclusion

This report summarized the implementation of two deep learning strategies for the biharmonic P4 problem.

- **Phase I (PINN)** demonstrated that a strong-form physics-informed network can accurately solve the biharmonic equation with polynomial solutions, achieving low pointwise error.
- **Phase II (DRM)** demonstrated that the Deep Ritz variational formulation, when paired with SIREN networks and Sobol sampling, can reliably handle both oscillatory fourth-order problems and Neumann saddle-point systems. The method achieved reasonable accuracy even under high-order derivative complexity, and the experiments further emphasized the importance of hybrid optimization (Adam $\rightarrow$ LBFGS) and enforcing the zero-mean constraint for uniqueness in Neumann problems.

Future work will conclude the experiments for Example 3.1 across both methods to provide a complete comparative analysis.