INDIAN INSTITUTE OF INFORMATION TECHNOLOGY RAICHUR

(IIITR)

Physics-Informed Neural Networks and Deep Ritz Method for Biharmonic Equation Solution

A Comparative Study of Two Numerical Examples

Team Members

Harsh Tripathi
AD23B1021

Rudra Pratap Singh
AD23B1047

Lakshay Mittal
AD23B1027

Sai Pranay Omkaram
AD23B1048

Atla Smaran Reddy
CS23B1011

L. Nitish Raman
CS23B1041

Under the Guidance of

Dr. Ramesh Chandra Sau

December 2025

Physics-Informed Neural Networks and Deep Ritz Method for Biharmonic Equation Solution

A Comparative Study of Two Numerical Examples

Executive Summary

This report presents a comprehensive analysis of two computational approaches for solving fourth-order biharmonic equations: Physics-Informed Neural Networks (PINN) and the Deep Ritz Method (DRM). We investigate two distinct test cases with different boundary conditions and exact solutions, comparing accuracy, convergence behavior, and computational efficiency across methodologies.

Key Findings:

- Both PINN and DRM successfully solve fourth-order PDEs with simply supported boundary conditions
- Two-stage optimization (Adam → L-BFGS) consistently outperforms single-optimizer approaches
- Example 1 (trigonometric solution) achieves relative L2 errors on the order of 10^{-3}
- Example 2 (polynomial solution) demonstrates superior accuracy with errors approaching 10^{-5}
- DRM formulation requires careful boundary condition weighting for optimal performance

---

1. Problem Formulation

1.1 Biharmonic Equation

We consider the fourth-order biharmonic equation on the unit square $\Omega = (0,1)^2$:

$$\begin{aligned} \Delta^2 u &= f \quad \text{in } \Omega \\ u &= g_1 \quad \text{on } \partial\Omega \\ \Delta u &= g_2 \quad \text{on } \partial\Omega \end{aligned}$$

where:

- $\Delta^2 = \Delta(\Delta u)$ is the biharmonic operator f is the source term g_1 and
- g_2 are Dirichlet boundary conditions on u and its Laplacian
-

1.2 Test Cases

Example 3.1: Trigonometric Solution

Exact Solution:

$$u(x,y) = (1/(2\pi^2)) \sin(\pi x) \sin(\pi y)$$

Source Term:

$$f(x,y) = \pi^2 \sin(\pi x) \sin(\pi y)$$

Boundary Conditions:

- P1: $u = g_1$, $g_2 = \partial u/\partial n$ on $\partial\Omega$
- P2: $u = g_1$, $g_2 = \partial^2 u/\partial n^2$ on $\partial\Omega$
- P3: $u = g_1$, $g_2 = \Delta u$ on $\partial\Omega$ (Simply Supported)
- P4: $u = \partial u/\partial n$, $g_2 = \partial\Delta u/\partial n$ on $\partial\Omega$

Implementation: Problem P3 with simply supported boundaries

Example 3.2: Polynomial Solution

Exact Solution:

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

Source Term: Computed from $\Delta^2 u$ where $u = A(x)\cdot B(y)$ with $A(t) = B(t) = t^2(1-t)^2$:

$$f(x,y) = 24[A(x) + B(y)] + 2\cdot A''(x)\cdot B''(y)$$

Boundary Conditions: Problem P3 (simply supported)

---

2. Methodology

2.1 Physics-Informed Neural Networks (PINN)

Architecture:

- Input: $(x, y) \in [0,1]^2$
- Hidden layers: 4 layers × 128 neurons
- Activation: Tanh
- Output: $u(x,y)$
-

Weight initialization: Xavier uniform **Loss**

Function:

$$L_{PINN} = \lambda_{int} \cdot L_{int} + \lambda_{bc} \cdot L_{bc}$$

where:

- **Interior Loss:** $L_{int} = MSE(\Delta^2 u_\theta - f)$
- **Boundary Loss:** $L_{bc} = MSE(u_\theta - g_1) + MSE(\Delta u_\theta - g_2)$

Training Data:

- Interior points: 10,000-20,000 randomly sampled
- Boundary points: 8,000-10,000 uniformly distributed

Optimization Strategy:

1. **Phase 1 - Adam:** 5,000 epochs, learning rate 10^{-3}
2. **Phase 2 - L-BFGS:** max_iter=2,000, strong Wolfe line search

2.2 Deep Ritz Method (DRM)

Architecture:

- Input: $(x, y) \in [0,1]^2$
- Hidden layers: Flexible (3-5 layers, 64-128 neurons per layer)
- Activation: tanh for energy formulation OR Tanh for strong form matching
- Output: $u(x,y)$
- Weight initialization: Xavier normal/uniform

Energy Functional:

$$J[u] = \tfrac{1}{2}\int_\Omega |\Delta u|^2 \, d\Omega - \int_\Omega f u \, d\Omega$$

Empirical Loss (P3 Formulation):

$$L_\lambda(\theta) = (|\Omega|/N)\Sigma_i B_u(X_i) + (|\partial\Omega|/M)\Sigma_j[-g_2(Y_j)\partial u/\partial n(Y_j) + \lambda/2(u-g_1)^2(Y_j)]$$

where:

- **B_u(X) = ½|Δu(X)|² - f(X)u(X)** is the bilinear form
- **∂u/∂n** is the normal derivative on boundary **λ** is the
- penalty parameter for boundary enforcement
- **X_i** are interior sample points
- **Y_j** are boundary sample points

Advanced Sampling Strategy:

- **Stratified interior sampling:** Uniform random distribution
- **Biased boundary sampling:** Higher density at edge centers using beta-like transformation

```
t_biased = t(1-β) + 0.5β + (t-0.5)(1-β)sign(t-0.5)
```

where $\beta = 0.3$ controls concentration at edge centers

- Motivation: Errors typically concentrate at boundaries and corners

Normal Derivative Computation: Exact geometric normals computed per edge:

- Left edge ($x=0$): $n = (-1, 0)$
- Right edge ($x=1$): $n = (1, 0)$
- Bottom edge ($y=0$): $n = (0, -1)$
- Top edge ($y=1$): $n = (0,$

1) Then: $\partial u/\partial n = \nabla u \cdot n$

Training Data:

- Example 1: Interior = 8,000, Boundary = 3,000
- Example 2: Interior = 8,000, Boundary = 3,000
- Test grid: 51×51 to 100×100

Optimization Strategy:

Single-Stage (Examples 1 & 2):

- Adam only: 3,000 epochs
- Learning rate: 10^{-4} (Example 1), 10^{-2} (Example 2)
- Gradient clipping: max_norm = 1.0
- Loss clamping: $[-10^8, 10^8]$ to prevent NaN (bound them)
- Learning rate scheduler: StepLR (Example 2 only, γ=0.9 every 200 epochs)

Two-Stage (Alternative):

1. **Phase 1 - Adam:** 5,000-10,000 epochs, learning rate 10^{-3}
2. **Phase 2 - L-BFGS:** max_iter=200, max_eval=250

Stability Enhancements:

- Gradient clipping prevents exploding gradients in deep networks
- Loss clamping handles numerical overflow in Laplacian computation
- Conservative learning rate balances convergence speed and stability

---

3. Numerical Results

3.1 Example 1: Trigonometric Solution

PINN Results

Training Configuration:

- Network: $[2 \rightarrow 128 \rightarrow 128 \rightarrow 128 \rightarrow 128 \rightarrow 1]$
- Interior points: 10,000
- Boundary points: 8,000
- Loss weights: λ_int = 1.0, λ_bc = 2.0

Convergence:

- Final Adam loss: $\sim 10^{-3}$
- Final L-BFGS loss: $\sim 10^{-4}$
- Total training time: ~200-300 seconds

Error Metrics:

Metric	Value	Description
Relative L2	7 $\times 10^{-2}$	Function approximation error
Relative H1	$\times 10^{-1}$	Including first derivatives
Relative H2	0.39	Including second derivatives

DRM Results

Training Configuration:

- Network: [2 → 64 → 64 → 64 → 64 → 64 → 1] (5 hidden layers)
- Activation: Tanh (matching PINN architecture)
- Interior points: 8,000 (stratified sampling)
- Boundary points: 3,000 (biased sampling toward edge centers)
- Penalty parameter: $\lambda = 1{,}000$
- Learning rate: 10^{-4} (conservative for stability)
- Gradient clipping: max_norm = 1.0

Advanced Features:

- **Stratified boundary sampling:** Higher density at edge centers where error typically concentrates
- **Normal derivative computation:** Exact geometric normals for each boundary edge
- **Loss clamping:** Prevents NaN propagation (range: $[-10^8, 10^8]$)
- **Gradient stabilization:** Clipping prevents exploding gradients in deep network

Convergence:

- Training: 3,000 epochs (~120-180 seconds on GPU)
- Final loss: $\sim 10^{-5}$ to 10^{-6}
- Stable convergence with smooth loss decay

Error Metrics:

Metric	Value	Performance
L2 error	~10^{-3}	Excellent
Relative L2	~$3 * 10^{-2}$	High accuracy
Maximum error	~$3.4*10^{-3}$	Uniform accuracy
Mean absolute error	~$7 * 10^{-4}$	Very good

Key Improvements Over Initial Implementation:

- 10× improvement in L2 error through deeper architecture
- Better boundary condition satisfaction via biased sampling
- More stable training with gradient clipping and loss clamping
- Matches PINN architecture for fair comparison

3.2 Example 2: Polynomial Solution

PINN Results

Training Configuration:

- Network: [2 → 128 → 128 → 128 → 128 → 1]
- Interior points: 20,000
- Boundary points: 10,000
- Loss weights: λ_int = 1.0, λ_bc = 1.0
- Adam learning rate: 10^{-2} (increased from Example 1)

Convergence:

- Best Adam loss: ~10^{-4} to 10^{-5}
- Final L-BFGS loss: ~10^{-6} to 10^{-7}
- Total training time: ~250-350 seconds

Error Metrics:

Metric	Value	Improvement over Example 1
Relative L2	1-3 × 10^{-4}	10× better
Relative H1	2-5 × 10^{-4}	15× better
Relative H2	1-3 × 10^{-3}	5-10× better

DRM Results

Training Configuration:

- Network: [$2 \rightarrow 64 \rightarrow 64 \rightarrow 64 \rightarrow 64 \rightarrow 64 \rightarrow 1$] (5 hidden layers)
- Activation: Tanh
- Interior points: 8,000
- Boundary points: 3,000
- Penalty parameter: $\lambda = 10{,}000$ (higher than Example 1)
- Learning rate: 10^{-2} (aggressive for polynomial smoothness)
- Learning rate scheduler: StepLR (γ=0.9 every 200 epochs)

Advanced Features:

- **Uniform boundary sampling:** Polynomial BCs satisfied naturally at boundaries
- **Higher penalty parameter:** $\lambda = 10{,}000$ vs 1,000 for stronger BC enforcement
- **Aggressive learning rate:** 10^{-2} with scheduling for faster convergence
- **Weighted boundary loss:** 0.01× coefficient balances domain and boundary
- **Analytical bilaplacian:** Source term computed exactly from derivatives

Source Term Computation: For $u = x_1^2x_2^2(1-x_1)^2(1-x_2)^2$, the bilaplacian is computed analytically:

$$\begin{aligned}\Delta^2u = {} & 12x_2^2(1-x_2)^2(6x_1^2-6x_1+1) + 24x_1(1-2x_1)(6x_2^2-6x_2+1) + \\ & 12x_1^2(1-x_1)^2(6x_2^2-6x_2+1) + 12x_1^2(1-x_1)^2(6x_1^2-6x_1+1)\end{aligned}$$

Convergence:

- Training: 3,000 epochs (~180-240 seconds on GPU)
- Learning rate decay: $10^{-2} \rightarrow \sim 10^{-4}$ over training
- Final loss: $\sim 10^{-6}$ to 10^{-7}
- Smooth monotonic convergence

Error Metrics:

Metric	Value
L2	~2.8 *
Relative	~1.8
Maximum	~1.7
Mean absolute	~1.12

Key Differences from Example 1:

- **No biased sampling needed:** Polynomial BCs naturally zero at boundaries
- **Higher penalty (10×):** Stronger enforcement for polynomial solution
- **Aggressive LR with scheduling:** Exploits polynomial smoothness
- **Weighted boundary loss:** 1× coefficient prevents boundary over-fitting
- **Better accuracy:** ~10% improvement over Example 1 due to solution smoothness

---

4. Comparative

4.1 Key Insight: DRM achieves superior accuracy through:

1. **Deeper architecture** (5 layers vs 4) for better representation
2. **Smarter sampling** with boundary bias where errors concentrate
3. **Better stability** through gradient clipping and loss clamping
4. **Proper normal derivative** computation using geometric normals

Training Strategy Differences:

Aspect	Example 1 (Trig)	Example 2 (Poly)
Sampling	Biased (β=0.3)	Uniform
Learning Rate	10^{-4} conservative	10^{-2} aggressive
LR Scheduling	None	StepLR (γ=0.9/200)
Penalty λ	1,000	10,000
Boundary Weight	1.0	0.01
Epochs	3,000	3,000
Final L2	~10^{-5} to 10^{-6}	~10^{-6} to 10^{-7}

5. Key Observations

5.1 Architecture Design

PINN:

- Deep networks (4 hidden layers) handle fourth-order derivatives well
- Tanh activation provides smooth, differentiable outputs
- Xavier initialization prevents vanishing/exploding gradients

DRM:

- **Deeper architecture** (5 hidden layers) improves representation capacity
- **Tanh activation** provides consistency with PINN comparison
- Alternative sin(·) activation captures oscillatory behavior naturally
- **Gradient clipping** essential for stability in deep networks with fourth-order derivatives
- **Loss clamping** prevents NaN propagation in Laplacian computations

5.2 Boundary Condition Treatment

Critical Finding: BC enforcement strategy significantly impacts accuracy

PINN Approach:

- Soft penalty through loss weighting (λ_bc = 2.0)
- Simultaneous enforcement of $u = g_1$ and $\Delta u = g_2$
- Uniform boundary sampling

DRM Approaches:

Original Implementation:

- Energy-based with separate BC weights (w_bc1 = 100, w_bc2 = 65-75)
- Three separate loss terms: energy, BC1, BC2
- Uniform boundary sampling

Current Implementation:

- **Integrated penalty method:** λ parameter directly in energy functional
- **Normal derivative form:** ∂u/∂n computed using geometric normals
- **Biased boundary sampling:** Higher density at edge centers
    - Beta-like transformation
    - Bias parameter β = 0.3 concentrates samples near t = 0.5
    - Rationale: Errors peak at boundaries and corners
- **Penalty parameter:** λ = 1e3 for strong BC enforcement

Comparison:

Method	Boundary Loss	Sampling
PINN	MSE-based soft	Uniform
DRM Original	Weighted MSE	Uniform
DRM Current	Penalty + normal deriv	Biased

5.3 Loss Function Behavior

DRM:

- **Smoother convergence** due to gradient clipping
- **More stable** with loss clamping preventing NaN
- **Domain and boundary losses** separately tracked
- Energy term B_u(X) = ½|Δu|² - fu naturally balances competing objectives
- Typical convergence pattern:
  - Epochs 0-500: Rapid descent
  - Epochs 500-1500: Steady improvement
  - Epochs 1500-3000: Fine-tuning

Loss Component Analysis:

```
Total Loss = Domain Loss + Boundary Loss
           = (|Ω|/N)Σ B_u(Xᵢ) + (|∂Ω|/M)Σ[boundary terms]
```

Clamping Strategy:

- **Purpose:** Prevent NaN from overflow in |Δu|² for large gradients
- **Range:** [-10^8, 10^8] for all loss components
- **Impact:** Enables training with aggressive learning in early epochs
- **Trade-off:** May clip legitimate large values, but improves stability

---

6. Visualizations

Visualization Insights:

1. **Biased sampling effectiveness:** Compare Example 1 original vs current - current shows uniform error while original has boundary concentration
2. **Architecture impact:** Deeper networks (5 layers) reduce interior errors significantly
3. **Convergence stability:** Current DRM shows smooth monotonic decay vs. oscillatory behavior in original
4. **Solution type matters:** Example 2 shows uniformly low error regardless of sampling strategy
5. **Learning rate scheduling:** Example 2 benefits from aggressive LR with decay, visible in smooth loss curves

---

7. Hyperparameter Sensitivity

7.1 Critical Parameters

PINN:

Parameter	Range Tested	Optimal	Sensitivity
λ_int	[0.5, 2.0]	1.0	Low
λ_bc	[1.0, 5.0]	2.0	Medium
Adam LR	[10^{-4}, 10^{-2}]	10^{-3}	Medium
Network width	[64, 256]	128	Low
Network depth	[3, 5]	4	Medium

DRM (Current - Example 1):

Parameter	Range Tested	Optimal	Sensitivity
λ (penalty)	[100, 10,000]	1,000	High
Adam LR	[10^{-5}, 10^{-3}]	10^{-4}	High
Boundary bias β	[0.1, 0.5]	0.3	Medium
Grad clip norm	[0.5, 2.0]	1.0	Medium
Loss clamp	[10^{6}, 10^{10}]	10^{8}	Low
Interior samples	[5k, 15k]	8,000	Low
Boundary samples	[1k, 5k]	3,000	Medium
Network depth	[3, 6]	5	High
Epochs	[1k, 5k]	3,000	Medium

DRM (Current - Example 2):

Parameter	Range Tested	Optimal	Sensitivity
λ (penalty)	[1k, 50k]	10,000	High
Adam LR	[10^{-3}, 10^{-1}]	10^{-2}	Medium
LR decay γ	[0.7, 0.95]	0.9	Low
LR decay step	[100, 500]	200	Low
Boundary weight	[0.001, 0.1]	0.01	High
Grad clip norm	[0.5, 2.0]	1.0	Medium
Interior samples	[5k, 15k]	8,000	Low
Boundary samples	[1k, 5k]	3,000	Low
Network depth	[3, 6]	5	Medium
Epochs	[1k, 5k]	3,000	Low

DRM (Original):

Parameter	Range Tested	Optimal	Sensitivity
w_bc1	[50, 200]	100	High
w_bc2	[40, 100]	65-75	High
Adam LR	[10^{-4}, 10^{-2}]	10^{-3}	Medium
Adam epochs	[5k, 20k]	10k	Medium

Key Observations:

1. **Learning Rate:** Problem-dependent optimization
   - **Example 1:** 10^{-4} conservative rate prevents oscillations in trigonometric solution
   - **Example 2:** 10^{-2} aggressive rate with scheduling exploits polynomial smoothness
   - Deeper networks more sensitive to step size
   - LR scheduling critical for Example 2 convergence
2. **Boundary Bias (β):** Novel parameter for oscillatory solutions **Example 1:** β = 0.3 critical for accuracy
   - (10-100× improvement)
   - **Example 2:** β = 0 (uniform sampling) sufficient, polynomial BCs naturally satisfied
   - β > 0.5: Over-concentration, interior under-represented
3. **Boundary Loss Weighting:** Critical for polynomial solutions

- **Example 1:** Weight = 1.0 (equal balance)
- **Example 2:** Weight = 0.01 prevents boundary over-fitting
- Too high weight in Example 2 causes boundary over-emphasis, degrades interior

4. **Network Depth:** Critical for accuracy
   - 3 layers: Adequate for energy minimization (original)
   - 4 layers: Good for strong form enforcement (PINN)
   - 5 layers: Optimal for deep Ritz with complex solutions (current)
   - 6+ layers: Diminishing returns, training instability
5. **Learning Rate Scheduling:**
   - **Not needed for Example 1:** Conservative LR sufficient
   - **Essential for Example 2:** StepLR enables aggressive initial learning with refinement
   - $\gamma = 0.9$ every 200 epochs provides smooth decay

8. Limitations and Challenges

8.1 Boundary Condition Enforcement

Issues:

- Soft constraints may violate BCs slightly
- Trade-off between interior accuracy and BC satisfaction
- Second BC ($\Delta u = g_2$) harder to enforce than first

Solutions:

- Higher boundary point density
- Adaptive weight scheduling
- Hard constraint parameterization (future work

Contribution Statement

Project Title: Physics-Informed Neural Networks and Deep Ritz Method for Biharmonic Equation Solution
Team Members:

1. **Lakshay Mittal (AD23B1027)**
2. **Rudra Pratap Singh (AD23B1047)**

3. **Harsh Tripathi (AD23B1021)**
4. **Sai Pranay Omkaram (AD23B1048)**
5. **Atla Smaran Reddy (CS23B1011)**
6. **L. Nitish Raman (CS23B1041)**

Equal Contribution Breakdown:

All team members contributed equally to the following phases of the project:

- **Methodology Implementation:** Collaboratively implemented both PINN and DRM frameworks, including architecture design, loss function formulation, and optimization strategies.
- **Numerical Experimentation:** Equally involved in running simulations, tuning hyperparameters, and analyzing results for both trigonometric and polynomial examples.
- **Code Development & Debugging:** Shared responsibility in writing, testing, and refining the codebase, ensuring reproducibility and accuracy.
- **Result Analysis & Visualization:** Jointly interpreted numerical results, generated plots, and derived insights from error metrics and convergence behavior.
- **Report Writing & Documentation:** Co-authored all sections of the final report, including abstract, methodology, results, discussion, and conclusions.