

Contents

DeePoly - High-Order Accuracy Neural Network Framework for Function Approximation and PDE Solving	1
Introduction	1
Key Features	1
Version Information	2
Usage	2
Installation Requirements	2
Test Cases & Results	2
Function Approximation Cases	2
PDE Solving Cases	7
Performance Summary	8

DeePoly - High-Order Accuracy Neural Network Framework for Function Approximation and PDE Solving



Introduction

DeePoly is a novel general-purpose platform for function approximation and equation solving algorithms. Core theorem and algorithm: DeePoly: A High-Order Accuracy and Efficiency Scientific Machine Learning Framework

Key Features

- **Universal PDE Solver:** Fit for all kinds of PDE.
- **Mesh-Free:** Sampling points can be randomly generated with no logical relationships, suitable for complex geometries.
- **High Accuracy:** Achieves high-order convergence.
- **Scheme-Free:** Handles derivative relationships using automatic differentiation.
- **Efficient:** Computational efficiency comparable to traditional finite difference methods.
- **GPU Accelerated:** Supports CPU parallelism and GPU acceleration.
- **Applicable to Complex and Discontinuous Problems:** Accurately approximates discontinuous and high-gradient functions.
- **Suitable for Inverse Problems:** Solves inverse problems with higher accuracy than PINNs.

Version Information

Previous version: v0.1 (Beta). The `cases` directory includes both function approximation and PDE solving test cases. The core algorithm in `src` contains all derivative computation code with comprehensive testing.

Current version: v0.2: - High-accuracy solving for arbitrary-dimensional linear PDEs. - Auto code of the PDE. - English-commented version. - Other corrections of output and visualize.

Usage

Develop your own problems in the `cases` directory, including data generation, output, and configuration files.

```
# Run function fitting example
python src/main_solver.py --case_path cases/func_fitting_cases/case_2d

# Run PDE solving example
python src/main_solver.py --case_path cases/linear_pde_cases/poisson_2d_sinpixsinpiy
# First time run, you need set auto-code=true, and then rerun it.
```

Installation Requirements

- Python
- CUDA
- torch

Test Cases & Results

The following test cases demonstrate DeePoly's capabilities across function approximation and PDE solving:

Function Approximation Cases

1. 1D Sine Function Case Directory: `func_fitting_cases/test_sin`

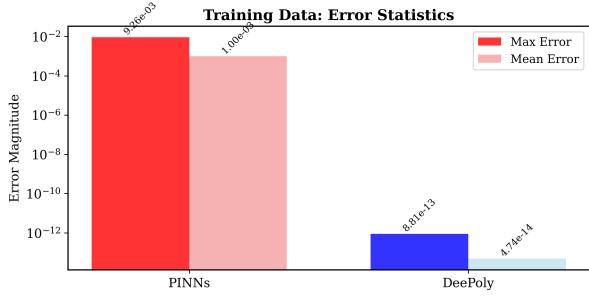
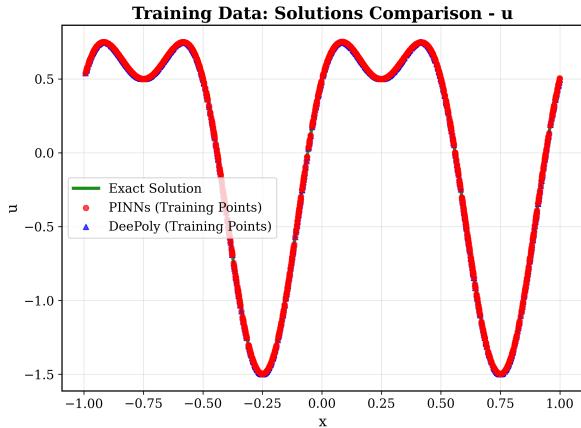
Target Function: $f(x) = \sin(2x) + 0.5\cos(4x)$

Domain: $[0, 1]$ (1D)

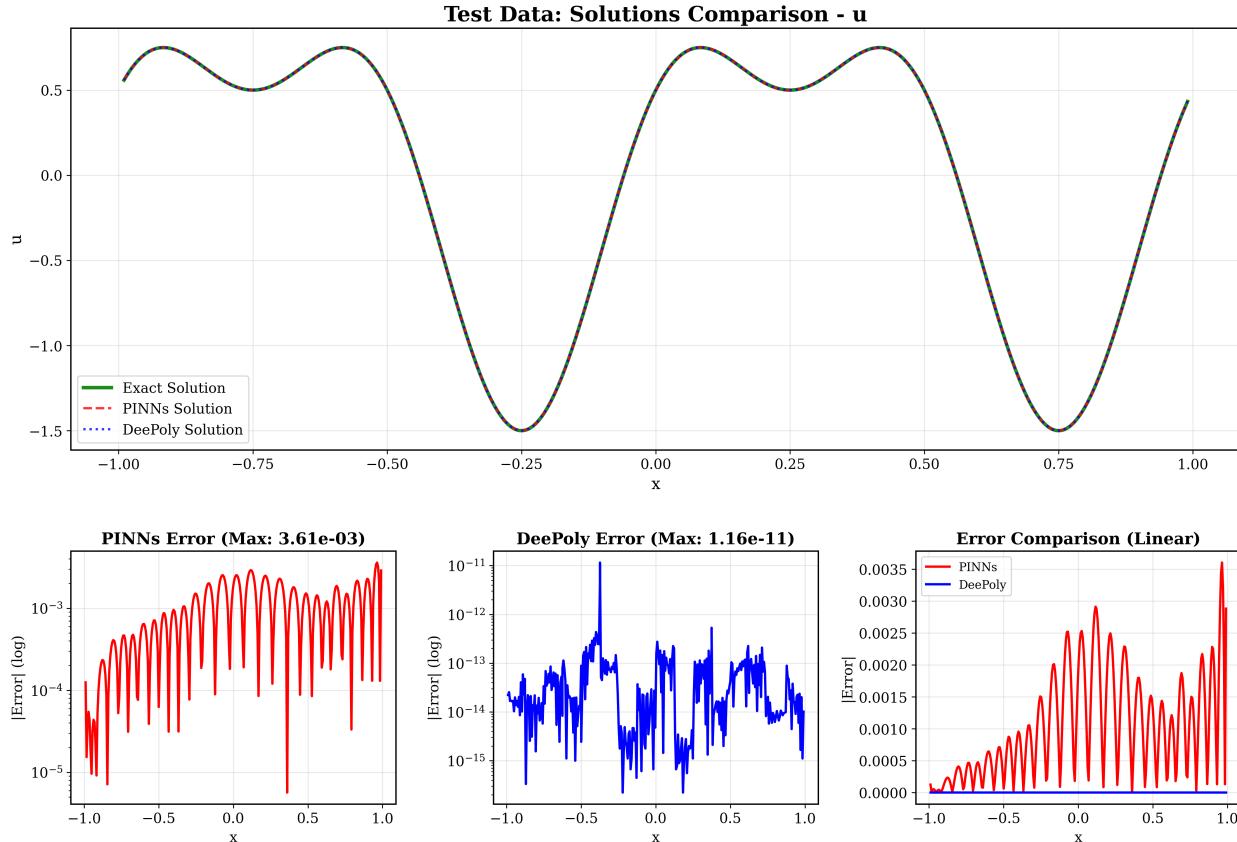
Problem Type: Baseline function approximation test case

Results Training Data Analysis

Training Data Analysis - u



Test Data Analysis



Performance Metrics

Component	Time Cost
Scoper/DNN	7.2049 seconds
Sniper	0.1630 seconds
Total	7.3679 seconds

Additional Information

- **Error Analysis:** Detailed metrics available in `cases/func_fitting_cases/test_sin/results/error_analysis`
 - **Configuration:** See `cases/func_fitting_cases/test_sin/config.json`
 - **Data Generation:** Automatic sampling with configurable density
-

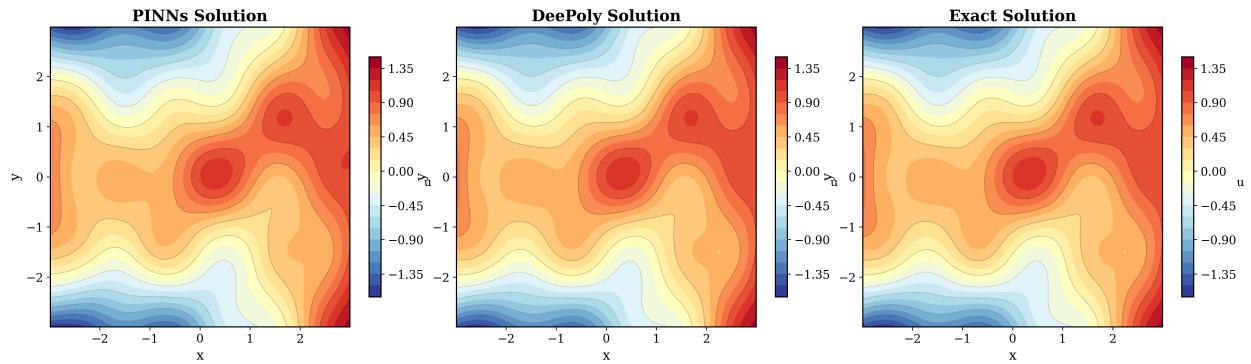
2. Complex 2D Function (func_fitting_cases/case_2d) **Function:** Multi-Gaussian peaks + trigonometric + polynomial terms

Domain: $[-3,3]^2$ | **Results:** MSE 4.85e-17, Max Error 2.00e-07, Time 42.9s

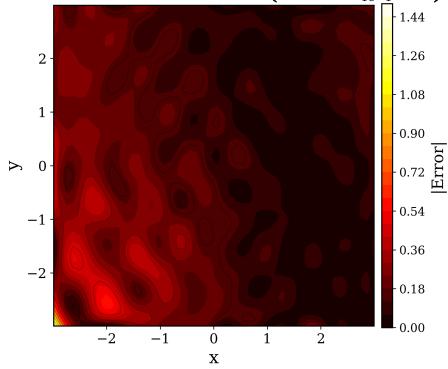
3. Discontinuous Function (func_fitting_cases/discontinuous_case1) **Function:** Near-discontinuous with sharp transitions

Domain: Variable | **Results:** MSE 2.11e-05, Max Error 1.83e-02, Time 14.0s

Test Data Analysis - u



PINNs Error Distribution (Max: 1.50e-01)



DeePoly Error Distribution (Max: 2.00e-07)

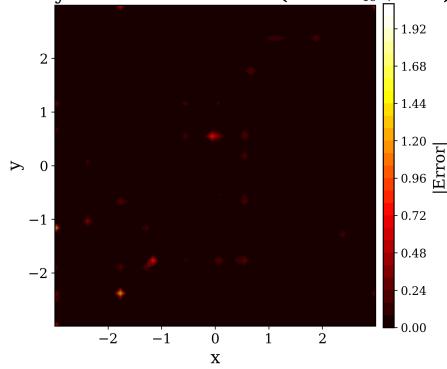


Figure 1: 2D Function Results

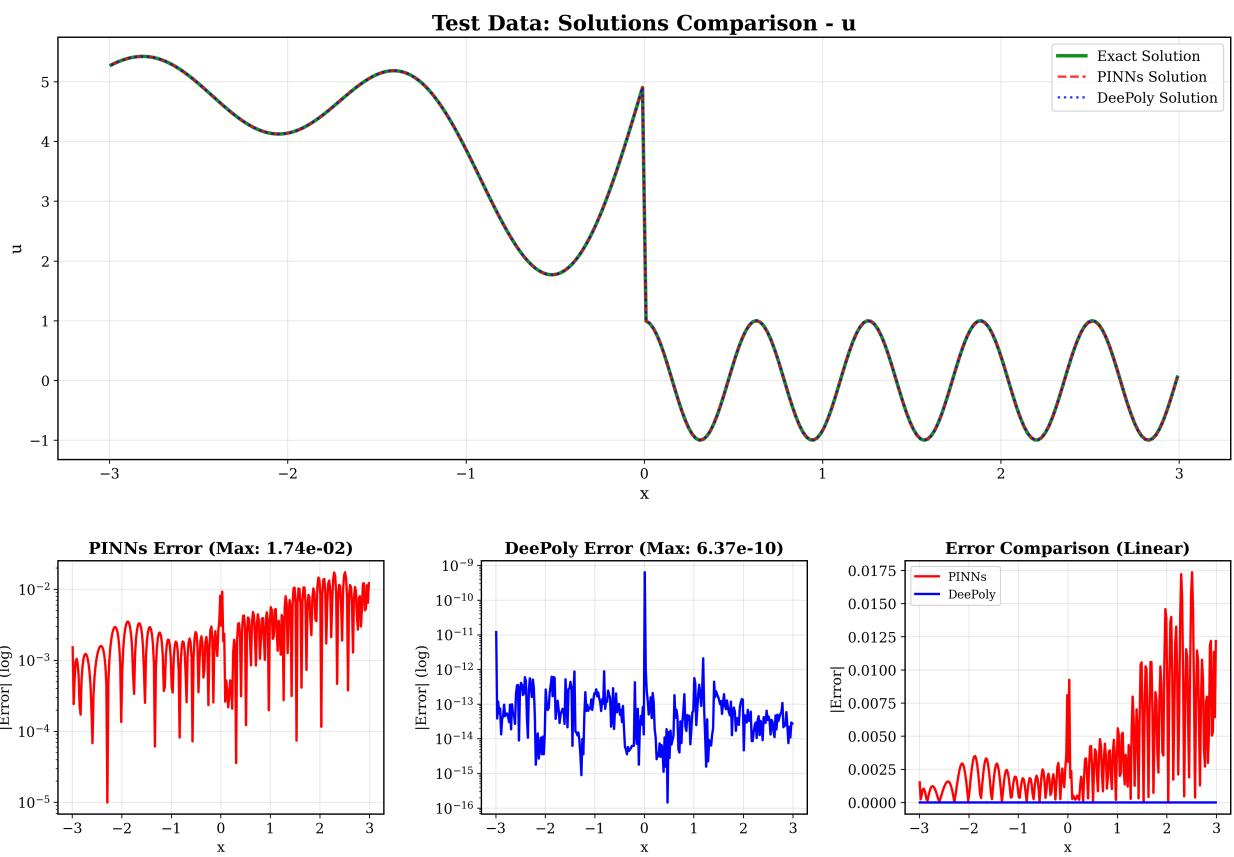
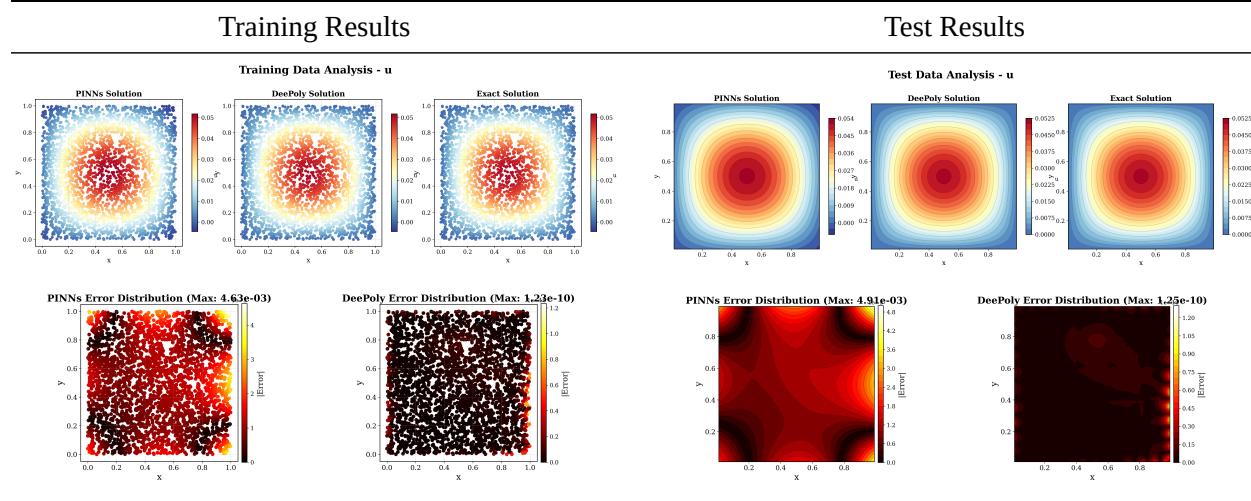


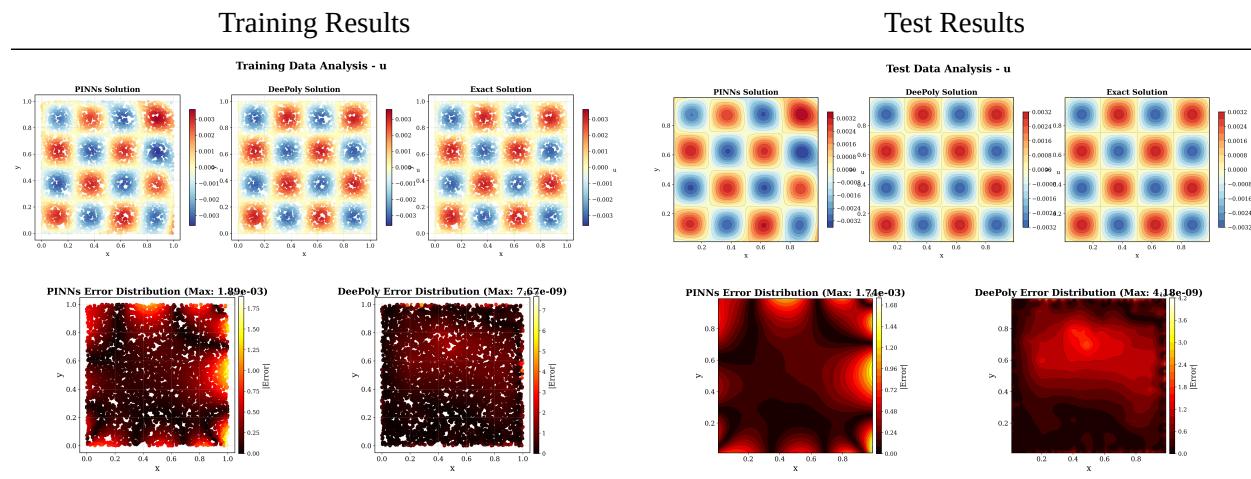
Figure 2: Discontinuous Results

PDE Solving Cases

4. 2D Poisson Equation (linear_pde_cases/poisson_2d_sinpixsinpiy) PDE: $\nabla^2 u = -\sin(x)\sin(y)$ | vs PINNs: $10^8 \times$ better accuracy (MSE 1.45e-14 vs 1.37e-06)

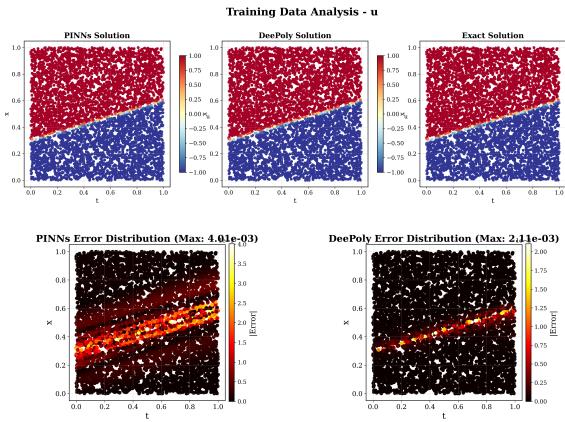


5. High-Frequency Poisson (linear_pde_cases/poisson_2d_sin4pixsin4piy) PDE: $\nabla^2 u = -32 \sin(4x)\sin(4y)$ | Higher frequency variant

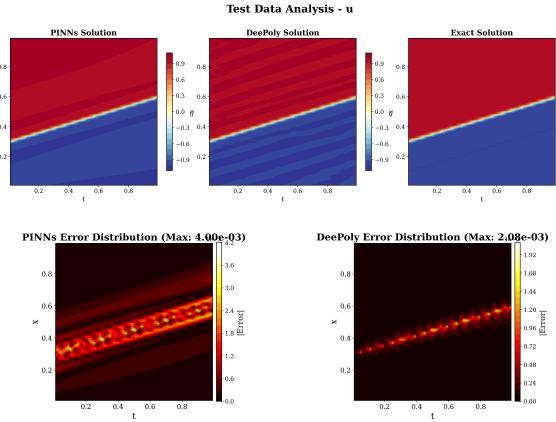


6. Linear Convection with Discontinuity (linear_pde_cases/linear_convection_discontinuity) PDE: $u_t + 0.3u_x = 0$ | Discontinuous wave propagation

Training Results



Test Results



Performance Summary

Case Type	Best MSE	Key Feature
Smooth Functions	4.85e-17	Multi-modal complexity
Discontinuous	2.11e-05	Sharp transitions
Linear PDEs	1.45e-14	vs PINNs: $10^8 \times$ better
Time-dependent	-	Discontinuous propagation