1. Visualizing Loss Landscapes: PINNs vs. Data-Driven (40 points + 20 points)

Physics-Informed Neural Networks (PINNs) have emerged as a powerful tool for solving partial differential equations (PDEs). However, unlike traditional supervised learning (Data-Driven), PINNs often suffer from "pathological" optimization behaviors. The loss landscape of a PINN is determined by high-order derivatives of the network, which can result in highly non-convex, rough, and ill-conditioned landscapes, especially as the frequency of the solution increases.

In this exercise, you will investigate the spectral bias and optimization complexity of neural PDE solvers. You will solve a multiscale Poisson equation using both a **PINN** (residual-based) and a **Data-Driven** (supervised) approach. Your goal is to visualize and compare how the loss landscape degrades as the frequency of the target solution increases.

Problem Statement

We focus on a prototypical linear elliptic PDE, the **Poisson Equation**, defined on a 2D square domain $D = [0,1]^2$:

$$\begin{aligned} -\Delta u &= f, \quad \text{in } D, \\ u &= 0, \quad \text{on } \partial D. \end{aligned}$$

To control the complexity of the problem, we define a source term f comprised of K spatial scales. The source term and its corresponding analytical solution $u(x,y)$ are given by:

$$f(x,y) = \frac{\pi}{K^2} \sum_{i,j=1}^{K} a_{ij} \cdot (i^2 + j^2)^r \sin(\pi i x) \sin(\pi j y)$$

$$u(x,y) = \frac{1}{\pi K^2} \sum_{i,j=1}^{K} a_{ij} \cdot (i^2 + j^2)^{r-1} \sin(\pi i x) \sin(\pi j y)$$

where $r = 0.5$ and coefficients $a_{ij} \sim \mathcal{N}(0,1)$. The parameter K acts as an indicator for the problem's complexity: as K increases, the solution contains higher frequency components.

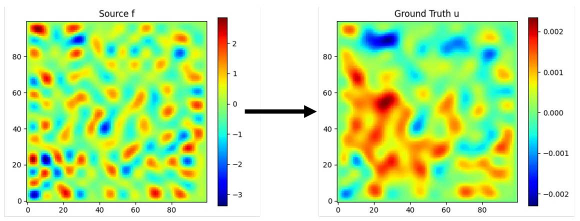

Figure 1: Visualization of one data sample with $K = 16$.

Tasks

Task 1: Data Generation (10 points)

Write a script to generate a dataset of input-output pairs $(f^{(i)}, u^{(i)})$. Discretize the domain D into an $N \times N$ structural grid (e.g. $N = 64$). Generate different samples by randomizing the coefficients

a_{ij} for each sample. Plot 3 examples of the source field f and the corresponding solution field u for $K = 1, 4, 8, 16$.

Task 2: Implementation and Training (30 points)

Implement a standard Multi-Layer Perceptron (MLP) with 3-4 hidden layers. The MLP here acts as a direct function approximator $u_\theta : \mathbb{R}^2 \to \mathbb{R}$. The input to the network is the spatial coordinates (x, y) of a point, and the output is the predicted scalar value $\hat{u}$ at that location. For this task, you will optimize the network to solve the PDE for one specific source term f only. The model is expected to fit the solution for this single case; no generalization to other source terms or distributions is required in this exercise. You will perform this optimization twice: once using the Data-Driven loss and once using the PINN loss.

- **(20 points)** For PINN loss, you need to define the loss calculator similar to the following form: $\mathcal{L}_{\text{PINN}}(\theta) = \frac{1}{N_f} \sum |-\Delta \hat{u}_\theta - f|^2 + \lambda \mathcal{L}_{\text{BC}}$. Then use the automatic differentiation to calculate the gradient.
- **(10 points)** For Data-Driven loss, the loss calculator: $\mathcal{L}_{\text{Data}}(\theta) = \frac{1}{N_d} \sum |\hat{u}_\theta - u_{\text{exact}}|^2$.

You need to train both models for different complexity levels: **Low** ($K = 1$), **Medium** ($K = 4$), and **High** ($K = 16$). The specific source term f to be solved can be defined by yourself. We recommend selecting one representative instance from the samples generated in the Task 1 to ensure consistency in your comparisons. Please record your final L_2 relative error, training loss curve, and model prediction in your report.

Note: To accelerate convergence, adopt a combined optimizer strategy: start with **Adam** and switch to **L-BFGS** for fine-tuning.

Task 3: Loss Landscape Visualization (BONUS - 20 points)

Using the techniques from *Li et al. (2018)*, visualize the geometry of the loss function around the converged (or stuck) solution parameters θ^*. Compute the loss on a 2D plane:

$$g(\alpha, \beta) = \mathcal{L}(\theta^* + \alpha\delta + \beta\eta) \tag{1}$$

where δ and η are two principal direction vectors.

- **(15 points)**: Provide 2D contour plots or 3D surface plots of the landscapes for both PINN and Data-Driven models across the three K levels.
- **(5 points)**: Qualitatively describe the differences. Is the PINN landscape sharper? Does it exhibit more local minima as K increases compared to the Data-Driven approach? Reference the "Failure modes" described by *Krishnapriyan et al. (2021)* in your discussion.

References

- Krishnapriyan et al. (2021). Characterizing possible failure modes in physics-informed neural networks.
- Li et al. (2018). Visualizing the loss landscape of neural nets.