Fourier Neural Operator

For Parametric Partial Differential Equations

Vincent Scholz & Dr. Yaohua Zang

Professorship for Data-driven Materials Modeling

June 25, 2025

Organizational Details – Final Exam

Registration:

- Exam is available in **TUM Online**
- Registration deadline: **2 July 2025**

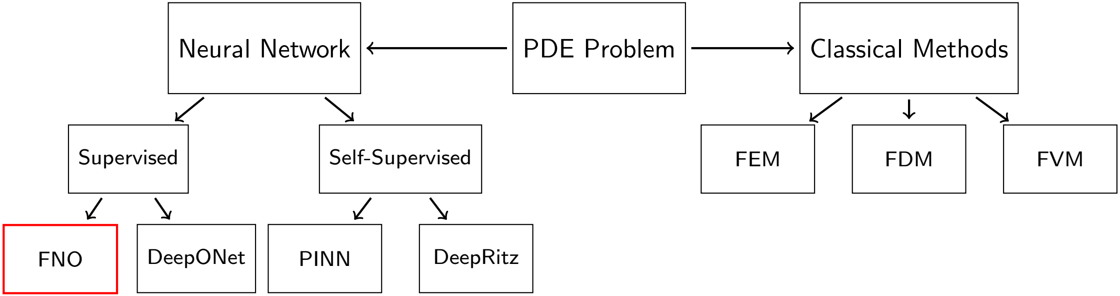
Exam Format:

- Exam task will be published on **2 July 2025** via **Moodle**
- You will:
  - Complete the assigned tasks
  - Submit a written report (template provided)
  - Submit your code (or link to code)
  - Give a short presentation (10–15 minutes, excl. Q&A)

Deadlines:

- Report & Code Submission: **23 July 2025**
- Presentation Day: **30 July 2025** (time slots will be assigned individually)

Overview



Neural Network
PDE Problem
Classical Methods
Supervised
Self-Supervised
FEM
FDM
FVM
FNO
DeepONet
PINN
DeepRitz

Table of Contents

1. Repetition to Parametric PDEs
2. Fourier Neural Operator (FNO)
3. FNO Training and Implementation
4. When to Use FNO: Advantages and Limitations

Table of Contents

1. Repetition to Parametric PDEs
2. Fourier Neural Operator (FNO)
3. FNO Training and Implementation
4. When to Use FNO: Advantages and Limitations

Parametric PDE Problems

We consider a general **parametric PDE** in form:

$$F(x, y, u, u_x, u_y, u_{xx}, u_{yy}, \cdots, a) = 0, \quad (x, y) \in \Omega$$

Where:

- $u(x, y)$ is the **solution**
- $a(x, y)$ is a **parameter** (e.g., conductivity or source)
- Appropriate boundary conditions on $\partial\Omega$

Operator Learning Problem

Goal: Solve for $u(x, y; a)$ given any input parameter $a(x, y)$

This defines an **operator learning problem**:

$$\mathcal{G} : a(x, y) \in \mathcal{A} \longrightarrow u(x, y) \in \mathcal{U}$$

Where:

- $\mathcal{A}$: function space for inputs
- $\mathcal{U}$: function space for outputs
- $\mathcal{G}$: **solution operator** we want to approximate

From Vectors to Functions

Traditional neural networks work on **finite-dimensional vectors**:

$$\mathbf{v}_{l+1} = \sigma_l(W_l \mathbf{v}_l)$$

But in scientific computing, we often deal with **functions** rather than vectors.

Goal: Learn mappings between functions directly, without discretization.

This leads us to **neural operators** – networks that learn mappings of the form:

$$\mathcal{G} : v(x) \mapsto u(x)$$

Integral Operators (I)

Key idea: Process **function-to-function** mappings via integration.

An **integral operator** maps a function $v(y)$ to another function $u(x)$ by integrating over a kernel $k(x,y)$:

$$u(x) = \int_{\Omega} k(x,y)\, v(y)\, dy$$

Used extensively in:

- Inverse problems
- Gaussian processes
- PDE theory

Integral Operators (II): Visual Concept

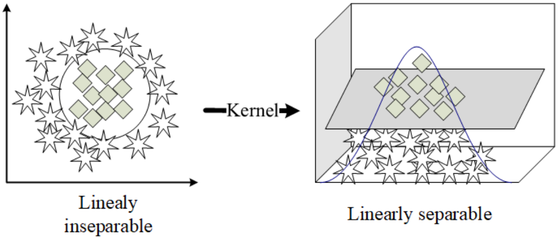


Figure: Illustration of a kernel-based operator acting on a function.

Kernel function: $k(x, y)$ describes how inputs y influence outputs at location x.

This is the foundation of **kernel-based neural operators**.

Kernel-Based Neural Operator

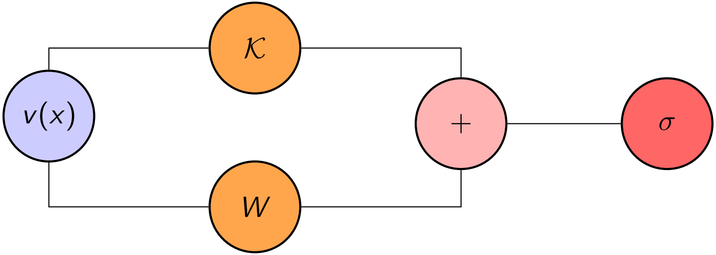
Extension to **function spaces** using **integral operators**:

$$v_{l+1}(x) = \sigma_l \left(\int_\Omega k_l(x, y) v_l(y)\, dy + W_l v_l(x) \right)$$

Where:

- $x, y \in \Omega$: spatial coordinates
- $k_l(x, y)$: learnable **kernel function**
- W_l: pointwise (local) linear transformation
- σ_l: nonlinear activation function

Kernel Operator: Computational Flow



Two pathways:

- **Non-local interaction** via kernel $\mathcal{K}(x, y)$
- **Local update** via W

Table of Contents

1. Repetition to Parametric PDEs
2. Fourier Neural Operator (FNO)
3. FNO Training and Implementation
4. When to Use FNO: Advantages and Limitations

Motivation Behind FNO

Computer Vision: CNNs work well because images have **local patterns**

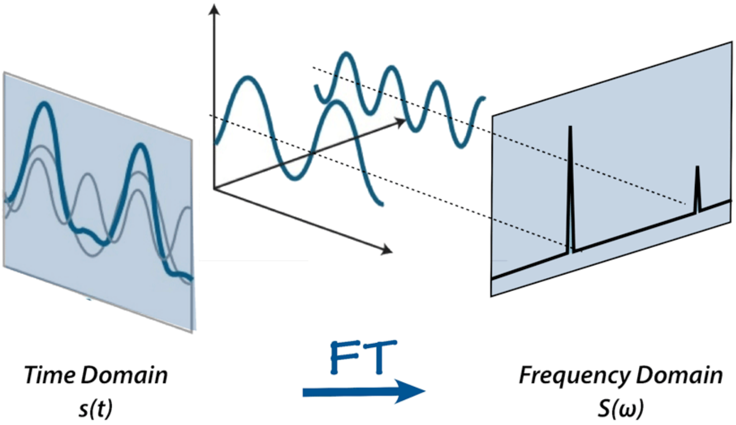
Parametric PDEs: Functions often exhibit **global correlations**

- Smooth variations across domain
- Long-range dependencies
- Local convolutions may be inefficient

FNO Solution:

- Replace local convolution with **global convolution**
- Use **Fourier transform** for efficiency
- Leverage compact representation in **frequency domain**

Fourier Transform Visualization



The Fourier Layer: Core Building Block

Why Fourier?

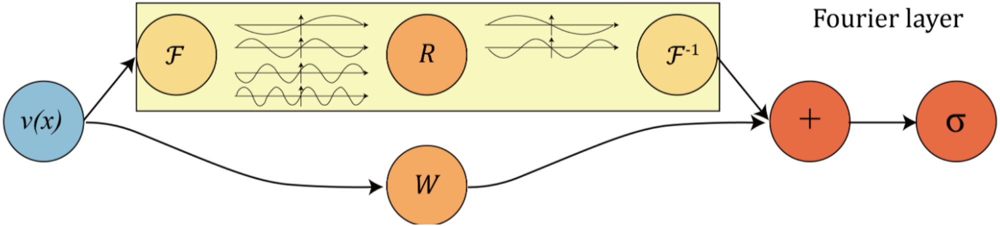
- **Speed**: $\mathcal{O}(n^2) \to$ quasilinear with FFT
- **Compactness**: PDE functions concentrated in low frequencies

Three-step process:

1. **Fourier Transform**: $v_l \to \mathcal{F}(v_l)$
2. **Linear Transformation**: Apply R to low-frequency modes
3. **Inverse Transform**: $\mathcal{F}^{-1} \to$ spatial domain

Fourier Layer Architecture

$$v_{l+1}(x) = \sigma\left(\mathcal{F}^{-1}\left(R \cdot \mathcal{F}(v_l)\right)(x) + W v_l(x)\right)$$



Key Design Choices

Remark A: Truncation

- **Discard high-frequency modes**
- Apply R only on lower modes
- Most useful information for PDEs in low frequencies

Remark B: Activation in Spatial Domain

- Activations applied **after** inverse Fourier transform
- Recovers **high-frequency details** omitted by truncation
- Captures **non-periodic boundary behaviors**

Complete FNO Architecture

Three main components:

1. **Input Layer**: **Lifting**

$$v_0 = P_\theta(a, x)$$

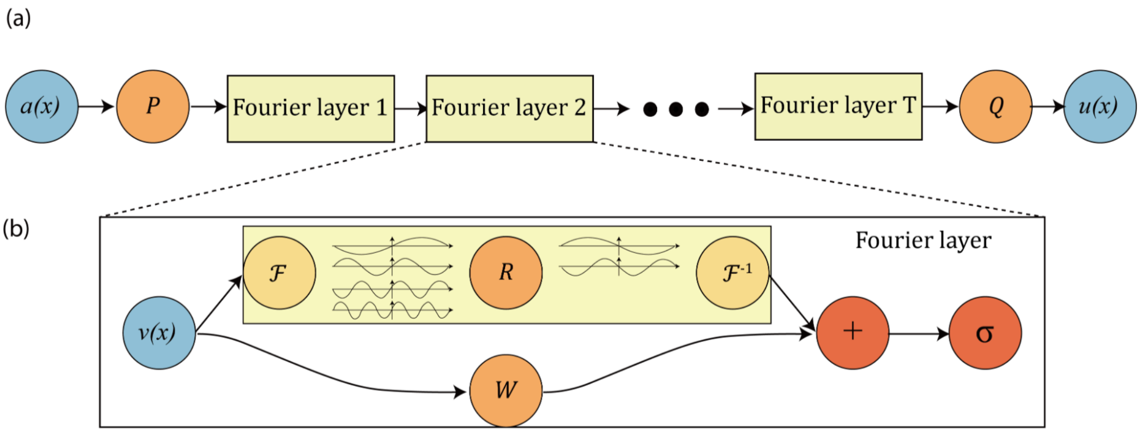
2. **Fourier Layers**: **Global Transformations**

$$v_{l+1} = \sigma\left(\mathcal{F}^{-1}(R \cdot \mathcal{F}(v_l)) + W v_l\right)$$

3. **Output Layer**: **Projection**

$$u = Q_\theta(v_L)$$

FNO Architecture



FNO Pipeline Summary

Input $a(x)$ $\rightarrow$ Lifting P_θ $\rightarrow$ Fourier Layers $\rightarrow$ Projection Q_θ $\rightarrow$ Solution $u(x)$

Key Innovation: Combines expressiveness of deep networks with efficiency of Fourier methods

Especially suitable for learning operators from parametric PDE problems

Table of Contents

1. Repetition to Parametric PDEs
2. Fourier Neural Operator (FNO)
3. **FNO Training and Implementation**
4. When to Use FNO: Advantages and Limitations

Step 1: Prepare Training Data

FNO learns from **labeled training pairs** (a, u) on a **fixed regular mesh**:

- **Mesh**: $\Xi = \{\xi_1, \xi_2, \ldots, \xi_n\}$
- **Discretization**: $a(\Xi), u(\Xi)$
- **Dataset**: $\mathcal{D} = \{(a^{(i)}(\Xi), \Xi), u^{(i)}(\Xi)\}_{i=1}^{N_{\text{data}}}$

Note: FNO requires **uniform grid** for Fourier transforms

Data Generation:

- Inputs a sampled from function space $\mathcal{A}$
- Outputs u computed using accurate solvers (FEM/FDM)

Step 2: Build FNO Model

Forward pass through FNO model $\mathcal{G}_\theta$:

$$v_0 = P_\theta(a(\Xi), \Xi) \quad \text{(input lifting)} \tag{1}$$

$$v_{l+1} = \sigma\left(\mathcal{F}^{-1}(R \cdot \mathcal{F}(v_l)) + W v_l\right), \quad l = 0, \ldots, L-1 \tag{2}$$

$$u(\Xi) = Q_\theta(v_L) \quad \text{(projection to output)} \tag{3}$$

Parameters:

- P_θ, Q_θ: input/output networks
- R, W: learnable parameters in Fourier layers
- σ: activation function (e.g., ReLU)

Steps 3 & 4: Training Process

Loss Function (Mean Squared Error):

$$\theta^* = \arg\min_\theta L(\theta) = \frac{1}{N_{\text{data}}} \sum_{i=1}^{N_{\text{data}}} \left| \mathcal{G}_\theta(a^{(i)}(\Xi), \Xi) - u^{(i)}(\Xi) \right|^2$$

Optimization (SGD/Adam):

$$\theta_{t+1} = \theta_t - l_r \nabla_\theta L(\theta_t)$$

Where:

- l_r: learning rate
- t: training step

Table of Contents

1. Repetition to Parametric PDEs
2. Fourier Neural Operator (FNO)
3. FNO Training and Implementation
4. When to Use FNO: Advantages and Limitations

Advantages of FNO

- **High accuracy and efficiency**
  - Typically outperforms DeepONet
  - Better prediction accuracy
  - Higher computational speed
- **Fast inference**
  - Rapid solving for any new input $a \in \mathcal{A}$
  - Suitable for real-time applications
  - Many-query scenarios
- **Global receptive field**
  - Captures long-range dependencies
  - Efficient frequency domain operations

Disadvantages of FNO

- **Data inefficiency**
    - Requires large number of labeled pairs (a, u)
    - Expensive numerical simulations
    - Costly experimental data collection
- **Limited generalization**
    - Struggles with out-of-distribution inputs
    - Performance degrades for unseen parameter ranges
- **Mesh dependence**
    - Requires regular grid for FFT
    - Less flexible for irregular domains
    - Complex geometries challenging
    - Reduced accuracy at non-mesh locations

Summary and Future Directions

FNO Summary:

- Powerful method for **parametric PDE operator learning**
- Combines deep learning with **Fourier efficiency**
- Excellent for **smooth, global** PDE solutions

Future Research Directions:

- Irregular domain handling
- Few-shot learning approaches
- Multi-scale and adaptive methods
- Integration with physics-informed constraints