

# Meta-Programming and Hybrid Parallel Strategies for Solving PDEs: An FDM and PINN Comparison<sup>1 2</sup>

Seminar Presentation III

LI YIHAI

Supervisor: Mike Peardon

Mathematical Institute

September 14, 2024



Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

<sup>1</sup> full docs: <https://liyihai.com/html/index.html>

<sup>2</sup> repository: <https://github.com/liyihai-official/Final-Project>

- 1 Introduction
  - Related Work
- 2 Problem Setups
- 3 N-Dimensional Matrix Implementation
- 4 Parallelization of Multi-dimensional Matrices on cartesian Topologies
- 5 PDE Solver Implementation
- 6 Experiments
- 7 Discussion

- 1 Introduction
  - Related Work
- 2 Problem Setups
- 3 N-Dimensional Matrix Implementation
- 4 Parallelization of Multi-dimensional Matrices on cartesian Topologies
- 5 PDE Solver Implementation
- 6 Experiments
- 7 Discussion

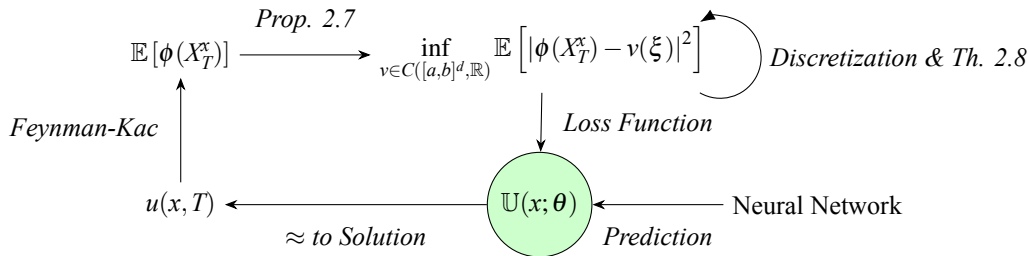
# Introduction

## Recap - Deep Neural Network

### Kolmogorov PDEs

Solving  $u(x, T)$ , for  $\mathbb{R}^1 \ni T > 0, x \in \mathbb{R}^d, t \in [0, T], u(t, x) = u \in \mathbb{R}^1, \mu(x) \in \mathbb{R}^d, \sigma(x) \in \mathbb{R}^{d \times d}$ ,

$$u_t = \frac{1}{2} \text{Trace}_{\mathbb{R}^d} [\sigma(x) [\sigma(x)]^* \text{Hess}_x u] + \langle \mu(x), \nabla_x u \rangle_{\mathbb{R}^d} \quad (1)$$



**Figure:** Deep Neural Network (DNN) Methodology of Solving Kolmogorov PDEs **[FIRST]**

# Introduction

## Recap - Physics Informed Neural Network

### General Form of PDEs

- $u(t, x)$  denotes with the target function,  $x \in \mathbb{R}^d$ .
- $\Gamma[\cdot; \lambda]$  is a non-linear operator parameterized by  $\lambda$ .

$$u_t(t, x) + \Gamma[u; \lambda] = 0 \quad (2)$$

Define  $f(t, x)$  to be given by  $f(t, x) = u_t(t, x) + \Gamma[u; \lambda]$  (3)

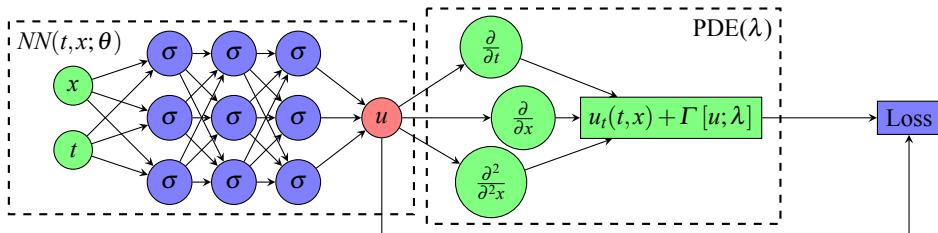


Figure: PINN, with with 3 fully connected hidden layers

### Comparing With Finite Difference Time Domain Method (FDTD)

#### ① Deep Neural Network [**FIRST**]

- Gives lower quality approximations.
- Takes longer time to train.
- Possible to solve high dimension PDEs

#### ② Physics Informed Neural Network

- Gives higher quality approximations.
- Takes longer time to train.
- Has more flexible way to get results.
- Possible to solve high dimension PDEs

- 1 Introduction
  - Related Work
- 2 Problem Setups
- 3 N-Dimensional Matrix Implementation
- 4 Parallelization of Multi-dimensional Matrices on cartesian Topologies
- 5 PDE Solver Implementation
- 6 Experiments
- 7 Discussion

# Statement

## Challenges & Objectives

### Challenges of Nonlinear Multi-Scale Systems

- ① Difficulty in modeling and predicting inhomogeneous cascades of scales.
- ② High computational costs and uncertainties in classical methods.

### Project Objectives

- ① Implement FDTD and PINN in C++/C.
- ② Implement FDTD hybrid parallel version using MPI/OpenMP.
- ③ Implement PINN GPU parallel version using Libtorch/CUDA.
- ④ Optimize performance through parallel computing frameworks.
- ⑤ Evaluate the efficiency and accuracy of FDTDs and PINNs.



# General Form of problem

The PDE parametrized by number  $\lambda$  and an operator  $\mathcal{N}[\cdot; \lambda]$ , and assume the variable  $x$  is a 2D or 3D spatio-vector which is written in

$$\begin{cases} \frac{\partial u}{\partial t}(t, \vec{x}) + \mathcal{N}[u; \lambda] = 0 \\ u(0, \vec{x}) = \varphi(\vec{x}) \end{cases} \quad (4)$$

where  $\varphi$  is the initial condition, and  $\vec{x} \in \Omega, t \in [0, +\infty)$ .

## Boundary Conditions

The Dirichlet and Von Neumann boundary conditions are formed as

$$\begin{cases} u(t, \vec{x}) = g(t, \vec{x}) \\ \frac{\partial u}{\partial \vec{n}} = g(t, \vec{x}) \end{cases} \quad (5)$$

where  $\vec{n}$  is the normal vector on  $\overline{\Omega}$  the boundary of domain  $\Omega$ .

# Heat Equation

## Specific Example

### Heat Equation 2D

The function  $u(t,x,y) = x + y - xy$ ,  $\forall \alpha \in \mathbb{R}^1$ , is the solution of 2D Heat Equation 6 below

$$\begin{aligned} \frac{\partial u}{\partial t} &= \alpha \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) & (x,y) \in \Omega, t \in [0, +\infty) \\ u(0,x,y) &= \varphi(x,y) = 0 & (x,y) \in \Omega \\ u(t,x,y) &= g(x,y) = \begin{cases} y, & x=0, y \in (0,1) \\ 1, & x=1, y \in (0,1) \\ x, & y=0, x \in (0,1) \\ 1, & y=1, x \in (0,1) \end{cases} & t \in [0, +\infty) \end{aligned} \quad (6)$$

# Heat Equation

## Specific Example

### Heat Equation 3D

The function  $u(t, x, y, z) = x + y + z - 2xy - 2xz - 2yz + 4xyz$ ,  $\forall \alpha \in \mathbb{R}^1$ , is the solution of 3D Heat Equation 7 below

$$\begin{aligned} \frac{\partial u}{\partial t} &= \alpha \left( \frac{\partial^2 u}{\partial^2 x} + \frac{\partial^2 u}{\partial^2 y} + \frac{\partial^2 u}{\partial^2 z} \right) & (x, y, z) \in \Omega, t \in [0, +\infty) \\ u(0, x, y, z) &= \varphi(x, y, z) = 0 & (x, y, z) \in \Omega \end{aligned} \quad (7)$$
$$u(t, x, y, z) = g(x, y, z) = \begin{cases} y + z - 2yz, & x = 0, \\ 1 - y - z + 2yz, & x = 1, \\ x + z - 2xz, & y = 0, \\ 1 - x - z + 2xz, & y = 1, \\ x + y - 2xy, & z = 0, \\ 1 - x - y + 2xy, & z = 1 \end{cases} \quad t \in [0, +\infty)$$

- 1 Introduction
  - Related Work
- 2 Problem Setups
- 3 N-Dimensional Matrix Implementation**
- 4 Parallelization of Multi-dimensional Matrices on cartesian Topologies
- 5 PDE Solver Implementation
- 6 Experiments
- 7 Discussion

- 1 Introduction
  - Related Work
- 2 Problem Setups
- 3 N-Dimensional Matrix Implementation
- 4 Parallelization of Multi-dimensional Matrices on cartesian Topologies
- 5 PDE Solver Implementation
- 6 Experiments
- 7 Discussion

- 1 Introduction
  - Related Work
- 2 Problem Setups
- 3 N-Dimensional Matrix Implementation
- 4 Parallelization of Multi-dimensional Matrices on cartesian Topologies
- 5 PDE Solver Implementation**
- 6 Experiments
- 7 Discussion

- 1 Introduction
  - Related Work
- 2 Problem Setups
- 3 N-Dimensional Matrix Implementation
- 4 Parallelization of Multi-dimensional Matrices on cartesian Topologies
- 5 PDE Solver Implementation
- 6 Experiments**
- 7 Discussion

- 1 Introduction
  - Related Work
- 2 Problem Setups
- 3 N-Dimensional Matrix Implementation
- 4 Parallelization of Multi-dimensional Matrices on cartesian Topologies
- 5 PDE Solver Implementation
- 6 Experiments
- 7 Discussion



Thank you for your attention!

Any questions?