Numerical Methods for Solving Partial Differential Equations

From Neural Networks to PDE Solving

Dr. Yaohua Zang & Vincent Scholz

Professorship for Data-driven Materials Modeling

15th May 2025

Outline

Table of Contents

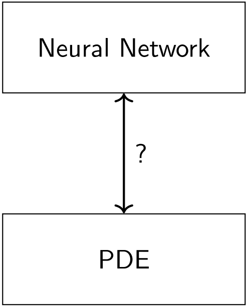
Recap of Previous Course Content

Our Journey So Far:

- Introduction to Neural Networks
- Structural Approaches to PDEs
- Machine Learning Data Generation

Current Focus:

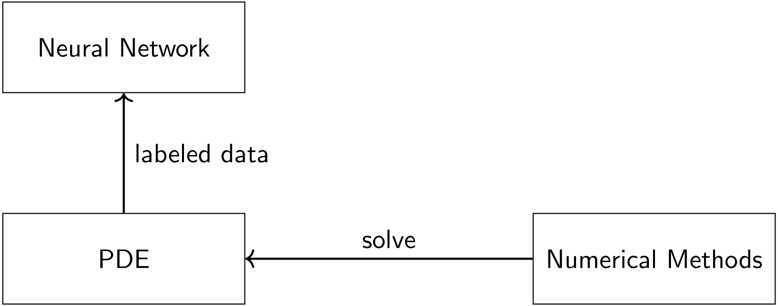
- Generating Training Data for ML
- Numerical Solutions to PDEs



Why Numerical Methods?

- Analytical solutions are rare
- Need computational approaches for complex problems
- Generate ground truth data for machine learning

The Role of Numerical Methods in Our Workflow



- Numerical methods bridge the gap between theoretical PDEs and practical computations
- Essential for generating high-quality training data
- Provide ground truth for machine learning models

Table of Contents

Three Major Numerical Approaches

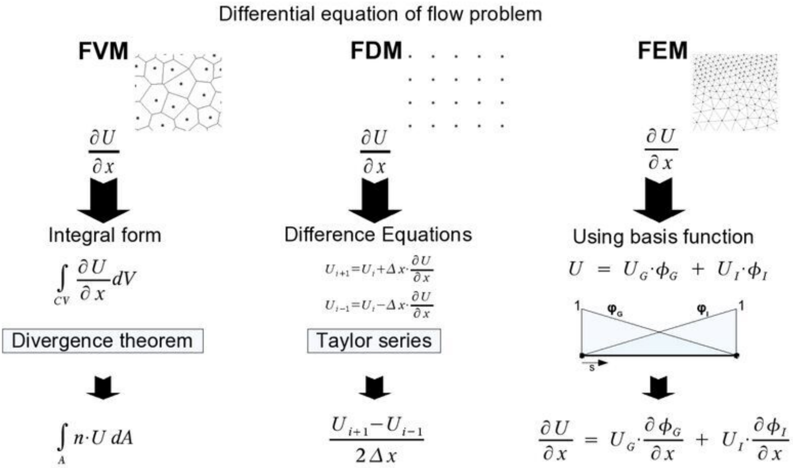


Figure 1: Visualization of Numerical Methods

Summary Comparison: FDM vs FEM vs FVM

Feature	**Finite Differences (FDM)**	**Finite Elements (FEM)**	**Finite Volumes (FVM)**
Main idea	Pointwise approximation of derivatives	Approximate solution via variational form	Integral conservation over volumes
Based on	Taylor expansion	Weak (variational) form	Integral form (conservation laws)
Mesh type	Structured grids	Unstructured/ structured meshes	Structured/ unstructured volumes
Unknowns represent	Values at grid points	Coefficients of basis functions	Cell averages
Conservation	✗ (not enforced)	✎ (global only)	✓ (local and global)
Ease of implementation	✓ (easiest)	✗ (needs integrals)	✎ (requires flux handling)
Geometry handling	✗ (limited)	✓ (very flexible)	✓ (flexible)
Best suited for	Simple prototyping	Structural mechanics	CFD and transport

Table of Contents

Why Finite Differences?

- Partial Differential Equations (PDEs) model various physical systems.
- PDEs involve **spatial and temporal derivatives**.
- Solving PDEs numerically means we must **approximate derivatives**.

Types of Derivatives in Typical PDEs:

- First-order spatial: $\frac{\partial u}{\partial x}$
- First-order temporal: $\frac{\partial u}{\partial t}$
- Second-order spatial: $\frac{\partial^2 u}{\partial x^2}$
- Second-order temporal: $\frac{\partial^2 u}{\partial t^2}$ (e.g., wave equations)

Goal: Approximate these using finite differences.

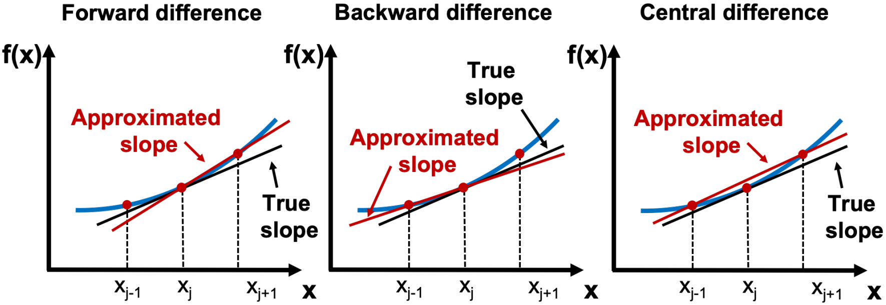
Finite Difference Method: Concept

- Based on Taylor series expansion
- Replace continuous derivatives with algebraic difference formulas
- Transform PDEs into solvable algebraic systems

Key steps:

- Discretize the domain
- Approximate derivatives using difference formulas
- Assemble algebraic equations

First-Order Spatial Derivative Approximations



First-Order Spatial Derivative Approximations

Using Taylor expansion

$$u(x+\delta x) = u(x) + u'(x)\delta x + u''(x)\frac{\delta^2 x}{2} + O(\delta^3 x), \tag{1}$$

we get:

- **Forward difference:**

$$\frac{du}{dx} \approx \frac{u(x+h) - u(x)}{h}, \quad O(h)$$

- **Backward difference:**

$$\frac{du}{dx} \approx \frac{u(x) - u(x-h)}{h}, \quad O(h)$$

- **Central difference (more accurate):**

$$\frac{du}{dx} \approx \frac{u(x+h) - u(x-h)}{2h}, \quad O(h^2)$$

First-Order Time Derivative Approximations

Time derivatives are handled similarly to spatial ones:

- **Forward Euler:**
$$\frac{\partial u}{\partial t} \approx \frac{u^{n+1} - u^n}{\Delta t}$$
- **Backward Euler:**
$$\frac{\partial u}{\partial t} \approx \frac{u^n - u^{n-1}}{\Delta t}$$
- **Central difference (for second-order accuracy):**

$$\frac{\partial u}{\partial t} \approx \frac{u^{n+1} - u^{n-1}}{2\Delta t}$$

Note: These are directly analogous to the spatial formulas.

Second-Order Derivative Approximations

From Taylor expansions:

- Adding $u(x+h)$ and $u(x-h)$ gives:

$$u(x+h) + u(x-h) = 2u(x) + h^2 u''(x) + O(h^4)$$

- Rearranging:

$$\frac{d^2 u}{dx^2} \approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}$$

- **Error:** $O(h^2)$

Table of Contents

From Derivatives to Discretized PDEs

- Now that we can approximate derivatives, we can discretize entire PDEs.
- Let's look at concrete examples:
  - Poisson Equation (Elliptic PDE)
  - Heat Equation (Parabolic PDE)
- We'll see how the discrete derivatives yield matrix equations.

1D Poisson Equation

The 1D Poisson Equation:

$$-k\frac{d^2u}{dx^2} = f(x), \quad x \in [0, L]$$

- Models steady-state problems in physics
- Applications in electrostatics, heat conduction, etc.
- Boundary conditions: values of u specified at $x = 0$ and $x = L$
- Visuzallization at: Stackexchange

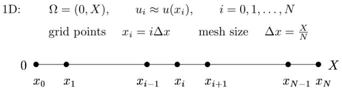
Domain Discretization

Discretizing the domain:

- Divide interval $[0, L]$ into N equal segments
- Mesh points: $\{0 = x_0 < x_1 < \cdots < x_N = L\}$
- Step size: $\Delta x = x_1 - x_0 = \frac{L}{N}$

Key concept:

- Replace continuous problem with discrete values at grid points
- Derivatives approximated using finite differences
- Results in a system of linear equations



Discretization of domain

Discretizing the Poisson Equation

Original PDE:

$$-k\frac{d^2u}{dx^2} = f(x), \quad x \in [0, L]$$

Discretized second derivative:

$$\frac{d^2u}{dx^2} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}$$

Gives:

$$-k\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} = f(x_i)$$

This leads to a system of linear equations for each u_i with $i = 1, \ldots, N-1$
Matrix form: $A\vec{u} = \vec{b}$

Matrix Form of the Discretized Poisson Equation

$$-\begin{bmatrix} -\frac{2k}{\Delta x^2} & \frac{k}{\Delta x^2} & \cdots & \cdots \\ \frac{k}{\Delta x^2} & -\frac{2k}{\Delta x^2} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \frac{k}{\Delta x^2} \\ \cdots & \cdots & \frac{k}{\Delta x^2} & -\frac{2k}{\Delta x^2} \end{bmatrix} \vec{u} = \begin{bmatrix} \frac{k}{\Delta x^2} u_0 \\ 0 \\ \cdots \\ 0 \\ \frac{k}{\Delta x^2} u_N \end{bmatrix} + \begin{bmatrix} f(x_1) \\ f(x_2) \\ \cdots \\ f(x_{N-2}) \\ f(x_{N-1}) \end{bmatrix}$$

- A is a tridiagonal matrix (sparse)
- u_0 and u_N are boundary values (known)
- The system $A\vec{u} = b$ can be solved using:
  - Direct methods: LU Decomposition, Cholesky
  - Iterative methods: Jacobi, Gauss-Seidel

Where Does the Matrix A Come From?

Discretization stencil:

$$-k\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} = f(x_i)$$

Rewriting for all i in vector form:

$$A = \frac{k}{\Delta x^2}\begin{bmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 1 & -2 & 1 \\ 0 & \dots & 0 & 1 & -2 \end{bmatrix}$$

Right-hand side:

$$\vec{b} = [f(x_1), \dots, f(x_{N-1})]^T + \text{Boundary terms}$$

Matrix form emerges from applying the stencil at each interior grid point!

1D Heat Equation

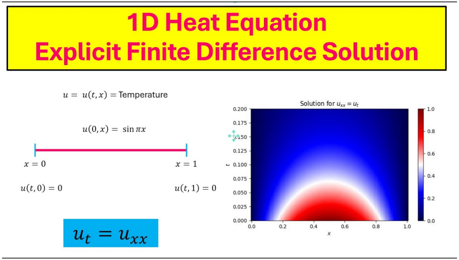
The 1D Heat Equation:

$$\frac{\partial u}{\partial t} = k\frac{\partial^2 u}{\partial x^2}$$

- $u(x,t)$ represents temperature at position x and time t
- k is thermal diffusivity
- Describes heat diffusion in a 1D rod

Initial and Boundary Conditions:

- Initial: $u(x,0) = \sin(\pi x)$ (Initial temperature)
- Boundary: $u(0,t) = 0$ and $u(1,t) = 0$ (Fixed temperatures)



Discretizing the Heat Equation

- Discretize space as before: $\{0 = x_0 < x_1 < \cdots < x_N = 1\}$
- Denote $u_i = u(x_i, t)$ and approximate PDE as:

$$\frac{du_i}{dt} = k\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}, \quad i = 1, \cdots, N-1$$

- Define vector $\vec{u}(t) = [u_1, u_2, \cdots, u_{N-1}]^T$
- System becomes a first-order ODE:

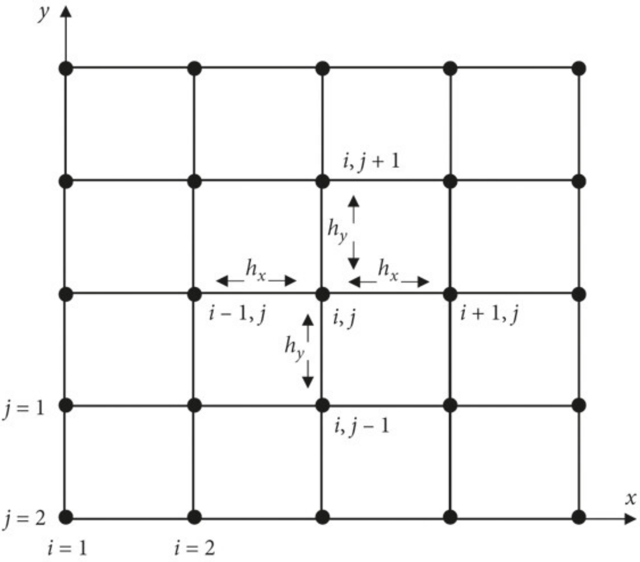
$$\frac{d\vec{u}}{dt} = A\vec{u} + b$$

Matrix Form of the Discretized Heat Equation

$$\frac{d\vec{u}(t)}{dt} = \begin{bmatrix} -\frac{2k}{\Delta x^2} & \frac{k}{\Delta x^2} & \cdots & \cdots \\ \frac{k}{\Delta x^2} & -\frac{2k}{\Delta x^2} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \frac{k}{\Delta x^2} \\ \cdots & \cdots & \frac{k}{\Delta x^2} & -\frac{2k}{\Delta x^2} \end{bmatrix} \vec{u} + \begin{bmatrix} \frac{k}{\Delta x^2} u_0 \\ 0 \\ \cdots \\ 0 \\ \frac{k}{\Delta x^2} u_N \end{bmatrix}$$

- This is a **first-order linear ODE system**
- Can be solved with numerical methods:
  - **Euler's method** (simplest but least accurate)
  - **Runge-Kutta methods** (higher order accuracy)
  - **Implicit methods** (better stability)

Time Discretization



Time Discretization Methods

For the system $\frac{d\vec{u}}{dt} = A\vec{u} + b$:

Explicit (Forward Euler):

$$\frac{\vec{u}^{n+1} - \vec{u}^{n}}{\Delta t} = A\vec{u}^{n} + b$$

$$\vec{u}^{n+1} = \vec{u}^{n} + \Delta t(A\vec{u}^{n} + b)$$

Properties:

- Simple to implement
- Conditionally stable
- Requires small time steps

Implicit (Backward Euler):

$$\frac{\vec{u}^{n+1} - \vec{u}^{n}}{\Delta t} = A\vec{u}^{n+1} + b$$

$$(\mathbf{I} - \Delta t A)\vec{u}^{n+1} = \vec{u}^{n} + \Delta t b$$

Properties:

- Requires solving a system at each step
- Unconditionally stable
- Allows larger time steps

Summary and Conclusions

- **Finite Difference Method (FDM)** is a powerful technique for solving PDEs numerically
- Key steps:
  - Discretize the domain into a grid
  - Replace derivatives with finite difference approximations
  - Form and solve a system of algebraic equations
- Considerations for implementation:
  - Trade-off between accuracy (grid refinement) and computational cost
  - Stability constraints for time-dependent problems
  - Choice of appropriate solver for the resulting system