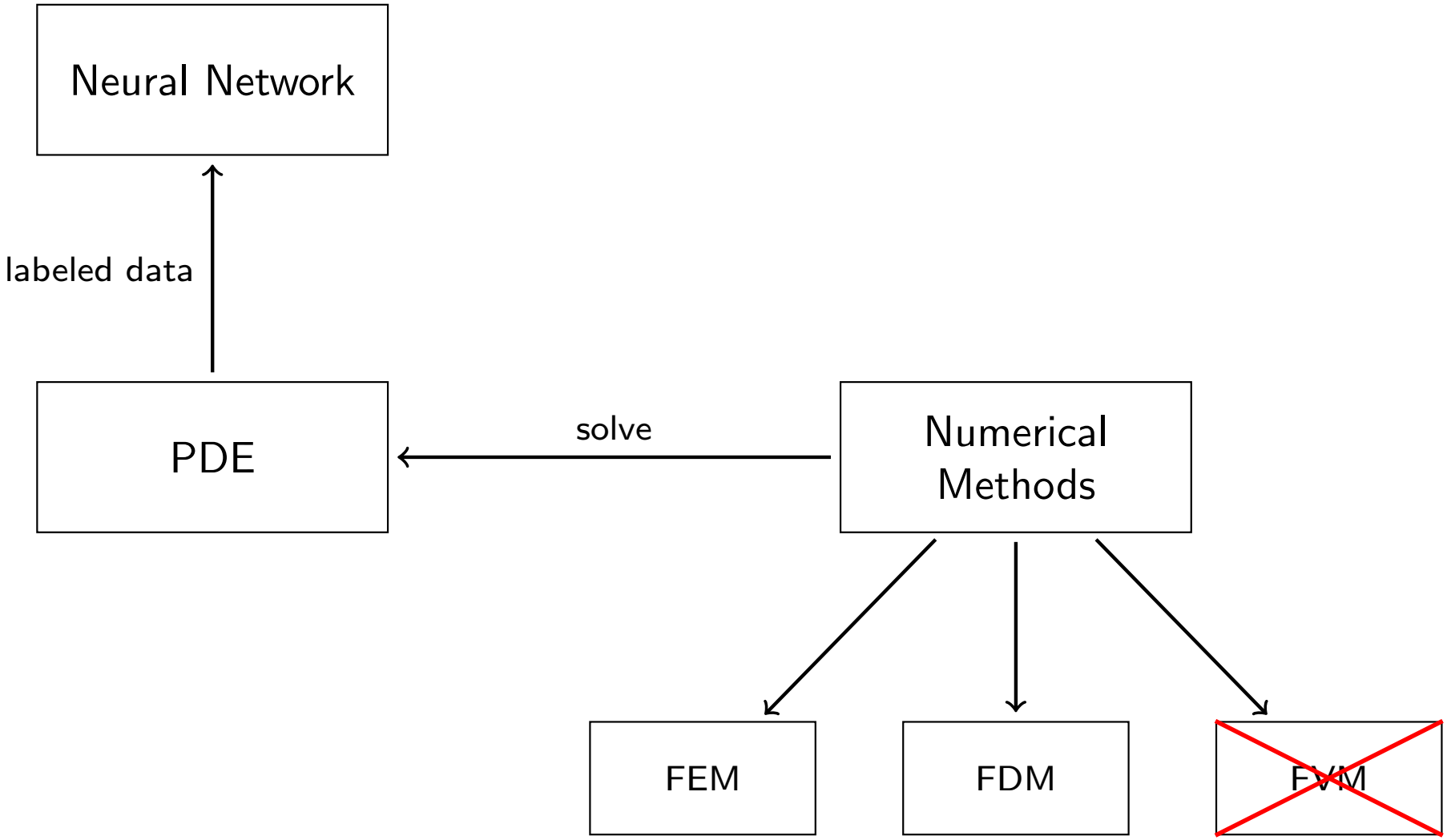
Introduction to the Finite Element Method (FEM)

Presented by: Dr. Yaohua Zang & Vincent Scholz

Professorship for Data-driven Materials Modeling

May 20, 2025

The Role of Numerical Methods in Our Workflow



- Numerical methods solve PDEs and generate labeled data
- This course focuses on FEM; FDM is mentioned; FVM is excluded

Overview

Table of Contents

What is the Finite Element Method?

- Powerful numerical technique for solving partial differential equations (PDEs)
- Constructs approximate solutions using piecewise functions
- Typically uses polynomials over subdomains called elements
- Particularly useful for complex geometries and boundary conditions

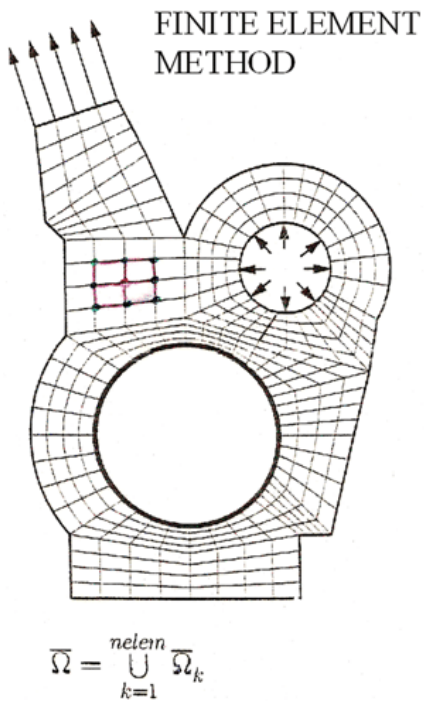


Figure: Conceptual representation of FEM discretization

Why FEM?

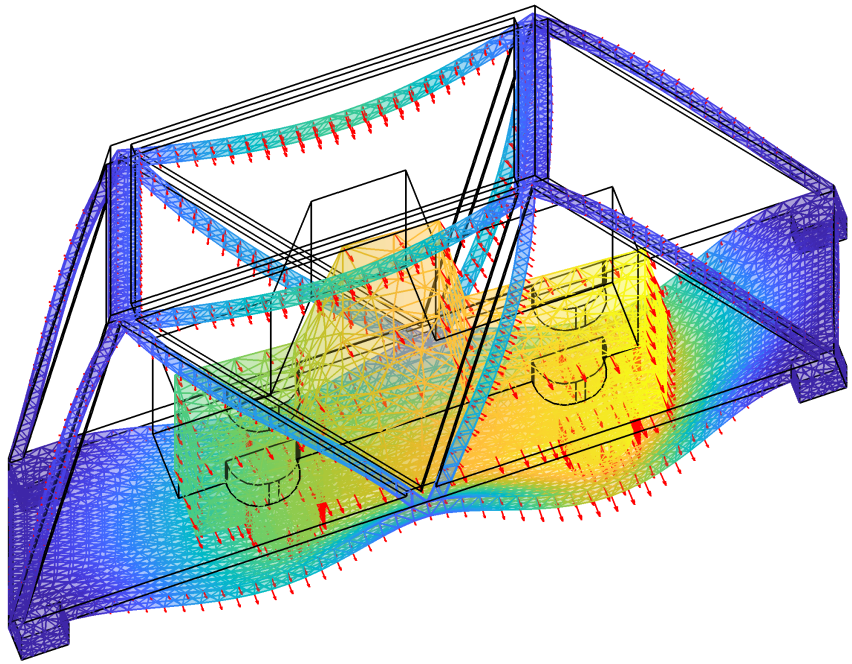
- Engineering structures: buildings, bridges, cars, biomedical devices
- Too complex to solve PDEs analytically
- Idea: **Break complex domain into small, manageable parts (elements)**

Analogy

Imagine trying to understand a jelly cube's deformation — it's too hard as a whole, so we slice it into smaller parts we can analyze!

Real-Life Applications of FEM

- Predict how bones react to stress
- Simulate airflow around a Formula 1 car
- Deformation of airplane wings
- Earthquake safety in buildings



FEM in Simple Terms

- **Discretize** the domain into small elements (triangles, quadrilaterals, etc.)
- Use simple basis functions to represent the solution in each element
- Formulate the problem so we only need to solve equations on these small elements

Concept Map of FEM

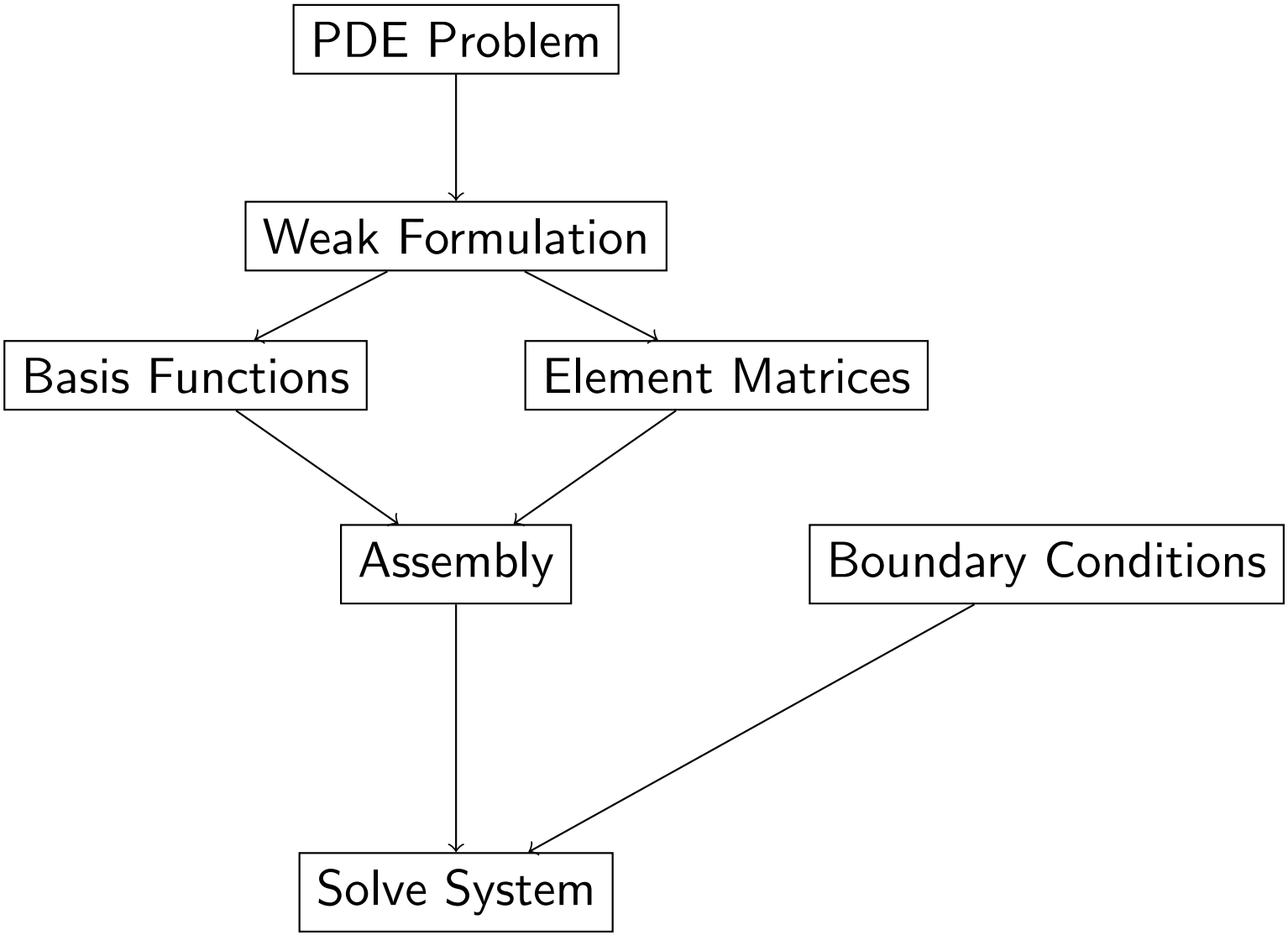


Table of Contents

The Strong Form

Poisson Equation in One Dimension

$$-k\frac{\partial^2 u}{\partial x^2} = f(x) \tag{1}$$

- k is a given coefficient
- u is the unknown function
- f is a known source function

What Can Be on the Right-Hand Side?

- **Body forces** (volume loads inside the domain)
  - Gravity: $f(x) = \rho g$
  - Electromagnetic forces: $f(x) = \rho E(x)$
- **Source terms** in diffusion, heat, or Poisson equations
  - Heat source: $f(x)$ is the heat generation rate
  - Charge density in electrostatics
- **Inertial terms** (in dynamic problems)
  - $f(x) = \rho \ddot{u}(x)$
- **Coupled field forcing** from other PDEs
  - Fluid-structure interaction: pressure fields from a fluid
  - Electromechanical: voltage as force in piezoelectricity

The Weak Form: Derivation (1)

1. Introduce a test function w from the function space $H^1(\Omega)$
2. Multiply both sides by w and integrate over domain Ω:

$$-\int_\Omega k\frac{\partial^2 u}{\partial x^2} w\, dx = \int_\Omega fw\, dx \tag{2}$$

3. Apply integration by parts:

$$\int_a^b h'g = \Big[hg\Big]_b^a - \int_a^b hg' \tag{3}$$

Intuition

We shift derivatives to test the function w to allow lower regularity of u

The Weak Form: Derivation (2)

$$\int_\Omega k\frac{\partial u}{\partial x}\cdot\frac{\partial w}{\partial x}\,dx - \left[w\frac{\partial u}{\partial x}\right]_{x_0}^{x_1} = \int_\Omega fw\,dx \tag{4}$$

$$\tag{5}$$

If we choose test functions that vanish on the boundary ($w \in H_0^1(\Omega)$), then the boundary term disappears:

$$\int_\Omega k\frac{\partial u}{\partial x}\cdot\frac{\partial w}{\partial x}\,dx = \int_\Omega fw\,dx \tag{6}$$

Table of Contents

Discretization of the Weak Formulation

- Discretize domain Ω into n nodes (indexed from 0 to $n-1$)
- Approximate unknown function u using a finite-dimensional subspace

Discrete Approximation

$$u_h = \sum_{i=0}^{n-1} u_i \cdot \psi_i(x) \tag{7}$$

where:

- u_i are unknown coefficients
- ψ_i are piecewise linear basis functions (shape functions)

Analogy

Each ψ_i is like a local influence zone for node i

Shape Functions

Shape functions (or base functions) ψ_i are defined as:

$$\psi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x_{i-1} \leq x \leq x_i \\ \frac{x_{i+1} - x}{x_{i+1} - x_i}, & x_i \leq x \leq x_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

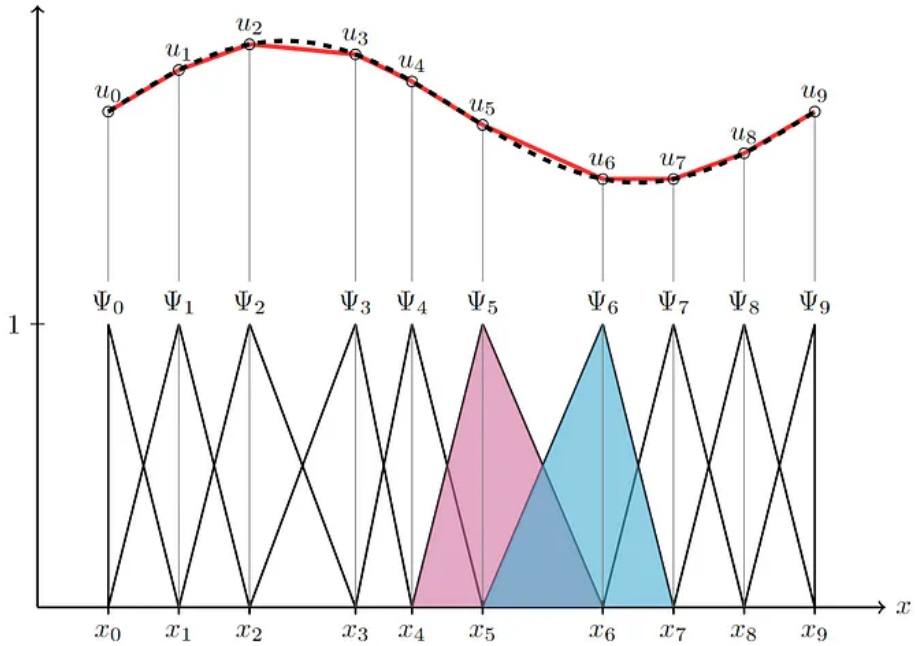


Figure: Illustration of shape functions and approximation

Approximation of Continuous Functions

- The continuous function u is approximated by the discrete function u_h
- Each node value u_i has an associated shape function ψ_i
- The resulting approximation is piecewise linear

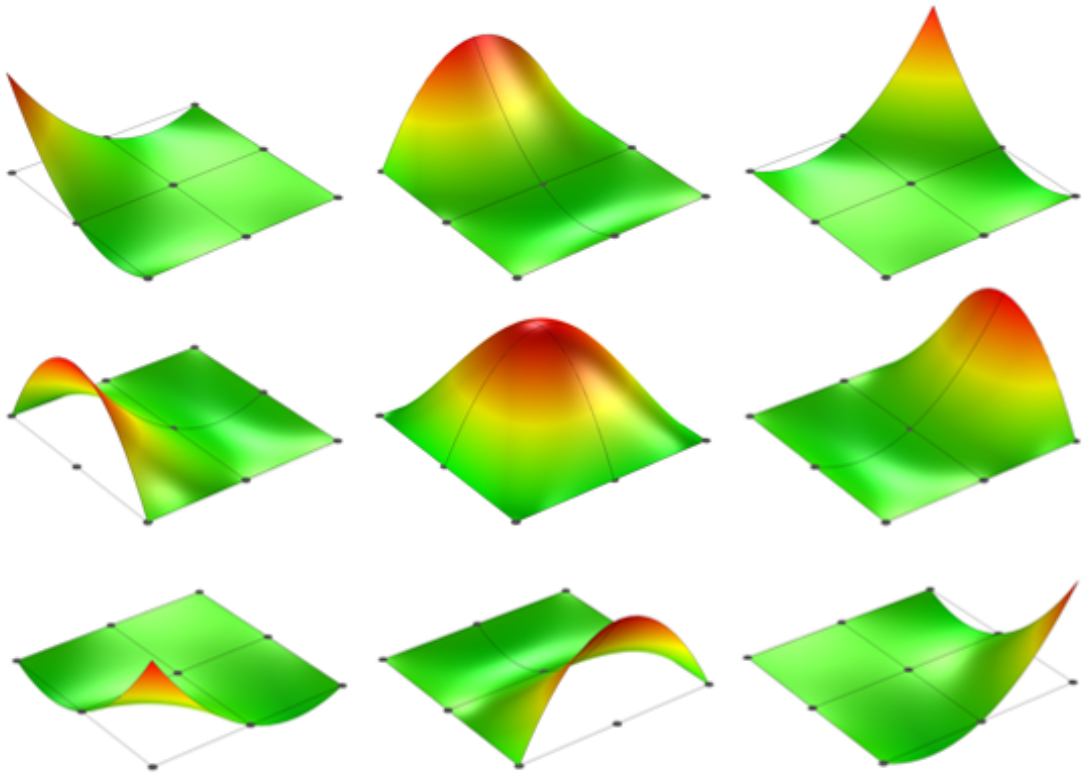


Figure: Discretization of function u using shape functions

Table of Contents

Galerkin Method and System Construction (1)

Using the Galerkin method, we use shape functions as test functions:

$$\int_{\Omega} k \left(\sum_{j=0}^{n-1} u_j \frac{\partial \psi_j}{\partial x} \right) \frac{\partial \psi_i}{\partial x} \, dx = \int_{\Omega} f(x) \psi_i \, dx \tag{9}$$

Rearranging:

$$\sum_{j=0}^{n-1} \underbrace{\left(\int_{\Omega} k \frac{\partial \psi_j}{\partial x} \frac{\partial \psi_i}{\partial x} \, dx \right)}_{A_{ij}} u_j = \underbrace{\int_{\Omega} f(x) \psi_i \, dx}_{b_i} \tag{10}$$

Galerkin Method and System Construction (2)

Linear System

$$A\vec{u} = b \tag{11}$$

where:

- $\vec{u} = [u_0, u_1, \cdots, u_{n-1}]^T$ are the unknowns
- A is the stiffness matrix
- b is the force vector

Intuition

Each A_e tells us how much element e resists deformation, A is the global deformation resistance

Stiffness Matrix and Force Vector

The entries of A and b are given by:

$$A_{ij} = \int_{\Omega} k \frac{\partial \psi_i}{\partial x} \frac{\partial \psi_j}{\partial x} \, dx \tag{12}$$

$$b_i = \int_{\Omega} f \psi_i \, dx \tag{13}$$

Note: Shape function ψ_i is only non-zero between x_{i-1} and x_{i+1}

Structure of the Stiffness Matrix

The stiffness matrix has a tridiagonal structure:

$$A_{ij} = \begin{cases} \frac{k}{x_i - x_{i-1}} + \frac{k}{x_{i+1} - x_i}, & i = j \\ -\frac{k}{x_i - x_{i-1}}, & j = i - 1 \\ -\frac{k}{x_{i+1} - x_i}, & j = i + 1 \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

For boundary cases ($i = 0, n$):

$$A_{0,0} = \frac{1}{x_1 - x_0} \tag{15}$$

$$A_{n,n} = \frac{1}{x_n - x_{n-1}} \tag{16}$$

- Local matrices mapped into global matrix
- Shared nodes imply shared equations

Structure of the Force Vector

For a constant source term $f(x) = 1$:

$$b_i = \frac{x_{i+1} - x_i}{2} + \frac{x_i - x_{i-1}}{2}, \quad i = 1, \cdots, n-1 \tag{17}$$

$$b_0 = \frac{x_1 - x_0}{2} \tag{18}$$

$$b_n = \frac{x_n - x_{n-1}}{2} \tag{19}$$

Complete Matrix System

The complete linear system in matrix form:

$$\begin{bmatrix} A_{0,0} & A_{0,1}, & 0, & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \cdots & -\frac{k}{x_i - x_{i-1}} & (A_{i,i}) & -\frac{k}{x_{i+1} - x_i} & \cdots \\ \vdots & \cdots & \ddots & \ddots & \vdots \\ \cdots & \cdots & 0 & A_{n-1,n-2} & A_{n-1,n-1} \end{bmatrix} \vec{u} = \begin{bmatrix} b_0 \\ \vdots \\ \frac{x_{i+1} - x_{i-1}}{2} \\ \vdots \\ b_{n-1} \end{bmatrix} \tag{20}$$

where $(A_{i,i}) = \frac{k}{x_i - x_{i-1}} + \frac{k}{x_{i+1} - x_i}$

Solving the Linear System

Once the entries for matrix A and vector b are filled out:

- The hard work is done
- The direct solution is defined as: $u = A^{-1}b$
- Other methods include:
  - Iterative solvers (Jacobi, Gauss-Seidel)
  - Conjugate gradient method
  - Multigrid methods

Table of Contents

Types of Boundary Conditions

Two most common boundary conditions for PDEs:

1. Dirichlet boundary conditions: Specify the value of u on the boundary
2. Neumann boundary conditions: Specify the derivative of u on the boundary

Intuition

- Dirichlet: fix a node (e.g., $u = 0$ on boundary)
- Neumann: apply a force or gradient

Dirichlet Boundary Conditions

Definition

Specifies the exact value of u on the boundary:

$$u(x) = g, \quad x \in \partial\Omega \tag{21}$$

- For 1D case: $u(x_0) = u_0$ and $u(x_{n-1}) = u_{n-1}$
- First and last equations become redundant
- System size reduces to $(n-2) \times (n-2)$

Modified System with Dirichlet Conditions

Modification of the second row:

$$A_{1,0}u_0 + A_{1,1}u_1 + A_{1,2}u_2 = b_1 \tag{22}$$

$$\Rightarrow A_{1,1}u_1 + A_{1,2}u_2 = b_1 - A_{1,0}u_0 \tag{23}$$

Similar modification for the last second row:

$$A_{n-2,n-3}u_{n-3} + A_{n-2,n-2}u_{n-2} + A_{n-2,n-1}u_{n-1} = b_{n-2} \tag{24}$$

$$\Rightarrow A_{n-2,n-2}u_{n-2} + A_{n-2,n-1}u_{n-1} = b_{n-2} - A_{n-2,n-3}u_{n-3} \tag{25}$$

Neumann Boundary Conditions

Definition

Specifies the derivative value on the boundary:

$$\frac{\partial u}{\partial \vec{n}} = \nabla u \cdot \vec{n} = g, \quad x \in \partial\Omega \tag{26}$$

- $\vec{n}$ is the normal vector on the boundary
- In 1D: $\vec{n} = -1$ on left boundary, $\vec{n} = 1$ on right boundary
- Conditions: $-u'(x_0) = g(x_0)$ and $u'(x_{n-1}) = g(x_{n-1})$

Implementation of Neumann Conditions

Neumann conditions modify the right-hand side vector b:

$$\left[w\frac{\partial u}{\partial x}\right]_{x_0}^{x_{n-1}} = w(x_{n-1})u'(x_{n-1}) - w(x_0)u'(x_0) \tag{27}$$

$$= w(x_{n-1})g(x_{n-1}) + w(x_0)g(x_0) \tag{28}$$

- For shape functions, $w(x_0) = w(x_{n-1}) = 1$
- Only the first and last entries of vector b are modified
- System size remains $n \times n$

Modified System with Neumann Conditions

Matrix equation system with Neumann conditions:

$$\begin{bmatrix} A_{0,0} & A_{0,1}, & 0, & \cdots & \cdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \cdots & \cdots & (A_{i,i}) & \cdots & \cdots \\ \vdots & \cdots & \ddots & \ddots & \vdots \\ \cdots & \cdots & 0 & A_{n-1,n-2} & A_{n-1,n-1} \end{bmatrix} \vec{u} = \begin{bmatrix} b_0 + g(x_0) \\ b_1 \\ \vdots \\ \vdots \\ b_{n-1} + g(x_{n-1}) \end{bmatrix} \tag{29}$$

where $(A_{i,i}) = \frac{k}{x_i - x_{i-1}} + \frac{k}{x_{i+1} - x_i}$

Table of Contents

Summary

- FEM is a powerful method for solving PDEs, especially with complex geometries
- Key steps in the FEM process:
  1. Convert strong form to weak form
  2. Discretize the domain and construct shape functions
  3. Formulate linear system (stiffness matrix and force vector)
  4. Apply boundary conditions
  5. Solve the resulting linear system
- Result is a piecewise approximation of the solution