Part II – Numerical methods for elliptic PDEs

MAT684 – Spring 2026 – Shukai Du

Topics:

- Finite difference method
- Finite element method
- Stability and error analysis

1 Finite difference methods

In this part, we investigate numerical methods for partial differential equations, with a focus on *elliptic* PDEs. These equations describe the spatial distribution of fields in steady-state systems. Typical examples include the temperature distribution in a room, the stress–strain distribution in a solid structure such as a bridge, and the distribution of electric or magnetic fields in a medium. In the next part, we will turn to numerical methods for time-dependent systems. The concepts and techniques developed here will serve as a foundation.

1.1 Finite difference method in 1D

Let us begin by considering a simple model problem:

$$-\partial_{xx} u = f \quad \text{in } [0, L], \tag{1a}$$

$$u(0) = u_0, \qquad u(L) = u_L. \tag{1b}$$

Here, u may represent, for instance, the (steady-state) temperature distribution along a thin rod of length L, giving fixed boundary temperature of u_0 and u_L.

A finite difference method starts by partitioning the interval $[0, L]$ into N subintervals:

$$0 = x_0 < x_1 < \cdots < x_N = L,$$

where $x_i = ih$ and $h = L/N$.

The key idea of finite difference methods is to replace derivatives by linear combinations of the *function values* at discrete points. For example, we approximate the first derivative at the midpoints by

$$\partial_x u(x_{i+1/2}) \approx \frac{u(x_{i+1}) - u(x_i)}{h},$$

$$\partial_x u(x_{i-1/2}) \approx \frac{u(x_i) - u(x_{i-1})}{h},$$

and hence

$$\begin{aligned} \partial_{xx} u(x_i) = \partial_x\big(\partial_x u\big)(x_i) &\approx \frac{\partial_x u(x_{i+1/2}) - \partial_x u(x_{i-1/2})}{h} \\ &\approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}. \end{aligned}$$

The above approximation naturally leads to the following finite difference method for computing a numerical solution to (1):

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = f(x_i) \quad \text{for } i = 1, 2, \ldots, N-1. \tag{2}$$

This is called the *centered difference scheme*. Since it couples the value at x_i with its two nearest neighbors, it is also referred to as the *three-point central difference scheme*.

Note that we have $N+1$ unknowns but only $N-1$ equations. However, we already know the boundary values

$$u_0 = u(0), \qquad u_N = u(L).$$

Therefore, we obtain a square system with $N-1$ unknowns and $N-1$ equations.

Example. Let us consider an example of (1). We interpret $u(x)$ as the steady-state temperature along a thin rod of length $L = 1$, with the endpoint temperatures fixed at

$$u(0) = 1, \qquad u(1) = 2.$$

We assume a uniform internal heat source throughout the rod, modeled by a constant source term

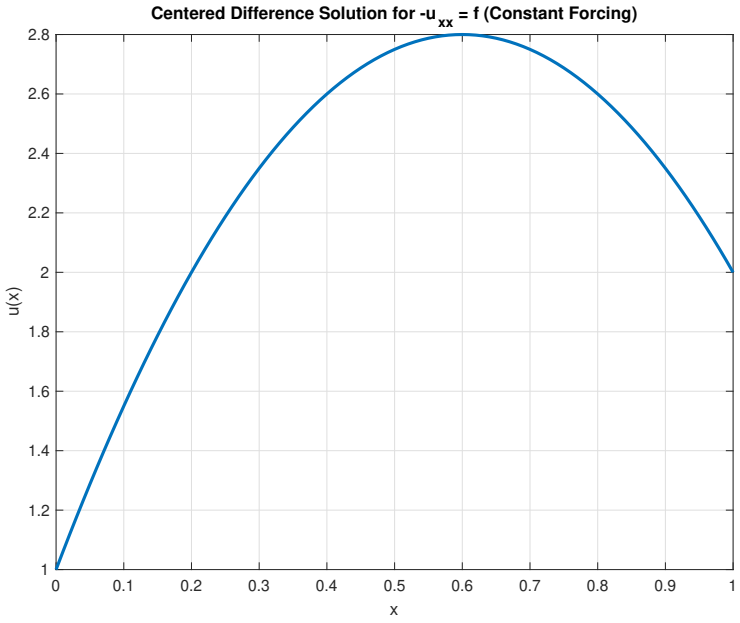
$$f(x) = 10, \qquad x \in [0, 1].$$

- Code

```
clc; clear; close all;

% Parameters
L  = 1;          % domain length
N  = 100;        % number of subintervals
h  = L / N;

x = linspace(0, L, N+1)';

% Boundary conditions
uL = 1;   % u(0)
uR = 2;   % u(L)

% Constant forcing
f0 = 10;

% Assemble RHS (interior points)
f = f0 * ones(N-1, 1);

% Assemble tridiagonal matrix A
e = ones(N-1, 1);
A = (1/h^2) * spdiags([-e 2*e -e], -1:1, N-1, N-1);

% Incorporate boundary conditions into RHS
b = f;
b(1)   = b(1)   + uL / h^2;
```

```
b(end) = b(end) + uR / h^2;

% Solve linear system
u_inner = A \ b;

% Full solution including boundaries
u = zeros(N+1, 1);
u(1)       = uL;
u(end)     = uR;
u(2:end-1) = u_inner;

% Plot
figure;
plot(x, u, 'LineWidth', 2);
xlabel('x');
ylabel('u(x)');
title('Centered Difference Solution for -u_{xx} = f (Constant Forcing)');
grid on;
```

- Results



Local truncation error. It is possible to estimate the local truncation error of the finite difference scheme in a fashion similar to what we did for time-stepping methods.

By replacing u_i with the exact solution value $u(x_i)$, we define the local truncation error as

$$\mathrm{LTE}_i := -\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - f(x_i).$$

Using Taylor expansions of $u(x_{i\pm1})$ about x_i, we obtain

$$u(x_{i+1}) = u(x_i) + h\,u'(x_i) + \frac{h^2}{2}u''(x_i) + \frac{h^3}{6}u^{(3)}(x_i) + \frac{h^4}{24}u^{(4)}(\xi_{i+1}),$$
$$u(x_{i-1}) = u(x_i) - h\,u'(x_i) + \frac{h^2}{2}u''(x_i) - \frac{h^3}{6}u^{(3)}(x_i) + \frac{h^4}{24}u^{(4)}(\xi_{i-1}),$$

for some $\xi_{i+1} \in (x_i, x_{i+1})$ and $\xi_{i-1} \in (x_{i-1}, x_i)$.

Substituting into the finite difference expression, we find

$$-\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} = -u''(x_i) + \mathcal{O}(h^2).$$

Since the exact solution satisfies $-u''(x_i) = f(x_i)$, it follows that

$$\mathrm{LTE}_i = \mathcal{O}(h^2),$$

provided that u is sufficiently smooth (in particular, $u \in C^4([0, L])$).

Remark 1.1. *A higher-order local truncation error can be achieved by using a wider stencil. For example, consider*

$$-\frac{a_1u_{i-2} + a_2u_{i-1} + a_3u_i + a_4u_{i+1} + a_5u_{i+2}}{h^2} = f(x_i),$$

where a_i $(i = 1, \ldots, 5)$ are fixed constants chosen to match the Taylor expansion to as high an order as possible. This is known as the five-point central difference scheme.

Exercise. Determine the coefficients $a_1, \ldots, a_5$ so that the local truncation error has the highest possible order. What is this order?

Global error. To obtain a global error estimate, we first establish a stability result for the centered difference scheme (2). The following lemma mimics the classical Poincaré inequality, which states that a function can be controlled by its derivatives under appropriate boundary conditions.

Lemma 1.1 (Poincaré inequality in 1D)**.** *Suppose $u \in C^1([0, L])$ satisfies $u(0) = u(L) = 0$. Then*

$$\int_0^L u(x)^2\,dx \;\le\; C\int_0^L |u'(x)|^2\,dx,$$

where the constant C depends only on L.

Lemma 1.2 (Discrete Poincaré inequality in 1D)**.** *Let $\{v_i\}_{i=0}^N$ satisfy $v_0 = v_N = 0$ on a uniform grid $x_i = ih$ with $h = L/N$. Then*

$$h\sum_{i=1}^{N-1} v_i^2 \;\le\; L^2\,h\sum_{i=0}^{N-1}\left(\frac{v_{i+1} - v_i}{h}\right)^2.$$

Proof. Since $v_0 = 0$, for each $i = 1, \dots, N-1$ we have the telescoping representation

$$v_i = \sum_{k=0}^{i-1} (v_{k+1} - v_k).$$

By Cauchy–Schwarz inequality,

$$v_i^2 \le \Big(\sum_{k=0}^{i-1} 1^2 \Big) \Big(\sum_{k=0}^{i-1} (v_{k+1} - v_k)^2 \Big) = i \sum_{k=0}^{i-1} (v_{k+1} - v_k)^2 \le N \sum_{k=0}^{N-1} (v_{k+1} - v_k)^2.$$

Summing over $i = 1, \dots, N-1$ gives

$$\sum_{i=1}^{N-1} v_i^2 \le (N-1)\, N \sum_{k=0}^{N-1} (v_{k+1} - v_k)^2 \le N^2 \sum_{k=0}^{N-1} (v_{k+1} - v_k)^2.$$

Multiplying by h and using $N = L/h$ yields

$$h \sum_{i=1}^{N-1} v_i^2 \le h N^2 \sum_{k=0}^{N-1} (v_{k+1} - v_k)^2 = \frac{L^2}{h} \sum_{k=0}^{N-1} (v_{k+1} - v_k)^2 = L^2\, h \sum_{k=0}^{N-1} \left(\frac{v_{k+1} - v_k}{h} \right)^2,$$

which proves the claim. □

Suppose $v(0) = v(L) = 0$. Recall the integration by parts formula:

$$\begin{aligned} \int_0^L -\partial_{xx} v(x)\, v(x)\, dx &= -\int_0^L v(x)\, d(v'(x)) \\ &= v(x) v'(x) \Big|_0^L + \int_0^L \big(v'(x) \big)^2 dx \\ &= \int_0^L \big(v'(x) \big)^2 dx. \end{aligned}$$

Let us replace $\partial_{xx} v$ by the centered difference formula $\frac{v_{i+1} - 2v_i + v_{i-1}}{h^2}$, replace $v'(x)$ by the forward difference $\frac{v_{i+1} - v_i}{h}$, and replace the integral by a summation. This motivates the discrete identity:

$$\sum_{i=1}^{N-1} \Big(-\frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} \Big) v_i\, h = \sum_{i=0}^{N-1} \Big(\frac{v_{i+1} - v_i}{h} \Big)^2 h.$$

Lemma 1.3 (Discrete summation by parts)**.** *Let* $\{v_i\}_{i=0}^N$ *satisfy* $v_0 = v_N = 0$*. Then*

$$\sum_{i=1}^{N-1} -\big(v_{i+1} - 2v_i + v_{i-1} \big)\, v_i = \sum_{i=0}^{N-1} (v_{i+1} - v_i)^2.$$

Proof. Expanding the left-hand side and shifting indices,

$$\begin{aligned}\sum_{i=1}^{N-1} -\big(v_{i+1}-2v_i+v_{i-1}\big)v_i &= \sum_{i=1}^{N-1}\big(2v_i^2 - v_{i+1}v_i - v_{i-1}v_i\big)\\ &= 2\sum_{i=1}^{N-1} v_i^2 - \sum_{i=1}^{N-1} v_{i+1}v_i - \sum_{i=1}^{N-1} v_{i-1}v_i\\ &= 2\sum_{i=1}^{N-1} v_i^2 - \sum_{i=1}^{N-1} v_{i+1}v_i - \sum_{i=0}^{N-2} v_{i+1}v_i\\ &= 2\sum_{i=1}^{N-1} v_i^2 - 2\sum_{i=1}^{N-2} v_{i+1}v_i,\end{aligned}$$

where in the last step we used $v_0 = v_N = 0$ to drop the endpoint terms v_1v_0 and v_Nv_{N-1}.

On the other hand,

$$\begin{aligned}\sum_{i=0}^{N-1}(v_{i+1}-v_i)^2 &= \sum_{i=0}^{N-1}\big(v_{i+1}^2 - 2v_{i+1}v_i + v_i^2\big)\\ &= \sum_{i=1}^{N-1} v_i^2 + \sum_{i=1}^{N-1} v_i^2 - 2\sum_{i=1}^{N-2} v_{i+1}v_i\\ &= 2\sum_{i=1}^{N-1} v_i^2 - 2\sum_{i=1}^{N-2} v_{i+1}v_i,\end{aligned}$$

again using $v_0 = v_N = 0$. Comparing the two expressions yields the identity. □

Theorem 1.1 (Second-order global error)**.** *Let $u \in C^4([0,L])$ solve (1), and let $\{u_i\}_{i=0}^N$ be the centered finite difference solution with $u_0 = u(0)$ and $u_N = u(L)$. Define $e_i := u(x_i) - u_i$. If the local truncation error satisfies*

$$\|\tau\|_{\ell_\infty} \le C_\tau h^2, \qquad \tau_i := -\frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2} - f(x_i),$$

then there exists a constant $C > 0$, independent of h, such that

$$\|e\|_{\ell_\infty} \le Ch^2.$$

Proof. Step 1: Error equation. For $i = 1, \dots, N-1$,

$$-\frac{e_{i+1} - 2e_i + e_{i-1}}{h^2} = \tau_i, \qquad e_0 = e_N = 0.$$

Equivalently,

$$-\big(e_{i+1} - 2e_i + e_{i-1}\big) = h^2\tau_i. \tag{3}$$

Step 2: Discrete energy identity. Multiply (3) by e_i and sum over $i = 1, \dots, N-1$ to obtain

$$\sum_{i=1}^{N-1} -\big(e_{i+1} - 2e_i + e_{i-1}\big)e_i = h^2 \sum_{i=1}^{N-1} \tau_i e_i.$$

By Lemma 1.3 (applied to $v_i = e_i$), the left-hand side equals $\sum_{i=0}^{N-1}(e_{i+1}-e_i)^2$. Therefore,

$$\sum_{i=0}^{N-1}(e_{i+1}-e_i)^2 = h^2\sum_{i=1}^{N-1}\tau_i e_i. \tag{4}$$

Step 3: Poincaré and Cauchy–Schwarz. By Cauchy–Schwarz,

$$h^2\sum_{i=1}^{N-1}\tau_i e_i \le h^2\Big(\sum_{i=1}^{N-1}\tau_i^2\Big)^{1/2}\Big(\sum_{i=1}^{N-1}e_i^2\Big)^{1/2}.$$

Using Lemma 1.2 (since $e_0 = e_N = 0$), we obtain

$$\Big(\sum_{i=1}^{N-1}e_i^2\Big)^{1/2} \le \frac{L}{h}\Big(\sum_{i=0}^{N-1}(e_{i+1}-e_i)^2\Big)^{1/2}.$$

Combining with (4) and canceling the common factor $\big(\sum_{i=0}^{N-1}(e_{i+1}-e_i)^2\big)^{1/2}$ yields

$$\Big(\sum_{i=0}^{N-1}(e_{i+1}-e_i)^2\Big)^{1/2} \le Lh\Big(\sum_{i=1}^{N-1}\tau_i^2\Big)^{1/2}. \tag{5}$$

Step 4: Conversion to ℓ_∞. Since $e_0 = 0$, for any i,

$$|e_i| = \Big|\sum_{k=0}^{i-1}(e_{k+1}-e_k)\Big| \le \sum_{k=0}^{N-1}|e_{k+1}-e_k| \le \sqrt{N}\Big(\sum_{k=0}^{N-1}(e_{k+1}-e_k)^2\Big)^{1/2}.$$

Therefore,

$$\|e\|_{\ell_\infty} \le \sqrt{N}\Big(\sum_{k=0}^{N-1}(e_{k+1}-e_k)^2\Big)^{1/2}.$$

Using (5) and

$$\Big(\sum_{i=1}^{N-1}\tau_i^2\Big)^{1/2} \le \sqrt{N}\,\|\tau\|_{\ell_\infty},$$

we obtain

$$\|e\|_{\ell_\infty} \le \sqrt{N}\cdot Lh\Big(\sum_{i=1}^{N-1}\tau_i^2\Big)^{1/2} \le \sqrt{N}\cdot Lh\cdot\sqrt{N}\,\|\tau\|_{\ell_\infty} = (Nh)\,L\,\|\tau\|_{\ell_\infty} = L^2\,\|\tau\|_{\ell_\infty}.$$

Finally, using $\|\tau\|_{\ell_\infty} \le C_\tau h^2$ yields

$$\|e\|_{\ell_\infty} \le L^2 C_\tau h^2.$$

□

1.2 Finite difference method in 2D

In this section, we shall consider the second-order elliptic equation (1) in higher dimension, in which case things become more interesting. To be more specific, we consider the following equation:

$$-(\partial_{xx} + \partial_{yy})u =: -\Delta u = f \quad \text{in} \quad \Omega := [0, L_x] \times [0, L_y], \tag{6a}$$

$$u = g \quad \text{on} \quad \partial\Omega. \tag{6b}$$

For instance, this model can describe the steady-state heat distribution on a thin plate, where g represents the fixed temperature on the boundary and f represents internal heat sources. It can also be used to model electrostatic potential, membrane displacement under load, or steady diffusion processes.

Similar to the 1D case, we first partition the domain Ω using a uniform Cartesian grid:

$$x_i = ih_x, \quad i = 0, 1, \ldots, N_x, \qquad y_j = jh_y, \quad j = 0, 1, \ldots, N_y,$$

where $h_x = L_x/N_x$ and $h_y = L_y/N_y$ represent the mesh/grid sizes. We denote the numerical approximation by

$$u_{i,j} \approx u(x_i, y_j).$$

With the grid, we then approximate the Laplacian at an interior grid point (x_i, y_j) by the following central difference scheme:

$$\Delta u(x_i, y_j) \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2}.$$

Namely, we simply replace $\partial_{xx}u(x_i, y_j)$ and $\partial_{yy}u(x_i, y_j)$ by the corresponding 1D three-point *stencil* central difference scheme. Therefore, for interior nodes $1 \le i \le N_x - 1$ and $1 \le j \le N_y - 1$, we obtain the following discrete equations:

$$-\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2}\right) = f_{i,j}. \tag{7}$$

Remark 1.2. *When $h_x = h_y = h$, we obtain the following classical five-point stencil central difference scheme in 2D:*

$$-\Delta_h u_{i,j} = \frac{1}{h^2}\left(4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}\right) = f_{i,j}. \tag{8}$$

We shall use this version for the rest of the discussion for simplicity in demonstrating the main ideas.

On boundary nodes, the Dirichlet condition is imposed directly:

$$u_{i,j} = g(x_i, y_j), \qquad (x_i, y_j) \in \partial\Omega.$$

Thus, the total number of unknown variables is given by the interior nodes $i = 1, 2, \ldots, N_x - 1$ and $j = 1, 2, \ldots, N_y - 1$, which matches the number of equations in (7) or (8).

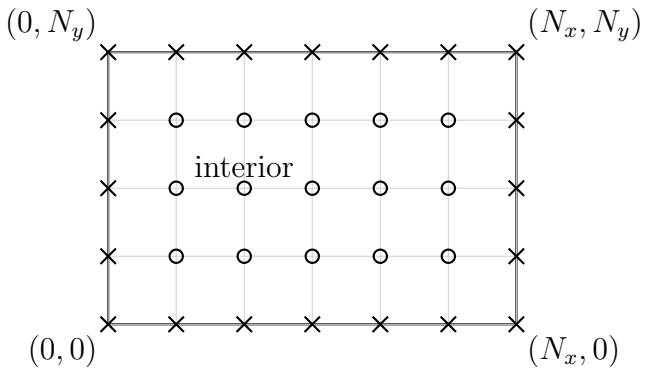
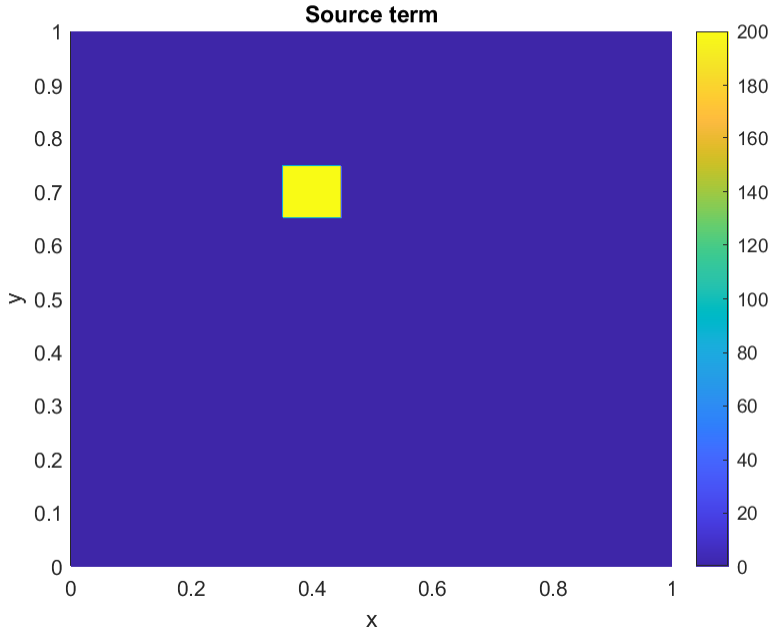


Figure 1: '×' denotes boundary nodes whose values are prescribed by the boundary data g. '○' denotes interior unknowns to be solved from (8).

Example. Let us consider an example of (6). We choose $L_x = L_y = 1$. The source term f is defined by

$$f(x, y) = 200\,(x > 0.35)\,(x < 0.45)\,(y > 0.65)\,(y < 0.75),$$

which mimics a localized heat source supported in the rectangular region $(0.35, 0.45) \times (0.65, 0.75)$.



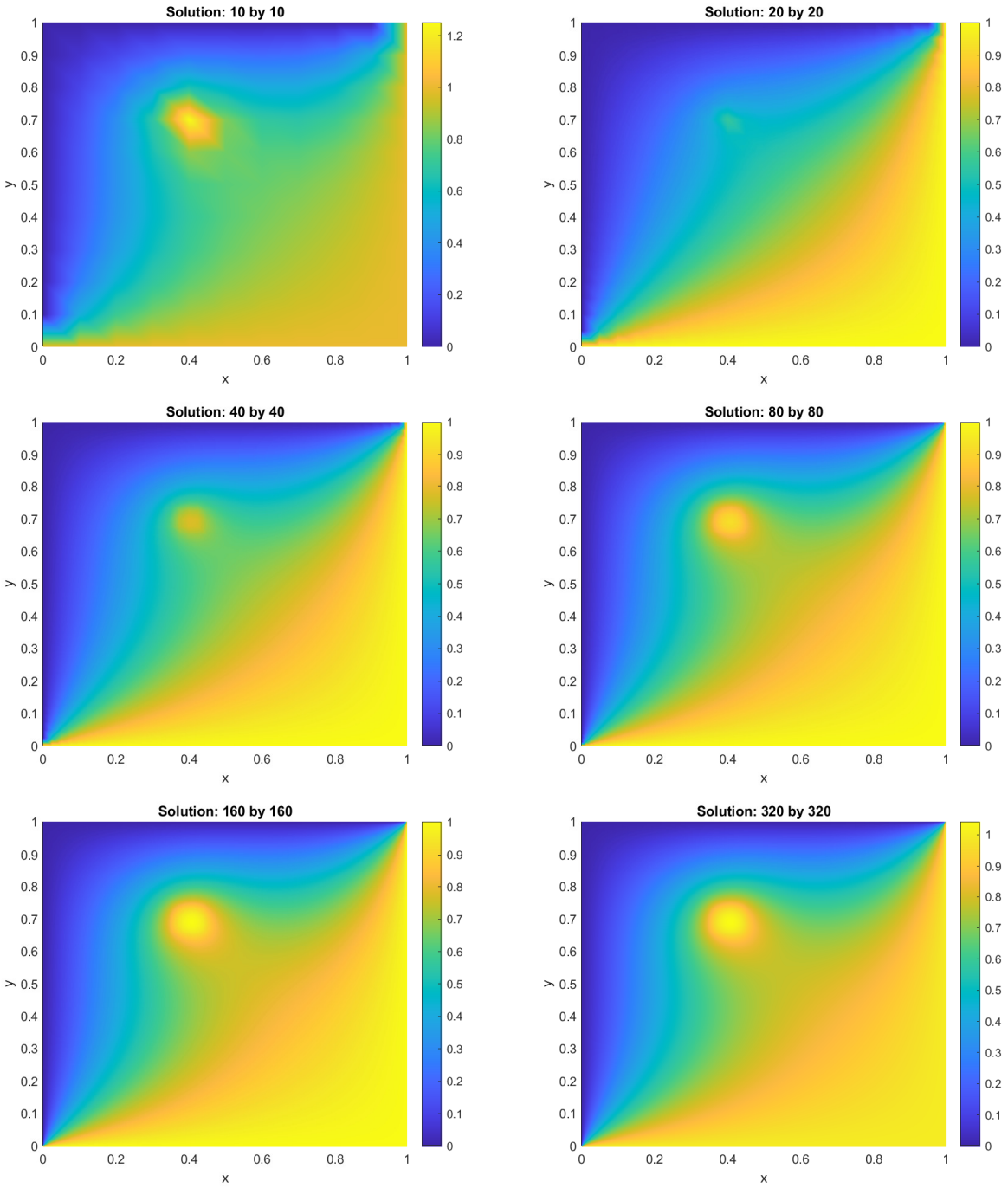
In addition, we impose Dirichlet boundary data such that $g = 1$ on the bottom and right sides of $\partial\Omega$, and $g = 0$ on the top and left sides. This mimics a contrast in the temperature distribution along the boundary walls.

- Code

```
% 2D Poisson: -Delta u = f on [0,Lx]x[0,Ly], Dirichlet u=g on boundary

clear; clc; close all

% --- parameters ---
Lx = 1; Ly = 1;
Nx = 10; Ny = 10;                       % intervals -> nodes are (Nx+1)x(Ny+1)
hx = Lx/Nx; hy = Ly/Ny;

% --- data ---
f = @(x,y) 200*(x>0.35).*(x<0.45).*(y>0.65).*(y<0.75);
g = @(x,y) double((y==0) | (x==1));

% --- grid (ndgrid) ---
x = linspace(0,Lx,Nx+1);
y = linspace(0,Ly,Ny+1);
[X,Y] = ndgrid(x,y);

% --- global indexing for ALL nodes (i=1..Nx+1, j=1..Ny+1) ---
idx = @(i,j) (j-1)*(Nx+1) + i;          % i varies fastest
Ntot = (Nx+1)*(Ny+1);

A = spalloc(Ntot, Ntot, 5*Ntot);
b = zeros(Ntot,1);

% --- assemble full system + mark Dirichlet DOFs on the fly ---
dir = false(Ntot,1);

for j = 1:Ny+1
    for i = 1:Nx+1
        k = idx(i,j);

        isBoundary = (i==1) || (i==Nx+1) || (j==1) || (j==Ny+1);

        if isBoundary
            dir(k) = true;                      % mark Dirichlet DOF
            b(k)   = g(X(i,j), Y(i,j));
        else
            A(k,k) = 2/hx^2 + 2/hy^2;
            A(k, idx(i-1,j)) = -1/hx^2;
            A(k, idx(i+1,j)) = -1/hx^2;
            A(k, idx(i,j-1)) = -1/hy^2;
            A(k, idx(i,j+1)) = -1/hy^2;

            b(k) = f(X(i,j), Y(i,j));
        end
    end
end

% --- free DOFs are the complement of Dirichlet DOFs ---
free = ~dir;

u = zeros(Ntot,1);
u(dir) = b(dir);                         % u_dir = g on boundary
```

```
% --- eliminate Dirichlet DOF ---
Aff = A(free, free);
Afd = A(free, dir);

u(free) = Aff \ ( b(free) - Afd*u(dir) );

% --- reshape back to grid and visualize ---
U = reshape(u, [Nx+1, Ny+1]);

surf(X,Y,U); shading interp; view([0,90])
title(['Solution: ',num2str(Nx),' by ',num2str(Ny)]);
xlabel x; ylabel y;colorbar;

% --- visualize the source term ---
F = f(X,Y);

figure;
surf(X,Y,F); shading interp; view([0,90])
title('Source term');
xlabel x; ylabel y;
colorbar;
```

- Results



2 Finite element methods

In this section, we discuss finite element methods (FEM), one of the most important numerical methods for solving partial differential equations (PDEs). FEM allows us to solve PDEs on complicated geometries with theoretically guaranteed convergence under suitable assumptions.

The derivation of the finite element method follows two key ideas:

1. **Divide and conquer.** To handle complicated geometries, FEM first decomposes the computational domain into a collection of simple geometric objects called *elements* (e.g., intervals, triangles, tetrahedra). On each element, the solution is approximated by a polynomial (or other basis functions with good computational properties), leading to a *piecewise-polynomial* approximation over the whole domain.
2. **Variational principle.** For many important physical systems, the solution state can be characterized as the minimizer (or stationary point) of a functional, often called an *action* (or energy). Setting the first variation to zero yields the *weak formulation*, which forms the basis for finite element discretizations. One advantage is that the weak formulation typically requires only one order of derivatives to be well-defined, thereby reducing the differentiability requirements on the basis functions.

2.1 Finite element in 1D

To illustrate the key ideas of finite element, let us again begin by considering the 1D case. We then extend to higher dimensional case later. Let us consider the following second-order ellpitic equations:

$$-\partial_x(a(x)\,\partial_x u(x)) = f(x) \quad \text{in } [0, L], \tag{9a}$$

$$u(0) = g_0, \qquad u(L) = g_L. \tag{9b}$$

where $a \geq a_0 > 0$ is the coefficient for the equation. For instance, when (9) models the heat distribution along a thin rod, a represents the heat conducting property of the rod.

Variational formulation. We first introduce the variational formulation of (9). This formulation provides a fundamental viewpoint from which the PDE system (9) can be derived.

We begin by introducing the so-called *action functional*. For different physical systems, the form of the action may vary. For the present second-order elliptic problem, the action is defined by

$$\mathcal{A}(u) = \frac{1}{2}\int_0^L a(x)\,|u'(x)|^2\,dx - \int_0^L f(x)\,u(x)\,dx.$$

The solution is the function u that minimizes the action, subject to the boundary conditions:

$$u = \operatorname{argmin}_u \mathcal{A}(u) \quad \text{subject to} \quad u(0) = g_0, \quad u(L) = g_L. \tag{10}$$

To characterize this minimizer, we compute the first variation of $\mathcal{A}$. Suppose u is a minimizer of (10). Consider a small perturbation δu such that $u + \delta u$ satisfies the same boundary conditions. Hence, $\delta u(0) = \delta u(L) = 0$. Then,

$$\begin{aligned}\mathcal{A}(u+\delta u) - \mathcal{A}(u) &= \frac{1}{2}\int_0^L a(x)\Big(|u' + \delta u'|^2 - |u'|^2\Big)\,dx - \int_0^L f(x)\,\delta u(x)\,dx \\ &= \int_0^L a(x)\,u'(x)\,\delta u'(x)\,dx - \int_0^L f(x)\,\delta u(x)\,dx + \frac{1}{2}\int_0^L a(x)|\delta u'(x)|^2.\end{aligned}$$

The last term is quadratic in $\delta u'$ and therefore small when δu is small. Hence, the *first variation* is given by

$$\mathcal{A}'(u)[\delta u] = \int_0^L a(x)\, u'(x)\, \delta u'(x)\, dx - \int_0^L f(x)\, \delta u(x)\, dx.$$

(For a finite-dimensional vector u, this corresponds to the gradient: $\mathcal{A}'(u)[\delta u] = \nabla_u \mathcal{A}(u) \cdot \delta u$.)

Since u is a minimizer, the first variation must vanish for all admissible perturbations δu. Therefore,

$$\int_0^L a(x)\, u'(x)\, \delta u'(x)\, dx = \int_0^L f(x)\, \delta u(x)\, dx \qquad \forall\, \delta u \text{ with } \delta u(0) = \delta u(L) = 0.$$

We can equivalently write

$$\int_0^L a(x)\, u'(x)\, v'(x)\, dx = \int_0^L f(x)\, v(x)\, dx \qquad \forall\, v \text{ with } v(0) = v(L) = 0. \tag{11}$$

The identity (11) is called the *weak formulation* of the problem, which serves as the starting point for the finite element discretization.

Applying integration by parts to the left term of (11) gives

$$-\int_0^L (a(x)u'(x))'\, v(x)\, dx = \int_0^L f(x)\, v(x)\, dx.$$

Since this holds for all v satisfying $v(0) = v(L) = 0$, we obtain the Euler–Lagrange equation

$$-(a(x)u'(x))' = f(x), \qquad x \in (0, L),$$

together with the boundary conditions $u(0) = g_0$ and $u(L) = g_L$. This is precisely the PDE system (9).

Remark 2.1. *The Hamiltonian system*

$$\dot{q} = \frac{\partial H(q,p)}{\partial p}, \qquad \dot{p} = -\frac{\partial H(q,p)}{\partial q}$$

can also be derived from the principle of least action. One convenient route is:

1. *Fix the endpoints. Given $q(0) = q_0$ and $q(T) = q_T$, we seek a trajectory $q(t)$ connecting these two positions.*
2. *Define the action. For any sufficiently smooth path q satisfying the endpoint constraints, define the action*

$$\mathcal{A}[q] := \int_0^T \mathcal{L}(q(t), \dot{q}(t))\, dt, \qquad \mathcal{L}(q, \dot{q}) := K(\dot{q}) - V(q),$$

where $\mathcal{L}$ is the Lagrangian, given here by kinetic energy minus potential energy.

3. *Euler–Lagrange equation. A stationary point of $\mathcal{A}$ (with fixed endpoints) satisfies*

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}}(q,\dot{q}) - \frac{\partial \mathcal{L}}{\partial q}(q,\dot{q}) = 0.$$

In particular, if $\mathcal{L}(q,\dot{q}) = K(\dot{q}) - V(q)$ with $K(\dot{q}) = \frac{1}{2}m|\dot{q}|^2$, then

$$m\ddot{q} + \nabla V(q) = 0,$$

which is Newton's second law.

4. *Legendre transform and Hamilton's equations. Define the conjugate momentum*

$$p := \frac{\partial \mathcal{L}}{\partial \dot{q}}(q,\dot{q}).$$

Assume this relation can be inverted so that the velocity can be written as

$$\dot{q} = v(q,p).$$

For instance, if $\mathcal{L}(q,v) = K(v) - V(q)$ with $K(v) = \frac{1}{2}m|v|^2$, then

$$p = \frac{\partial \mathcal{L}(q,v)}{\partial v} = mv, \qquad \text{so} \qquad v(q,p) = \frac{p}{m}.$$

Define the Hamiltonian (Legendre transform of $\mathcal{L}$) by

$$\mathcal{H}(q,p) := p \cdot v(q,p) - \mathcal{L}\big(q, v(q,p)\big).$$

Then the Euler–Lagrange equation is equivalent to Hamilton's equations

$$\dot{q} = \frac{\partial \mathcal{H}}{\partial p}(q,p), \qquad \dot{p} = -\frac{\partial \mathcal{H}}{\partial q}(q,p).$$

Piecewise-linear approximation. We now introduce the first finite element method in this lecture. To illustrate the key idea and keep things simple, we begin with perhaps the simplest finite element setting: a one-dimensional problem with piecewise linear functions.

Recall that, by the variational formulation, (9) can be rewritten as the weak problem (11): find u satisfying the boundary conditions $u(0) = g_0$ and $u(L) = g_L$ such that

$$\int_0^L a(x)\, u'(x)\, v'(x)\, dx = \int_0^L f(x)\, v(x)\, dx \qquad \forall\, v \text{ with } v(0) = v(L) = 0.$$

The finite element method begins by partitioning the interval $[0, L]$ into N subintervals, which we call elements:

$$[0,L] = \bigcup_{i=1}^{N} E_i, \qquad E_i = [x_{i-1}, x_i], \qquad 0 = x_0 < x_1 < \cdots < x_N = L.$$

The collection of all elements is called a mesh and is denoted by

$$\mathcal{T}_h := \{E_i\}_{i=1}^N.$$

Here h is the mesh size, defined as the maximum element length,

$$h := \max_{1\le i\le N} |x_i - x_{i-1}|.$$

On each element E_i, we approximate functions by linear polynomials. In general, a piecewise polynomial function may have discontinuous jumps across element interfaces. If such jumps are present, then the derivative on $[0, L]$ is no longer well-defined in the classical sense. Therefore, we enforce continuity of the piecewise polynomial approximation across the element interfaces, which leads to the following continuous piecewise-linear finite element space:

$$V_h := \Big\{ w \in C([0,L]) \ : \ w|_{E_i} \in \mathcal{P}_1(E_i) \text{ for } i = 1, \dots, N \Big\}.$$

We also define the affine subspace that enforces the boundary conditions:

$$V_{h,g} := \{\, w \in V_h \ : \ w(0) = g_0, \quad w(L) = g_L \,\}.$$

Similarly, we define the subspace with homogeneous boundary conditions:

$$V_{h,0} := \{\, w \in V_h \ : \ w(0) = 0, \quad w(L) = 0 \,\}.$$

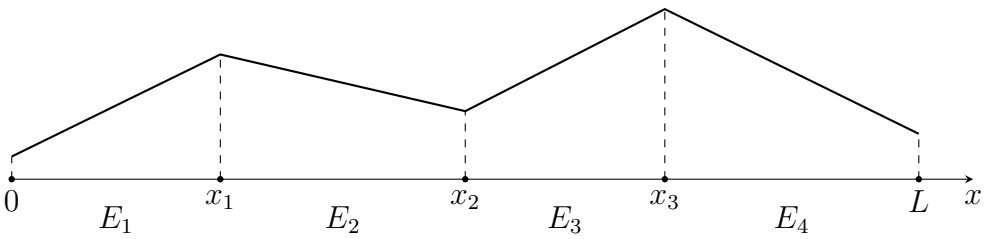


Figure 2: A 1D mesh $\mathcal{T}_h$ and a sample function $w \in V_h$ that is linear on each element and continuous at mesh points.

The finite element method is then: find $u_h \in V_{h,g}$ such that

$$\int_0^L a(x)\, u_h'(x)\, v_h'(x)\, dx = \int_0^L f(x)\, v_h(x)\, dx \qquad \forall\, v_h \in V_{h,0}. \tag{12}$$

Basis function and the linear algebraic system. The remaining task is to transform (12) into a linear algebraic system so that we can solve it on a computer. To achieve this goal, we first choose a set of basis functions for the finite element space V_h. Let the mesh points be $0 = x_0 < x_1 < \cdots < x_N = L$ and define the local mesh lengths

$$h_i := x_i - x_{i-1}, \qquad i = 1, \dots, N.$$

We define the standard *nodal (hat) basis* $\{\varphi_i\}_{i=0}^{N} \subset V_h$ by the interpolation property

$$\varphi_i(x_j) = \delta_{ij} \qquad (0 \le i, j \le N), \tag{13}$$

i.e., φ_i equals 1 at node x_i, equals 0 at all other nodes, and is linear on each element.

More explicitly, for an interior node $i = 1, \dots, N-1$,

$$\varphi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{h_i}, & x \in [x_{i-1}, x_i], \\ \dfrac{x_{i+1} - x}{h_{i+1}}, & x \in [x_i, x_{i+1}], \\ 0, & \text{otherwise}, \end{cases}$$

and at the endpoints,

$$\varphi_0(x) = \begin{cases} \dfrac{x_1 - x}{h_1}, & x \in [x_0, x_1], \\ 0, & \text{otherwise}, \end{cases} \qquad \varphi_N(x) = \begin{cases} \dfrac{x - x_{N-1}}{h_N}, & x \in [x_{N-1}, x_N], \\ 0, & \text{otherwise}. \end{cases}$$

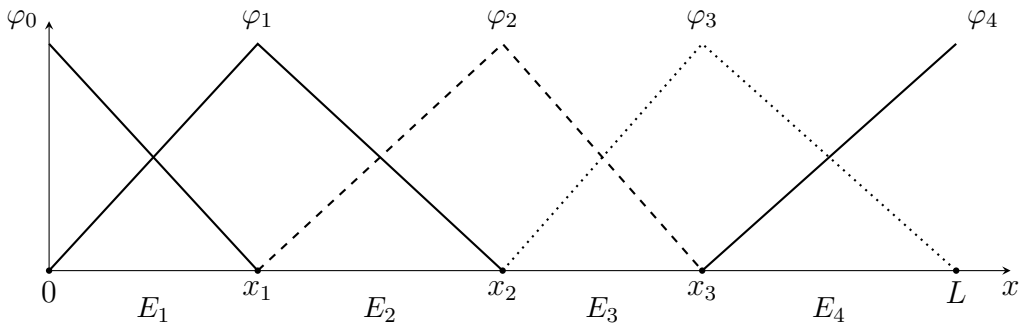


Figure 3: Nodal (hat) basis functions on the same 1D mesh. Each φ_i is piecewise linear, equals 1 at node x_i, and 0 at all other nodes.

With this basis, any $u_h \in V_h$ can be written uniquely as

$$u_h(x) = \sum_{i=0}^{N} u_i \, \varphi_i(x).$$

Moreover, by the interpolation property (13), namely, $\varphi_i(x_j) = \delta_{ij}$, we have

$$u_h(x_j) = \sum_{i=0}^{N} u_i \, \varphi_i(x_j) = \sum_{i=0}^{N} u_i \, \delta_{ij} = u_j, \qquad j = 0, 1, \dots, N.$$

We now return to the finite element formulation (12). Write the unknown as

$$u_h(x) = \sum_{i=0}^{N} u_i \, \varphi_i(x),$$

and enforce the boundary conditions by setting

$$u_0 = u_h(x_0) = u_h(0) = g_0, \qquad u_N = u_h(x_N) = u_h(L) = g_L.$$

Thus, the only unknown coefficients are u_i for $i = 1, 2, \ldots, N-1$. Next, we test (12) with each basis function $\varphi_j \in V_{h,0}$, i.e., for $j = 1, 2, \ldots, N-1$. This gives

$$\int_0^L a(x)\, u_h'(x)\, \varphi_j'(x)\, dx = \int_0^L f(x)\, \varphi_j(x)\, dx, \qquad j = 1, 2, \ldots, N-1.$$

Substituting $u_h = \sum_{i=0}^N u_i \varphi_i$ and using linearity yields

$$\sum_{i=0}^N u_i \int_0^L a(x)\, \varphi_i'(x)\, \varphi_j'(x)\, dx = \int_0^L f(x)\, \varphi_j(x)\, dx, \qquad j = 1, 2, \ldots, N-1.$$

Define the stiffness matrix $A \in \mathbb{R}^{(N-1)\times(N+1)}$ and load vector $F \in \mathbb{R}^{N-1}$ by

$$A_{ji} := \int_0^L a(x)\, \varphi_i'(x)\, \varphi_j'(x)\, dx, \qquad F_j := \int_0^L f(x)\, \varphi_j(x)\, dx, \qquad 0 \le i \le N, \quad 1 \le j \le N-1. \tag{14}$$

Then the interior unknowns $(u_1, \ldots, u_{N-1})^T$ satisfy the linear system

$$\sum_{i=1}^{N-1} A_{ji}\, u_i = F_j - u_0\, A_{j0} - u_N\, A_{jN}, \qquad j = 1, 2, \ldots, N-1,$$

Since $u_0 = g_0$ and $u_N = g_L$ are known, this is a square linear system for the unknown coefficients $u_1, \ldots, u_{N-1}$.

Assembly of the stiffness matrix and load vector. Note that the support of the basis function φ_i is

$$\operatorname{supp}(\varphi_i) = \begin{cases} [x_0, x_1], & i = 0, \\ [x_{i-1}, x_{i+1}], & i = 1, \ldots, N-1, \\ [x_{N-1}, x_N], & i = N. \end{cases}$$

In particular, φ_i and φ_j overlap only when $|i - j| \le 1$. Hence,

$$A_{ji} = 0 \quad \text{if } |i - j| > 1,$$

so the stiffness matrix is sparse; the sub-matrix for the interior unknowns $(u_1, \ldots, u_{N-1})$ is tridiagonal.

Moreover, since φ_i is linear on each element, its derivative is constant on every element: for $i = 1, \ldots, N-1$,

$$\varphi_i'(x) = \begin{cases} \dfrac{1}{h_i}, & x \in (x_{i-1}, x_i), \\ -\dfrac{1}{h_{i+1}}, & x \in (x_i, x_{i+1}), \\ 0, & \text{otherwise}, \end{cases}$$

and similarly for φ_0' and φ_N' on their single supporting elements.

Thus, each entry in (14) can be written as a sum of element contributions. Let $E_\ell := [x_{\ell-1}, x_\ell]$ for $\ell = 1, \dots, N$. Then

$$A_{ji} = \sum_{\ell=1}^{N} \int_{E_\ell} a(x)\, \varphi_i'(x)\, \varphi_j'(x)\, dx, \qquad F_j = \sum_{\ell=1}^{N} \int_{E_\ell} f(x)\, \varphi_j(x)\, dx.$$

On a fixed element $E_\ell = [x_{\ell-1}, x_\ell]$, only the two local basis functions $\varphi_{\ell-1}$ and φ_ℓ are nonzero. Therefore,

$$\int_{E_\ell} a(x)\, \varphi_i'(x)\, \varphi_j'(x)\, dx = 0 \quad \text{unless } i, j \in \{\ell-1, \ell\},$$

and likewise

$$\int_{E_\ell} f(x)\, \varphi_j(x)\, dx = 0 \quad \text{unless } j \in \{\ell-1, \ell\}.$$

Therefore, in implementation, we proceed as follows to assemble the stiffness matrix A and load vector F:

1. Initialize $A = 0$ and $F = 0$.
2. For $\ell = 1, 2, \dots, N$:
   (a) For all (i, j) with $i, j \in \{\ell-1, \ell\}$, if $1 \le j \le N-1$, update
   $$A_{ji} \mathrel{+}= \int_{E_\ell} a(x)\, \varphi_i'(x)\, \varphi_j'(x)\, dx.$$
   (b) For all $j \in \{\ell-1, \ell\}$, if $1 \le j \le N-1$, update
   $$F_j \mathrel{+}= \int_{E_\ell} f(x)\, \varphi_j(x)\, dx.$$

To evaluate these element integrals numerically, we use a quadrature rule on each element. For example, using the midpoint rule,

$$\int_{x_{\ell-1}}^{x_\ell} g(x)\, dx \approx h_\ell\, g\left(\frac{x_{\ell-1} + x_\ell}{2}\right), \qquad h_\ell := x_\ell - x_{\ell-1}.$$

Alternatively, one may use a higher-order quadrature rule for higher accuracy. When a is constant (or piecewise constant on each element), the midpoint rule is exact for the element integrals in the stiffness matrix, since $a(x)\, \varphi_i'(x)\, \varphi_j'(x)$ is constant on each element.

Example. We consider the boundary value problem (9):

$$\begin{aligned} -\partial_x\big(a(x)\, \partial_x u(x)\big) &= f(x) \qquad \text{in } (0, L), \\ u(0) = g_0, &\qquad u(L) = g_L. \end{aligned}$$

To test our finite element code, we prescribe an exact solution and a diffusion coefficient:

$$u(x) = x^{1/2}\left(1 - \frac{x}{L}\right), \qquad a(x) = 1 + \frac{1}{2}\sin(2\pi x).$$

(If $L = 1$, this reduces to $u(x) = x^{1/2}(1 - x)$.) We then define the right-hand side f by substituting the above u and a into the differential operator,

$$f(x) := -\partial_x\big(a(x)\,\partial_x u(x)\big),$$

and set the boundary data

$$g_0 := u(0), \qquad g_L := u(L).$$

With these inputs (a, f, g_0, g_L), the finite element code produces a numerical approximation u_h to u. Since the exact solution is known, we can directly evaluate and plot the error $u_h - u$.

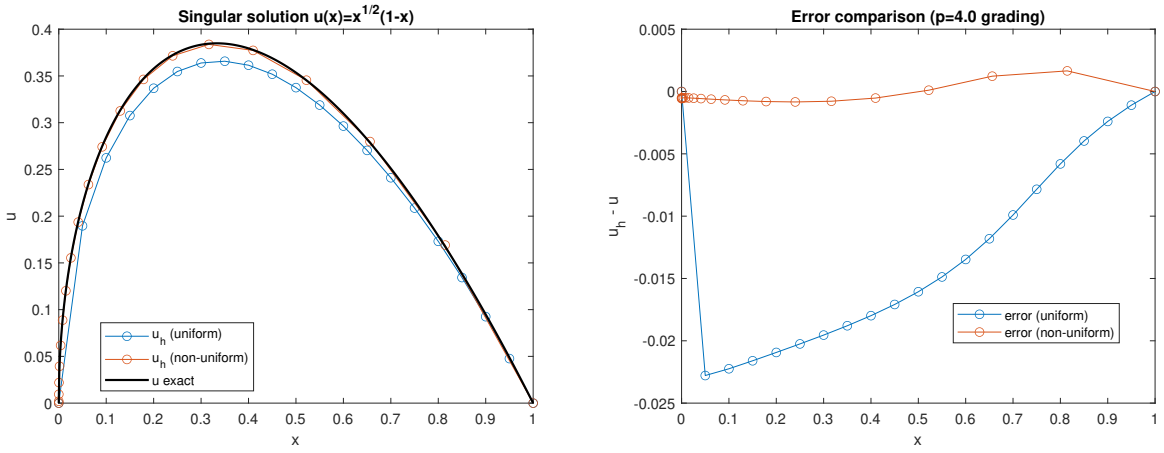
- Code

```
% Compare uniform vs non-uniform grid for a solution singular at x=0
% 1D FEM: -(a(x) u'(x))' = f(x), u(0)=g0, u(L)=gL
% P1 FEM + 2-pt Gauss quadrature

clear; close all; clc;

%% parameters
L  = 1.0;
N  = 20;

% grids
x_uni = linspace(0,L,N+1)';
p = 4.0;
x_non = ((0:N)'/N).^p * L;

%% symbolic: define a(x), exact u(x) (singular at 0), and f(x)=-(a u')'
syms xs real
a_sym = 1 + sym(1)/2*sin(2*pi*xs);

alpha = sym(2)/4;                 % <-- singular slope: u'(x) ~ x^(alpha-1) blows up
u_sym = xs^alpha*(1 - xs);        % u(0)=u(1)=0

f_sym = -diff(a_sym*diff(u_sym,xs), xs);

a       = matlabFunction(a_sym, 'Vars', xs);
f       = matlabFunction(f_sym, 'Vars', xs);
u_exact = matlabFunction(u_sym, 'Vars', xs);

g0 = u_exact(0);
gL = u_exact(L);

%% solve on both grids
[U_uni, err_uni] = fem1d(x_uni,a,f,g0,gL,u_exact);
[U_non, err_non] = fem1d(x_non,a,f,g0,gL,u_exact);

%% ---- plot solutions ----
figure;
plot(x_uni, U_uni, 'o-'); hold on;
plot(x_non, U_non, 'o-');
xx = linspace(0,L,1000);
plot(xx, u_exact(xx), 'k-','LineWidth',1.5);
```

```
legend('u_h (uniform)','u_h (non-uniform)','u exact','Location','best');
xlabel('x'); ylabel('u');
title(sprintf('Singular solution u(x)=x^{%s}(1-x)', char(alpha)));

%% ---- plot errors ----
figure;
plot(x_uni, err_uni, 'o-'); hold on;
plot(x_non, err_non, 'o-');
legend('error (uniform)','error (non-uniform)','Location','best');
xlabel('x'); ylabel('u_h - u');
title(sprintf('Error comparison (p=%.1f grading)',p));

fprintf('max nodal error (uniform)    = %.3e\n', max(abs(err_uni)));
fprintf('max nodal error (nonuniform) = %.3e\n', max(abs(err_non)));

%% ============================================================
function [U, err] = fem1d(x,a,f,g0,gL,u_exact)

N = numel(x)-1;

gp = [-1/sqrt(3),  1/sqrt(3)];
gw = [1,            1];

K = sparse(N+1,N+1);
F = zeros(N+1,1);

for ell = 1:N
    xL = x(ell); xR = x(ell+1); h = xR - xL;
    nodes = [ell, ell+1];

    dphi = [-1/h; 1/h];
    Ke = zeros(2,2);
    Fe = zeros(2,1);

    for q = 1:2
        xi = gp(q);  w = gw(q);
        xq = 0.5*(xL+xR) + 0.5*h*xi;
        J  = 0.5*h;

        phi = [(1-xi)/2; (1+xi)/2];

        Ke = Ke + w * ( a(xq) * (dphi*dphi.') ) * J;
        Fe = Fe + w * ( f(xq) * phi ) * J;
    end

    K(nodes,nodes) = K(nodes,nodes) + Ke;
    F(nodes)       = F(nodes) + Fe;
end

U = zeros(N+1,1);
U(1) = g0; U(end) = gL;

I = 2:N;  b = [1, N+1];
U(I) = K(I,I) \ ( F(I) - K(I,b)*U(b) );
```

```
ue  = u_exact(x);
err = U - ue;

end
```

- Results



Error analysis of the finite element method. In this part, we analyze the error of the finite element method (12), namely: find $u_h \in V_{g,h}$ such that

$$\int_0^L a(x)\, u_h'(x)\, v_h'(x)\, dx = \int_0^L f(x)\, v_h(x)\, dx \qquad \forall\, v_h \in V_{0,h}. \tag{15}$$

Recall that V_h denotes the space of all continuous piecewise linear functions on the given mesh. We have also introduced

$$V_{0,h} := \{v_h \in V_h \ : \ v_h(0) = v_h(L) = 0\},$$

and

$$V_{g,h} := \{v_h \in V_h \ : \ v_h(0) = g_0,\ v_h(L) = g_L\},$$

where $V_{0,h}$ is a subspace of V_h, while $V_{g,h}$ is an affine space.

Let us compare this with the weak formulation of the original problem: find $u \in V_g$ such that

$$\int_0^L a(x)\, u'(x)\, v'(x)\, dx = \int_0^L f(x)\, v(x)\, dx \qquad \forall\, v \in V_0, \tag{16}$$

where

$$V_0 := \{v \ : \ v(0) = v(L) = 0\}, \qquad V_g := \{v \ : \ v(0) = g_0,\ v(L) = g_L\}.$$

The functions in V_0 and V_g require sufficient regularity, which is formally described using Sobolev spaces. Let us not worry about the precise definitions here; intuitively, the requirement is simply that the functions and their first derivatives are square-integrable (L^2-integrable).

Since $V_{0,h} \subset V_0$, we may take $v = v_h$ in (16) for any $v_h \in V_{0,h}$. Therefore,

$$\int_0^L a(x)\, u'(x)\, v_h'(x)\, dx = \int_0^L f(x)\, v_h(x)\, dx \qquad \forall\, v_h \in V_{0,h}.$$

Comparing this identity with the discrete formulation (15), we obtain

$$\int_0^L a(x)\, (u - u_h)'(x)\, v_h'(x)\, dx = 0 \qquad \forall\, v_h \in V_{0,h}.$$

Let $e := u - u_h$ denote the error. Then the above identity can be written as

$$\int_0^L a(x)\, e'(x)\, v_h'(x)\, dx = 0 \qquad \forall\, v_h \in V_{0,h}.$$

This property is called *Galerkin orthogonality*. It states that the error is orthogonal to the finite element space $V_{0,h}$ with respect to the bilinear form

$$a(w, v) := \int_0^L a(x)\, w'(x)\, v'(x)\, dx.$$

Galerkin orthogonality is the key starting point for the error analysis. It implies that the finite element solution u_h is the best approximation to the exact solution u inside the space $V_{g,h}$ with respect to the norm

$$\|v\|_a^2 := \int_0^L a(x)\, |v'(x)|^2\, dx.$$

In fact, we have the following fundamental estimate:

Lemma 2.1 (Best approximation property)**.** *For any $v_h \in V_{g,h}$,*

$$\|u - u_h\|_a \le \|u - v_h\|_a.$$

Proof. Let $e := u - u_h$. Since $u_h, v_h \in V_{g,h}$, their difference $v_h - u_h$ belongs to $V_{0,h}$. Therefore, by Galerkin orthogonality,

$$a(e,\, v_h - u_h) = 0.$$

Now write

$$u - v_h = (u - u_h) - (v_h - u_h) = e - (v_h - u_h).$$

Taking the energy inner product with e, we obtain

$$a(e,\, u - v_h) = a(e, e) - a(e,\, v_h - u_h) = a(e, e).$$

Hence,

$$\|e\|_a^2 = a(e,\, u - v_h) \le \|e\|_a\, \|u - v_h\|_a,$$

where we used the Cauchy–Schwarz inequality for the bilinear form $a(\cdot,\cdot)$. Dividing by $\|e\|_a$ (provided $e \neq 0$) gives

$$\|u - u_h\|_a \leq \|u - v_h\|_a.$$

□

This result shows that the finite element solution u_h is the *best approximation* to u in the finite element space $V_{g,h}$ with respect to the energy norm.

Therefore, the error of the finite element method is controlled entirely by how well the exact solution u can be approximated by piecewise linear functions. In other words,

$$\|u - u_h\|_a \leq \inf_{v_h \in V_{g,h}} \|u - v_h\|_a.$$

However, the norm $\|\cdot\|_a$ is problem-dependent since it involves the coefficient $a(x)$. We would like to measure the error in a more problem-independent norm. To this end, we introduce the norm on V:

$$\|u\|_V^2 := \int_0^L \left(|u(x)|^2 + |u'(x)|^2\right) dx.$$

This is called H^1 norm or energy norm, and is a widely used norm for measuring the error of finite element method.

To relate $\|\cdot\|_a$ to $\|\cdot\|_V$, we assume the coefficient satisfies the uniform bounds

$$0 < a_0 \leq a(x) \leq a_1 \qquad \text{for all } x \in [0, L].$$

Then for any $u \in V$,

$$\|u\|_a^2 = \int_0^L a(x)\,|u'(x)|^2\,dx \leq a_1 \int_0^L |u'(x)|^2\,dx \leq a_1 \|u\|_V^2.$$

That is, the bilinear form is *bounded* with respect to $\|\cdot\|_V$.

On the other hand, a lower bound of the form

$$\|u\|_a \geq C\,\|u\|_V \qquad \forall\, u \in V$$

cannot hold in general, because $a(u,u)$ only involves u' and vanishes for constant functions. Instead, the required lower bound holds on the subspace V_0 of functions satisfying 0 boundary conditions. Indeed, for any $u \in V_0$ we use the Poincaré inequality

$$\int_0^L |u(x)|^2\,dx \leq C_P^2 \int_0^L |u'(x)|^2\,dx,$$

and therefore

$$\|u\|_V^2 = \int_0^L |u|^2\,dx + \int_0^L |u'|^2\,dx \leq (C_P^2 + 1) \int_0^L |u'|^2\,dx.$$

Combining this with the uniform positivity $a(x) \geq a_0$ yields, for all $u \in V_0$,

$$\|u\|_a^2 = \int_0^L a(x)\,|u'|^2\,dx \geq a_0 \int_0^L |u'|^2\,dx \geq \frac{a_0}{C_P^2 + 1}\,\|u\|_V^2.$$

In particular, the error $e := u - u_h$ satisfies $e(0) = e(L) = 0$, hence $e \in V_0$, and the above bound applies to e.

With these continuity and coercivity bounds, we obtain Céa's lemma.

Lemma 2.2 (Céa's lemma)**.** *Assume $0 < a_0 \le a(x) \le a_1$ on $[0, L]$. Let u and u_h solve (16) and (15). Then*

$$\|u - u_h\|_V \le \sqrt{\frac{a_1}{a_0}(1 + C_P^2)} \inf_{v_h \in V_{g,h}} \|u - v_h\|_V,$$

where C_P is the Poincaré constant on $(0, L)$.

Proof. Let $e = u - u_h$. Since $e(0) = e(L) = 0$, the Poincaré inequality gives

$$\|e\|_V^2 \le (1 + C_P^2)\|e'\|_{L^2}^2.$$

Using $a(x) \ge a_0$,

$$\|e'\|_{L^2}^2 \le \frac{1}{a_0}\|e\|_a^2, \quad \Rightarrow \quad \|e\|_V \le \sqrt{\frac{1 + C_P^2}{a_0}}\,\|e\|_a.$$

By Galerkin orthogonality,

$$\|e\|_a \le \inf_{v_h \in V_{g,h}} \|u - v_h\|_a,$$

and since $a(x) \le a_1$,

$$\|u - v_h\|_a \le \sqrt{a_1}\|u - v_h\|_V.$$

Combining the inequalities yields the result. □

Remark 2.2 (Best approximation and interpolation)**.** *Céa's lemma shows that the finite element error is controlled by the best approximation error in $V_{g,h}$:*

$$\inf_{v_h \in V_{g,h}} \|u - v_h\|_V.$$

To control the above term, we introduce a suitable interpolation operator I_h such that $I_h u \in V_{g,h}$. If the exact solution u is sufficiently smooth, one can prove

$$\|u - I_h u\|_V \le Ch,$$

where C depends on the regularity of u but is independent of the mesh size h. Consequently,

$$\inf_{v_h \in V_{g,h}} \|u - v_h\|_V \le \|u - I_h u\|_V \le Ch.$$

Combining the above estimate with Céa's lemma yields the first-order convergence of the finite element method in the $\|\cdot\|_V$ norm.

In one of the exercises, you will construct such an interpolation operator and establish this estimate.

Remark 2.3 (Stability and consistency)**.** *Recall the convergence proofs for the Euler method and the finite difference method. In both cases, convergence required two ingredients.*

First, the scheme must be locally accurate: the local truncation error must be small. This is the consistency *of the scheme.*

Second, the scheme must satisfy a global stability property. For the Euler method, this means controlling the recursive error relation. For the finite difference method, this required a discrete Poincaré-type inequality to control the discrete solution by its difference sequence.

The same philosophy applies to the finite element method.

Céa's lemma provides the stability *mechanism: the finite element error is bounded by the best approximation error, with a constant independent of h but depend on the stability of the scheme, especially the constant C_{cp}.*

The interpolation estimate

$$\|u - I_h u\|_V \leq Ch$$

provides the consistency *mechanism: it quantifies how well the discrete space approximates the exact solution. In this sense, the interpolation error plays a role analogous to the local truncation error in finite difference methods.*

2.2 Finite element method in 2D

In the previous section, we investigated the finite element method in 1D. Now, we shall use the same idea to devise a finite element method in 2D. Consider a rectangular domain:

$$\Omega = [0, L_x] \times [0, L_y].$$

The transition from 1D to 2D introduces additional geometric complexity. However, the other steps remain unchanged.

The model problem and weak formulation. Let us consider again the second-order elliptic equation:

$$\begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega, \end{cases} \tag{17}$$

where $u(\mathbf{x})$ is the solution we aim to obtain, $a = a(\mathbf{x})$ is a given coefficient function (representing a physical property such as thermal conductivity or permeability), $f(\mathbf{x})$ is the source term, and $g(\mathbf{x})$ represents the boundary data. To ensure that the problem has a unique solution, we assume that $a(\mathbf{x})$ is strictly positive and bounded, i.e., $0 < a_{\min} \leq a(\mathbf{x}) \leq a_{\max}$ for all $\mathbf{x} \in \Omega$.

The procedure for deriving the weak formulation follows the same logic as in the one-dimensional case. The corresponding action (or energy) functional is

$$\mathcal{A}(u) = \frac{1}{2} \int_\Omega a(\mathbf{x}) |\nabla u|^2 \, d\mathbf{x} - \int_\Omega f u \, d\mathbf{x}.$$

Taking the first variation of $\mathcal{A}(u)$ and setting it to zero gives the weak formulation.

Alternatively, the weak form can be obtained more directly. We multiply the strong form (namely equation (17)) by a test function v where $v = 0$ on $\partial\Omega$, and integrate over the domain Ω. Applying integration by parts, we obtain

$$\int_\Omega a\nabla u \cdot \nabla v \, d\mathbf{x} = \int_\Omega f v \, d\mathbf{x}. \tag{18}$$

For notational convenience, we define the bilinear form:

$$a(u, v) = \int_\Omega a\nabla u \cdot \nabla v d\mathbf{x}.$$

Then, the weak formulation can be written as follows: find u satisfying $u = g$ on $\partial\Omega$, such that

$$a(u, v) = \int_\Omega f v d\mathbf{x} \qquad \text{for all } v \text{ satisfying } v = 0 \text{ on } \partial\Omega.$$

Triangulation of the domain. In one dimension, a mesh is simply a collection of intervals. In two dimensions, the domain Ω is partitioned into a set of triangles. We denote this collection by

$$\mathcal{T}_h = \{K\},$$

where each K is a triangle and h denotes the mesh size, defined as the diameter of the largest triangle in the mesh. Such a partition is called a *triangulation* of the domain Ω.

A triangulation typically satisfies the following properties:

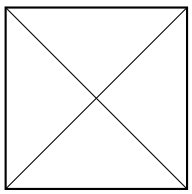
1. The triangles cover the entire domain:

$$\bigcup_{K \in \mathcal{T}_h} \overline{K} = \overline{\Omega}.$$

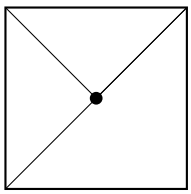
   Here the overline denotes the closure of a set.

2. We require the mesh to be conforming. Namely, for any $K_1, K_2 \in \mathcal{T}_h$ with $K_1 \neq K_2$, their intersection is either empty, a shared vertex, or a shared edge:

$$\overline{K_1} \cap \overline{K_2} = \emptyset, \quad \text{a vertex}, \quad \text{or a common edge.}$$



Conforming



Non-conforming

Finite element space and basis function. With the mesh or triangulation set up, we now define the corresponding (linear) *finite element space*. Let $\mathcal{T}_h$ be a triangulation of the domain Ω. We define

$$V_h = \{v_h : v_h \text{ is continuous on } \bar{\Omega},\ v_h|_K \in \mathbb{P}_1(K) \text{ for every } K \in \mathcal{T}_h\}. \tag{19}$$

Here $\mathbb{P}_1(K)$ denotes the space of linear polynomials on the triangle K,

$$\mathbb{P}_1(K) = \{\, p(x,y) = a + bx + cy \,\}.$$

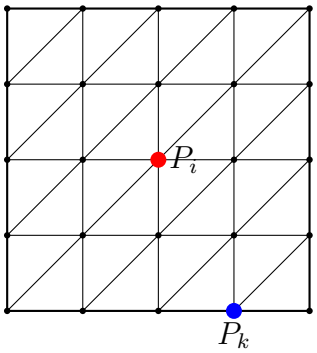
Thus functions in V_h are *piecewise linear*: they are linear on each triangle, but may have different coefficients on different triangles. The continuity requirement ensures that the function does not jump across element edges.

Just as in the one-dimensional case, a function $v_h \in V_h$ is uniquely determined by its values at the vertices (nodes) of the triangulation. To make this more concrete, we introduce the *tent basis* (also called nodal basis) for V_h.

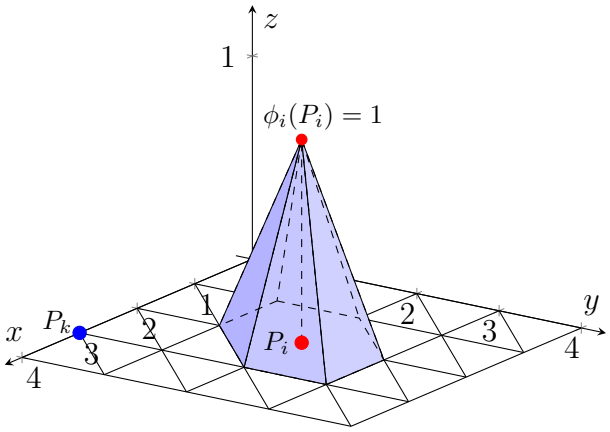
For each interior node P_i, we define a basis function $\phi_i(\mathbf{x}) \in V_h$ satisfying the property

$$\phi_i(P_j) = \delta_{ij}.$$

On any triangle K that has P_i as one of its vertices, ϕ_i is the unique linear function that takes the value 1 at P_i and 0 at the other two vertices of K. On all other triangles that do not contain P_i, the function ϕ_i is identically zero.



(a) Triangulation of Ω.



(b) Tent basis function ϕ_i.

Now, for any function $u_h \in V_h$, we can express it as a linear combination of the nodal (tent) basis functions:

$$u_h(\mathbf{x}) = \sum_{i=1}^{N} u_i \phi_i(\mathbf{x}),$$

where N is the total number of nodes in the triangulation and ϕ_i is the tent function associated with the node P_i. Evaluating this expression at the node P_i gives

$$u_h(P_i) = \sum_{j=1}^{N} u_j \phi_j(P_i) = \sum_{j=1}^{N} u_j \delta_{ij} = u_i.$$

Therefore, the coefficient u_i is exactly the value of the function at the node P_i, that is,

$$u_i = u_h(P_i).$$

Remark 2.4. *Each basis function ϕ_i is supported only on the triangles that share the node P_i. This* local support *of the basis functions is one of the key reasons why the stiffness matrix in the finite element method is sparse.*

2.2.1 Assembly and the sparse linear system

We now derive the algebraic system corresponding to the finite element approximation. To incorporate the boundary condition, we introduce the spaces

$$V_{h,0} = \{v_h \in V_h : \ v_h(P_k) = 0 \text{ for every boundary node } P_k\},$$
$$V_{h,g} = \{v_h \in V_h : \ v_h(P_k) = g(P_k) \text{ for every boundary node } P_k\}.$$

Here a boundary node means a node of the triangulation lying on $\partial\Omega$.

The finite element method is then: **find** $u_h \in V_{h,g}$ **such that**

$$a(u_h, v_h) = \int_\Omega f v_h \, d\mathbf{x} \qquad \text{for all } v_h \in V_{h,0}. \tag{20}$$

Recall that we have shown

$$u_j = u_h(P_j).$$

Therefore, for every boundary node P_k, the coefficient u_k is already prescribed by the boundary data:

$$u_k = g(P_k).$$

Only the coefficients associated with interior nodes remain unknown.

To determine these unknown coefficients, we test the equation against the basis functions associated with interior nodes. If P_i is an interior node, then $\phi_i \in V_{h,0}$, so taking $v_h = \phi_i$ gives

$$a(u_h, \phi_i) = \int_\Omega f \phi_i \, d\mathbf{x}.$$

Substituting the expansion of u_h, we obtain

$$\sum_{j=1}^{N} u_j \, a(\phi_j, \phi_i) = \int_\Omega f \phi_i \, d\mathbf{x}, \qquad P_i \text{ an interior node.}$$

Let us define the matrix entries

$$A_{ij} := a(\phi_j, \phi_i) = \int_\Omega a \, \nabla\phi_j \cdot \nabla\phi_i \, d\mathbf{x},$$

and the load vector entries

$$F_i = \int_\Omega f \phi_i \, d\mathbf{x}.$$

Hence we obtain

$$\sum_{j=1}^{N} A_{ij}u_j = F_i, \qquad P_i \text{ an interior node.}$$

Recall that the coefficients corresponding to boundary nodes are already known, since

$$u_k = g(P_k) \qquad \text{for every boundary node } P_k.$$

Therefore we separate the sum into contributions from interior nodes and boundary nodes:

$$\sum_{j\in I} A_{ij}u_j + \sum_{k\in B} A_{ik}g(P_k) = F_i,$$

where I denotes the index set of interior nodes and B the index set of boundary nodes.

Moving the known boundary contribution to the right-hand side gives

$$\sum_{j\in I} A_{ij}u_j = F_i - \sum_{k\in B} A_{ik}g(P_k), \qquad \forall i \in I.$$

Thus we obtain a linear system for the unknown coefficients associated with interior nodes:

$$A_{I,I}U_I = F_I - A_{I,B}g_B,$$

where

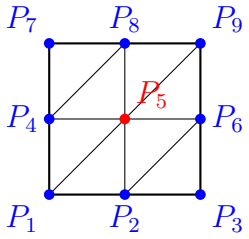
- $A_{I,I}$ is the submatrix of A consisting of rows and columns corresponding to interior nodes,
- $A_{I,B}$ consists of rows corresponding to interior nodes and columns corresponding to boundary nodes,
- $g_B = (g(P_k))_{k\in B}$ denotes the vector of prescribed boundary values.

Once the coefficients u_j for interior nodes are determined, the finite element solution is reconstructed as

$$u_h(\mathbf{x}) = \sum_{j=1}^{N} u_j\phi_j(\mathbf{x}),$$

where the coefficients corresponding to boundary nodes satisfy $u_k = g(P_k)$.

Example. To illustrate the key ideas, let us consider a simple example in which the domain is partitioned into a uniform mesh consisting of 8 triangles.



$B = \{1, 2, 3, 4, 6, 7, 8, 9\}$

$I = \{5\}$

Research project: finite element method on square and L-shaped domains

In this project you will implement a two-dimensional finite element method (FEM) for the Laplace equation

$$-\Delta u = 0 \qquad \text{in } \Omega, \qquad u = g \qquad \text{on } \partial\Omega.$$

Such equations arise, for example, in steady-state heat conduction, where $u(\mathbf{x})$ represents the temperature distribution inside a material whose boundary temperature is prescribed by the boundary data g.

You will solve this problem on two different domains:

$$\Omega_1 = (-1,1)^2 \qquad \text{(square domain)}$$

$$\Omega_2 = (-1,1)^2 \setminus [0,1] \times [-1,0] \qquad \text{(L-shaped domain)}$$

Project tasks:

1. **Finite element implementation.**
   Implement a piecewise-linear finite element method for the Laplace equation. Your program should

   - generate a triangulation of the domain,
   - assemble the stiffness matrix
   $$A_{ij} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, d\mathbf{x},$$
   - impose Dirichlet boundary conditions,
   - solve the resulting linear system,
   - visualize the mesh and the numerical solution.

   You may use MATLAB, Python, or another programming language. You may also use AI tools to assist you, but you must understand your implementation and be able to explain the main steps.

2. **Convergence study.**
   Choose an exact *harmonic* solution u and impose the corresponding boundary data
   $$g = u|_{\partial\Omega}.$$
   Compute numerical solutions on a sequence of refined meshes and measure the error between the numerical solution u_h and the exact solution.
   Possible error measures include

   - nodal maximum error: $\max_{i=1}^{N} |u(P_i) - u_h(P_i)|$
   - L^2 error: $\left(\int_\Omega (u(x) - u_h(x))^2 dx\right)^{1/2}$

   Plot the error as a function of the mesh size h and estimate the convergence rate.

Hints. On the square domain, you may test a smooth harmonic function, for example

$$u(x, y) = \sin(\pi x) \sinh(\pi y).$$

On the L-shaped domain, a commonly used singular harmonic solution is

$$u(r, \theta) = r^{2/3} \sin\left(\frac{2\theta}{3}\right),$$

which reflects the reduced regularity near the re-entrant corner of the domain.

3. **Domain comparison.**
   Repeat the convergence experiment on both domains:

   - the square domain,
   - the L-shaped domain.

   Compare the observed convergence rates and explain the differences.

Deliverables
You will submit the following individually:

- a written report describing your numerical method, implementation, and numerical experiments,
- a 10–12 minute presentation summarizing your findings.

You may use any tools you find helpful (including AI), but you must understand and be able to explain your work. Questions will be asked during your presentation.

Grading rubric. Each project will be graded using the same criteria:

- **Numerical method** (2 points) Understanding and formulation of the numerical method(s) used in the project.
- **Implementation** (2 points) Correctness, clarity, and organization of the computational implementation.
- **Numerical experiments** (2 points) Quality of experiments, figures/tables, and analysis of numerical results.
- **Report quality** (2 points) Clarity of writing, logical organization, and explanation of the work.
- **Presentation** (2 points) Clear oral presentation and ability to answer questions.