Introduction and Basic Implementation for Finite Element Methods

Chapter 1: Finite Elements for 1D second order elliptic equation

Xiaoming He
Department of Mathematics & Statistics
Missouri University of Science & Technology

Outline

Outline

Target problem

- Solve

$$-\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x),\ a < x < b,$$
$$u(a) = g_a, u(b) = g_b$$

  for $u(x)$.

- Why do we start from this problem?
- An easy look at the basic idea of the finite element method.
- Numerical methods for partial differential equations: finite element method, finite difference method, finite volume method, boundary element method, etc., which use different techniques to discretize partial differential equations.

Weak formulation

- First, multiply a function $v(x)$ on both sides of the original equation,

$$
\begin{aligned}
& -\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x),\ a < x < b \\
\Rightarrow\quad & -\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) v(x) = f(x)v(x),\ a < x < b \\
\Rightarrow\quad & -\int_a^b \frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) v(x)\ dx = \int_a^b f(x)v(x)\ dx.
\end{aligned}
$$

- $u(x)$ is called a trial function and $v(x)$ is called a test function.

Weak formulation

- Second, using integration by parts, we obtain

$$\begin{aligned}
& \int_a^b \frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) v(x)\ dx \\
= & \int_a^b \left(cu'\right)' v\ dx \\
= & \int_a^b v\ d(cu') \\
= & \ cu'v|_a^b - \int_a^b cu'\ dv \\
= & \ c(b)u'(b)v(b) - c(a)u'(a)v(a) - \int_a^b cu'v'\ dx.
\end{aligned}$$

Weak formulation

- Then

$$-c(b)u'(b)v(b) + c(a)u'(a)v(a) + \int_a^b cu'v' \, dx = \int_a^b fv \, dx.$$

- Since the solution at $x = a$ and $x = b$ are given by $u(a) = g_a, u(b) = g_b$, then we can choose the test function $v(x)$ such that $v(a) = v(b) = 0$.
- Hence

$$\int_a^b cu'v' \, dx = \int_a^b fv \, dx.$$

- What spaces should u and v belong to? Sobolev spaces!

1D Sobolev spaces

Definition (Support)

If u is a function, then its support $supp(u)$ is the closure of the set on which u is nonzero.

Definition (Compactly supported)

If u is a function defined on an open interval I and $supp(u)$ is a compact subset (that is, a closed and bounded subset), then u is said to be compactly supported in I.

Lemma (I)

A function compactly supported in an open interval I is zero on and near the boundary of I.

1D Sobolev spaces

Definition

$C_0^\infty(I)$ is the set of all functions that are infinitely differentiable on I and compactly supported in I.

- Recall integration by parts:

$$\begin{aligned}\int_a^b u'v\ dx &= uv|_a^b - \int_a^b uv'\ dx \\ &= u(b)v(b) - u(a)v(a) - \int_a^b uv'\ dx.\end{aligned}$$

- For $v \in C_0^\infty(I)$, we have $v(a) = v(b) = 0$. Then

$$\int_a^b u'v\ dx = -\int_a^b uv'\ dx.$$

1D Sobolev spaces

Definition (weak derivative)

Suppose u is a real-valued function defined on an open interval $I = (a, b)$ and that u is integrable over every compact subset of I. If there exists another locally integrable function w defined on I such that

$$\int_a^b wv \, dx = -\int_a^b uv' \, dx$$

for all $v \in C_0^\infty(I)$, then u is said to be weakly differentiable and w is called the weak derivative of u.

1D Sobolev spaces

Lemma (II)

If u is differentiable, then u is weakly differentiable and its weak derivative is u'.

Remark

In the Sobolev spaces, which will be defined below, u' is used to represent the weak derivative.

1D Sobolev spaces

Definition (L^2 space)

$$L^2(I) = \{v : I \to \mathbf{R} : \int_a^b v^2 \, dx < \infty\}$$

where $I = (a, b)$.

Definition (H^1 space)

$$H^1(I) = \{v \in L^2(I) : v' \in L^2(I)\}$$

where $I = (a, b)$.

Definition (H_0^1 space)

$$H_0^1(I) = \{v \in H^1(I) : v(a) = v(b) = 0\}$$

where $I = (a, b)$.

Weak formulation

- Weak formulation: find $u \in H^1(I)$ such that

$$\int_a^b cu'v' \, dx = \int_a^b fv \, dx.$$

for any $v \in H_0^1(I)$ where $I = (a, b)$.

- Let $a(u, v) = \int_a^b cu'v' \, dx$ and $(f, v) = \int_a^b fv \, dx$.

- Weak formulation: find $u \in H^1(I)$ such that

$$a(u, v) = (f, v)$$

for any $v \in H_0^1(I)$ where $I = (a, b)$.

Galerkin formulation

- Assume there is a finite dimensional subspace $U_h \subset H^1(a,b)$. Define U_{h0} to be the space which consists of the functions of U_h with value 0 on the Dirichlet boundary.
- Galerkin formulation: find $u_h \in U_h$ such that

$$a(u_h, v_h) = (f, v_h)$$
$$\Leftrightarrow \quad \int_a^b c u_h' v_h' \, dx = \int_a^b f v_h \, dx$$

  for any $v_h \in U_{h0}$.
- Basic idea of Galerkin formulation: use finite dimensional space to approximate infinite dimensional space.
- Question: How to obtain U_h?

Outline

Mesh

- Assume that we have a uniform partition of $[a, b]$ into N elements with mesh size $h = \frac{b-a}{N}$.
- Let $x_i = a + (i-1)h$ $(i = 1, \cdots, N+1)$ denote the mesh nodes.
- Let $E_n = [x_n, x_{n+1}]$ $(n = 1, \cdots, N)$ denote the mesh elements.

1D linear finite element space

- Define 1D linear finite element space:

$$U_h = \{\phi \in C[a,b] : \ \phi(x) \text{ is linear on each } [x_n, x_{n+1}] \ (n = 1, 2, \cdots, N)\}.$$

- U_h is actually a piecewise linear function space based on the mesh in the previous page.

1D linear finite element space

Theorem (I)

U_h is an $(N+1)-$dimensional subspace of $C[a,b]$.

Proof:

- First, it is easy to verify that U_h is a subspace of $C[a,b]$.
- If we can find a continuous piecewise linear basis of $N+1$ functions for U_h, then the proof is completed.
- Consider $\phi_j(x) \in U_h$ such that

$$\phi_j(x_i) = \delta_{ij} = \begin{cases} 0, & \text{if } j \neq i, \\ 1, & \text{if } j = i. \end{cases}$$

for $i, j = 1, \cdots, N+1$.

1D linear finite element space

Continued proof:

- In fact,

$$\phi_1(x) = \begin{cases} \frac{x_2-x}{h}, & if\ x_1 \le x \le x_2, \\ 0, & otherwise, \end{cases}$$

$$\phi_j(x) = \begin{cases} \frac{x-x_{j-1}}{h}, & if\ x_{j-1} \le x \le x_j, \\ \frac{x_{j+1}-x}{h}, & if\ x_j \le x \le x_{j+1}, \\ 0, & otherwise, \end{cases}$$

$$(j = 2, \cdots, N)$$

$$\phi_{N+1}(x) = \begin{cases} \frac{x-x_N}{h}, & if\ x_N \le x \le x_{N+1}, \\ 0, & otherwise, \end{cases}$$

(Plot these basis functions by yourself)

- In order to show that $\phi_j(x)$ $(i = 1, \cdots, N+1)$ form a basis of U_h, we need to show the linear independence of $\{\phi_j\}_{j=1}^{N+1}$ and $U_h = span\{\phi_j\}_{j=1}^{N+1}$.

1D linear finite element space

Continued proof:

- Linear independence: consider

$$\sum_{j=1}^{N+1} c_j \phi_j(x) = 0$$

  for any $x \in [a, b]$.

- Let $x = x_i$ $(i = 1, \cdots, N+1)$, then

$$\phi_j(x_i) = \delta_{ij} = \begin{cases} 0, & \text{if } j \neq i, \\ 1, & \text{if } j = i. \end{cases}$$

$$\Rightarrow \quad c_i = 0 \ (i = 1, \cdots, N+1)$$

- So $\phi_j(x)$ $(j = 1, \cdots, N+1)$ are linearly independent.

1D linear finite element space

Continued proof:

- Span: Given any $f \in U_h$, let $c_j = f(x_j)$ and consider

$$g(x) = \sum_{j=1}^{N+1} c_j \phi_j(x).$$

- First, $g(x_i) = c_i = f(x_i)$ $(i = 1, \cdots, N+1)$.
- Second, both $f(x)$ and $g(x)$ are linear in each piece $[x_i, x_{i+1}]$ $(j = 1, \cdots, N)$.
- Hence $f(x) = g(x)$ in each piece $[x_i, x_{i+1}]$ $(i = 1, \cdots, N)$.
- Then $f(x) = g(x) = \sum_{j=1}^{N+1} c_j \phi_j(x)$.
- This implies $U_h = span\{\phi_j\}_{j=1}^{N+1}$.
- Therefore $\phi_j(x)$ $(j = 1, \cdots, N+1)$ form a basis of U_h.

Outline

Discretization formulation

- Recall the Galerkin formulation: find $u_h \in U_h$ such that

$$a(u_h, v_h) = (f, v_h)$$

$$\Leftrightarrow \quad \int_a^b c u_h' v_h' \, dx = \int_a^b f v_h \, dx$$

for any $v_h \in U_{h0}$.

- For an easier implementation, we use the following Galerkin formulation (without considering the Dirichlet boundary condition, which will be handled later): find $u_h \in U_h$ such that

$$a(u_h, v_h) = (f, v_h)$$
$$\Leftrightarrow \quad \int_a^b c u_h' v_h' \, dx = \int_a^b f v_h \, dx$$

  for any $v_h \in U_h$.

- Since $u_h \in U_h = span\{\phi_j\}_{j=1}^{N+1}$, then

$$u_h = \sum_{j=1}^{N+1} u_j \phi_j$$

  for some coefficients u_j $(j = 1, \cdots, N+1)$.

Discretization formulation

- If we can set up a linear algebraic system for u_j $(j = 1, \cdots, N+1)$ and solve it, then we can obtain the finite element solution u_h.
- Therefore, we choose the test function $v_h = \phi_i$ $(i = 1, \cdots, N+1)$. Then the finite element formulation gives

$$\int_a^b c \left(\sum_{j=1}^{N+1} u_j \phi_j \right)' \phi_i' \, dx = \int_a^b f \phi_i \, dx, \ i = 1, \cdots, N+1$$

$$\Rightarrow \quad \sum_{j=1}^{N+1} u_j \left[\int_a^b c \phi_j' \phi_i' \, dx \right] = \int_a^b f \phi_i \, dx, \ i = 1, \cdots, N+1.$$

Discretization formulation

- Define the stiffness matrix

$$A = [a_{ij}]_{i,j=1}^{N+1} = \left[\int_a^b c\phi_j' \phi_i' \, dx \right]_{i,j=1}^{N+1}.$$

- Define the load vector

$$\vec{b} = [b_i]_{i=1}^{N+1} = \left[\int_a^b f\phi_i \, dx \right]_{i=1}^{N+1}.$$

- Define the unknown vector

$$\vec{X} = [u_j]_{j=1}^{N+1}.$$

- Then we obtain the linear algebraic system

$$A\vec{X} = \vec{b}.$$

- Here A is symmetric positive-definite if the original elliptic equation is symmetric positive-definite.

Discretization formulation

Remark

- *In fact, since*

$$\phi_j(x_k) = \delta_{jk} = \begin{cases} 0, & \text{if } j \neq k, \\ 1, & \text{if } j = k. \end{cases}$$

then

$$u_h(x_k) = \sum_{j=1}^{N+1} u_j \phi_j(x_k) = u_k.$$

- *Hence the coefficient u_j is actually the numerical solution at the node x_j $(j = 1, \cdots, N+1)$.*
- *Once $\vec{X} = [u_j]_{j=1}^{N+1}$ is obtained, the finite element solution $u_h = \sum_{j=1}^{N+1} u_j \phi_j$ and the numerical solutions at all the mesh nodes are obtained.*

Assembly of the stiffness matrix

- In this section we will first introduce the matrix and vector assembly by using a special method. In the later section "FE method", we will discuss a different universal framework.
- From the definition of ϕ_j $(j = 1, \cdots, N+1)$, we can see that ϕ_j are non-zero only on the elements adjacent to the node x_j, but 0 on all the other elements.
- This observation motivates us to think about

$$a_{ij} = \int_a^b c\phi_j'\phi_i' \, dx = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx, \ i, j = 1, \cdots, N+1.$$

- It is easy to see that most of $\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx$ will be 0.
- So we only need to use numerical integration to compute those nonzero integrals.

Assembly of the stiffness matrix

- Case 1: when $|i-j|>1$, x_i and x_j are not neighboring mesh nodes.
- Then on any element $[x_n, x_{n+1}]$ $(n=1,\cdots,N)$, at least one of ϕ_j and ϕ_i is 0.
- Hence

$$\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx = 0 \ (n=1,\cdots,N)$$

$$\Rightarrow \quad a_{ij} = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} c\phi_j'\phi_i'\,dx = 0.$$

Assembly of the stiffness matrix

- Case 2: when $i = j + 1$ $(j = 1, \cdots, N)$, the only element, on which both ϕ_j and ϕ_i are not zero, is $[x_j, x_{j+1}]$.
- Hence

$$\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx = 0 \ (n = 1, \cdots, j-1, j+1, \cdots, N)$$

$$\Rightarrow \quad a_{ij} = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx = \int_{x_j}^{x_{j+1}} c\phi_j'\phi_i' \, dx$$

$$\Rightarrow \quad a_{j+1,j} = \int_{x_j}^{x_{j+1}} c(x) \left(\frac{x_{j+1} - x}{h}\right)' \left(\frac{x - x_j}{h}\right)' \, dx$$

$$= -\frac{1}{h^2} \int_{x_j}^{x_{j+1}} c(x) \, dx.$$

Assembly of the stiffness matrix

- Case 3: when $i = j - 1$ $(j = 2, \cdots, N+1)$, the only element, on which both ϕ_j and ϕ_i are not zero, is $[x_{j-1}, x_j]$.
- Hence

$$\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx = 0 \ (n = 1, \cdots, j-2, j, \cdots, N)$$

$$\Rightarrow \quad a_{ij} = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx = \int_{x_{j-1}}^{x_j} c\phi_j'\phi_i' \, dx$$

$$\Rightarrow \quad a_{j-1,j} = \int_{x_{j-1}}^{x_j} c(x) \left(\frac{x - x_{j-1}}{h}\right)' \left(\frac{x_j - x}{h}\right)' \, dx$$

$$= -\frac{1}{h^2} \int_{x_{j-1}}^{x_j} c(x) \, dx.$$

Assembly of the stiffness matrix

- Case 4: when $i = j$ $(j = 2, \cdots, N)$, the only two elements, on which both ϕ_j and ϕ_i are not zero, are $[x_{j-1}, x_j]$ and $[x_j, x_{j+1}]$.
- Hence

$$\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx = 0 \ (n = 1, \cdots, j-2, j+1, \cdots, N)$$

$$\Rightarrow \quad a_{ij} = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx = \int_{x_{j-1}}^{x_j} c\phi_j'\phi_i' \, dx + \int_{x_j}^{x_{j+1}} c\phi_j'\phi_i' \, dx$$

$$\Rightarrow \quad a_{jj} = \int_{x_{j-1}}^{x_j} c(x) \left(\frac{x - x_{j-1}}{h}\right)' \left(\frac{x - x_{j-1}}{h}\right)' dx$$

$$+ \int_{x_j}^{x_{j+1}} c(x) \left(\frac{x_{j+1} - x}{h}\right)' \left(\frac{x_{j+1} - x}{h}\right)' dx$$

$$= \frac{1}{h^2} \int_{x_{j-1}}^{x_j} c(x) \, dx + \frac{1}{h^2} \int_{x_j}^{x_{j+1}} c(x) \, dx.$$

Assembly of the stiffness matrix

- Case 5: when $i = j = 1$, the only element, on which both ϕ_j and ϕ_i are not zero, is $[x_1, x_2]$.
- Hence

$$\int_{x_n}^{x_{n+1}} c\phi_1'\phi_1' \, dx = 0 \ (n = 2, \cdots, N)$$

$$\Rightarrow \quad a_{11} = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} c\phi_1'\phi_1' \, dx = \int_{x_1}^{x_2} c\phi_1'\phi_1' \, dx$$

$$\Rightarrow \quad a_{11} = \int_{x_1}^{x_2} c(x) \left(\frac{x_2 - x}{h}\right)' \left(\frac{x_2 - x}{h}\right)' \, dx$$

$$= \frac{1}{h^2} \int_{x_1}^{x_2} c(x) \, dx.$$

Assembly of the stiffness matrix

- Case 6: when $i = j = N + 1$, the only element, on which both ϕ_j and ϕ_i are not zero, is $[x_N, x_{N+1}]$.
- Hence

$$\int_{x_n}^{x_{n+1}} c\phi'_{N+1}\phi'_{N+1}\, dx = 0 \ (n = 1, \cdots, N-1)$$

$$\Rightarrow \quad a_{N+1,N+1} = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} c\phi'_{N+1}\phi'_{N+1}\, dx = \int_{x_N}^{x_{N+1}} c\phi'_{N+1}\phi'_{N+1}\, dx$$

$$\Rightarrow \quad a_{N+1,N+1} = \int_{x_N}^{x_{N+1}} c(x) \left(\frac{x - x_N}{h}\right)' \left(\frac{x - x_N}{h}\right)' dx$$

$$= \frac{1}{h^2} \int_{x_N}^{x_{N+1}} c(x)\, dx.$$

Assembly of the stiffness matrix

- From the above discussion, we can see that most of the elements a_{ij} $(i, j = 1, \cdots, N+1)$ are 0.
- Hence the stiffness matrix A is called a sparse matrix.
- We can also see that we only need to compute the integrals on local elements instead of the whole domain, which later will lead to the "local assembly" idea of finite elements.

Assembly of the stiffness matrix

Algorithm 1:

- Initialize the matrix: $A = sparse(N+1, N+1)$;
- Compute the integrals and assemble them into A:

$FOR\ j = 1, \cdots, N+1$:

$IF\ j \leq N, THEN$

Compute $A(j+1, j) = -\frac{1}{h^2} \int_{x_j}^{x_{j+1}} c(x)\ dx$;

END

$IF\ j \geq 2, THEN$

Compute $A(j-1, j) = -\frac{1}{h^2} \int_{x_{j-1}}^{x_j} c(x)\ dx$;

END

$IF\ 2 \leq j \leq N, THEN$

Compute $A(j, j) = \frac{1}{h^2} \int_{x_{j-1}}^{x_j} c(x)\ dx + \frac{1}{h^2} \int_{x_j}^{x_{j+1}} c(x)\ dx$;

END

END

Compute $A(1, 1) = \frac{1}{h^2} \int_{x_1}^{x_2} c(x)\ dx$;

Compute $A(N+1, N+1) = \frac{1}{h^2} \int_{x_N}^{x_{N+1}} c(x)\ dx$;

Assembly of the load vector

- The idea for the assembly of the load vector is similar. We have

$$b_i = \int_a^b f\phi_i \, dx \quad = \quad \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} f\phi_i \, dx, \;\; i = 1, \cdots, N+1,$$

- Case 1: when $2 \leq i \leq N$, the only two elements, on which ϕ_i is not zero, are $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$. Then

$$\int_{x_n}^{x_{n+1}} f\phi_i \, dx = 0 \;\; (n = 1, \cdots, i-2, i+1, \cdots, N)$$

$$\Rightarrow \quad b_i = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} f\phi_i \, dx = \int_{x_{i-1}}^{x_i} f\phi_i \, dx + \int_{x_i}^{x_{i+1}} f\phi_i \, dx$$

$$= \int_{x_{i-1}}^{x_i} f(x) \frac{x - x_{i-1}}{h} \, dx + \int_{x_i}^{x_{i+1}} f(x) \frac{x_{i+1} - x}{h} \, dx.$$

Assembly of the load vector

- Case 2: when $i = 1$, the only element, on which ϕ_1 is not zero, is $[x_1, x_2]$. Then

$$\int_{x_n}^{x_{n+1}} f\phi_1 \, dx = 0 \ (n = 2, \cdots, N)$$

$$\Rightarrow \quad b_1 = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} f\phi_1 \, dx$$

$$= \int_{x_1}^{x_2} f\phi_1 \, dx = \int_{x_1}^{x_2} f(x) \frac{x_2 - x}{h} \, dx.$$

Assembly of the load vector

- Case 3: when $i = N + 1$, the only element, on which ϕ_{N+1} is not zero, is $[x_N, x_{N+1}]$. Then

$$\int_{x_n}^{x_{n+1}} f\phi_{N+1}\, dx = 0 \ (n = 1, \cdots, N-1)$$

$$\Rightarrow \quad b_{N+1} = \sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} f\phi_{N+1}\, dx = \int_{x_N}^{x_{N+1}} f\phi_{N+1}\, dx$$

$$= \int_{x_N}^{x_{N+1}} f(x)\frac{x - x_N}{h}\, dx.$$

Assembly of the load vector

Algorithm 2:

- Initialize the matrix: $b = zeros(N+1, 1)$;
- Compute the integrals and assemble them into $\vec{b}$:
  $FOR\ i = 2, \cdots, N$:
  Compute
  $b(i) = \int_{x_{i-1}}^{x_i} f(x)\frac{x-x_{i-1}}{h}\ dx + \int_{x_i}^{x_{i+1}} f(x)\frac{x_{i+1}-x}{h}\ dx$;
  END
  Compute $b(1) = \int_{x_1}^{x_2} f(x)\frac{x_2-x}{h}\ dx$;
  Compute $b(N+1) = \int_{x_N}^{x_{N+1}} f(x)\frac{x-x_N}{h}\ dx$;

Outline

Dirichlet boundary condition

- Basically, the Dirichlet boundary condition $u(a) = g_a, u(b) = g_b$ give the solutions at $x_1 = a$ and $x_{N+1} = b$.
- Since the coefficient u_j in the finite element solution $u_h = \sum_{j=1}^{N+1} u_j \phi_j$ is actually the numerical solution at the node x_j $(j = 1, \cdots, N+1)$, we actually know that $u_1 = u(a) = g_a$ and $u_{N+1} = u(b) = g_b$.
- Therefore, we don't really need the first and last equations in the linear system since they are set up for u_1 and u_{N+1} by using ϕ_1 and ϕ_{N+1}.

Dirichlet boundary condition

- One of the popular ways to impose the Dirichlet boundary condition is to replace the first and last equations in the linear system by the following two equations

$$u_1 = g_a \Rightarrow 1 \cdot u_1 + 0 \cdot u_2 + \cdots + 0 \cdot u_{N+1} = g_a,$$
$$u_{N+1} = g_b \Rightarrow 0 \cdot u_1 + \cdots + 0 \cdot u_N + 1 \cdot u_{N+1} = g_b.$$

- That is, the first and last rows of the matrix A should become

$$(1, 0, \cdots, 0)$$

and

$$(0, \cdots, 0, 1)$$

respectively.

- And the first and last elements of the vector $\vec{b}$ should become g_a and g_b respectively.

Dirichlet boundary condition

Algorithm 3:

- Deal with the Drichlet boundary conditions:

$$A(1,:) = 0;$$
$$A(1,1) = 1;$$
$$A(N+1,:) = 0;$$
$$A(N+1,N+1) = 1;$$
$$b(1) = g_a;$$
$$b(N+1) = g_b;$$

Outline

Basic algorithm

- Input a, b, and N. Compute $h = \frac{b-a}{N}$ and $x_j = a + (j-1)h$ $(j = 1, \cdots, N+1)$.
- Assemble the stiffness matrix A by using Algorithm 1.
- Assemble the load vector $\vec{b}$ by using Algorithm 2.
- Deal with the Drichlet boundary condition by using Algorithm 3.
- Solve $A\vec{X} = \vec{b}$ for $\vec{X}$ by using a direct or iterative method.

Remark

The above algorithm uses the Algorithms 1, 2, and 3, which are designed for some particular cases with a special method. It is not general enough to deal with different types of PDEs. Therefore, we will discuss a more universal framework in the following.

Universal framework of the finite element method

- Generate the information matrices: P, T, E;
- Assemble the matrices and vectors: local assembly based on P, T, E only;
- Deal with the boundary conditions: boundary information matrix and local assembly;
- Solve linear systems: numerical linear algebra.

Mesh information matrices

- Define your global indices for all the mesh elements and mesh nodes. Let N denote the number of mesh elements and N_m denote the number of mesh nodes. Here $N_m = N + 1$.
- Define matrix P to be an information matrix consisting of the coordinates of all mesh nodes.
- Define matrix T to be an information matrix consisting of the global node indices of the mesh nodes of all the mesh elements.

Mesh information matrices

- For example, for the mesh used in this chapter, we can use the j^{th} column of the matrix P to store the coordinates of the j^{th} mesh node and the n^{th} column of the matrix T to store the global node indices of the mesh nodes of the n^{th} mesh element:

$$
\begin{aligned}
P &= \begin{pmatrix} x_1 & x_2 & \cdots & x_{N_m-1} & x_{N_m} \end{pmatrix} \\
&= \begin{pmatrix} x_1 & x_2 & \cdots & x_N & x_{N+1} \end{pmatrix}, \\
T &= \begin{pmatrix} 1 & 2 & \cdots & N_m-2 & N_m-1 \\ 2 & 3 & \cdots & N_m-1 & N_m \end{pmatrix} \\
&= \begin{pmatrix} 1 & 2 & \cdots & N-1 & N \\ 2 & 3 & \cdots & N & N+1 \end{pmatrix}.
\end{aligned}
$$

Finite element information matrices

- The above mesh information matrices P and T are for the mesh nodes.
- We also need similar finite element information matrices P_b and T_b for the finite elements nodes, which are the nodes corresponding to the finite element basis functions.
- For example, the finite element nodes of the linear finite element are the same as those mesh nodes since all the linear basis functions are corresponding to mesh nodes.
- Note: For the nodal finite element basis functions, the correspondence between the finite elements nodes and the finite element basis functions is one-to-one in a straightforward way. But it could be more complicated for other types of finite element basis functions in the future.

Finite element information matrices

- Define your global indices for all the mesh elements and finite element nodes (or the finite element basis functions). Let N_b denote the total number of the finite element basis functions (= the number of unknowns = the total number of the finite element nodes). Here $N_b = N + 1$.
- Then

$$u_h = \sum_{j=1}^{N_b} u_j \phi_j.$$

- Define matrix P_b to be an information matrix consisting of the coordinates of all finite element nodes.
- Define matrix T_b to be an information matrix consisting of the global node indices of the finite element nodes of all the mesh elements.

Finite element information matrices

- For the linear finite elements we use here, $P_b = P$ and $T_b = T$ since the nodes of the linear finite element basis functions are the same as those of the mesh. We use the j^{th} column of the matrix P_b to store the coordinates of the j^{th} finite element node and the n^{th} column of the matrix T_b to store the global node indices of the finite element nodes of the n^{th} mesh element:

$$
\begin{aligned}
P_b &= \begin{pmatrix} x_1 & x_2 & \cdots & x_{N_b-1} & x_{N_b} \end{pmatrix} \\
&= \begin{pmatrix} x_1 & x_2 & \cdots & x_N & x_{N+1} \end{pmatrix}, \\
T_b &= \begin{pmatrix} 1 & 2 & \cdots & N_b-2 & N_b-1 \\ 2 & 3 & \cdots & N_b-1 & N_b \end{pmatrix} \\
&= \begin{pmatrix} 1 & 2 & \cdots & N-1 & N \\ 2 & 3 & \cdots & N & N+1 \end{pmatrix}.
\end{aligned}
$$

Finite element information matrices

Remark

For many types of finite elements, such as the quadratic elements which will be discussed later and some elements which will be introduced in Chapter 2, P_b and T_b are different from P and T since the nodes for the finite element basis functions are different from those of the mesh.

Local assembly

Observation based on Algorithm 1:

- All the non-zero entries in the stiffness matrix A come from the non-zero local integrals defined on the mesh elements.
- In each non-zero local integral, the trial and test basis functions are only corresponding to the nodes of the element which is the integral interval.
- On each element, all the local integrals, whose trial and test basis functions are corresponding to the nodes of this element, have non-trivial contribution to some non-zero entries of the stiffness matrix A.

Local assembly

New assembly idea for the stiffness matrix A:

- Loop over all the mesh elements;
- Compute all non-zero local integrals on each element for A;
- Assemble these non-zero local integrals into the corresponding entries of the stiffness matrix A.

Local assembly

Compute all non-zero local integrals on each element for A:

- On the n^{th} element $E_n = [x_n, x_{n+1}]$, we get non-zero local integrals only when the trial and test basis functions are corresponding to the finite element nodes of the element.
- That is, we only consider the trial and test basis functions to be ϕ_n or ϕ_{n+1}.
- There are only four non-zero local integrals on E_n with the global basis functions ϕ_n and ϕ_{n+1}:

$$\int_{x_n}^{x_{n+1}} c\phi_n'\phi_n' dx, \int_{x_n}^{x_{n+1}} c\phi_{n+1}'\phi_n' dx, \int_{x_n}^{x_{n+1}} c\phi_n'\phi_{n+1}' dx, \int_{x_n}^{x_{n+1}} c\phi_{n+1}'\phi_{n+1}' dx.$$

- They can be rewritten as

$$\int_{x_n}^{x_{n+1}} c\phi_j'\phi_i' \, dx \ (i, j = n, n+1).$$

Local assembly

- Recall

$$\phi_1(x) = \begin{cases} \frac{x_2-x}{h}, & if\ x_1 \le x \le x_2, \\ 0, & otherwise, \end{cases}$$

$$\phi_j(x) = \begin{cases} \frac{x-x_{j-1}}{h}, & if\ x_{j-1} \le x \le x_j, \\ \frac{x_{j+1}-x}{h}, & if\ x_j \le x \le x_{j+1}, \\ 0, & otherwise. \end{cases}$$

$$(i = 2, \cdots, N)$$

$$\phi_{N+1}(x) = \begin{cases} \frac{x-x_N}{h}, & if\ x_N \le x \le x_{N+1}, \\ 0, & otherwise, \end{cases}$$

- Define two local linear basis functions:

$$\psi_{n1} = \phi_n|_{E_n} = \frac{x_{n+1} - x}{h}, \quad \psi_{n2} = \phi_{n+1}|_{E_n} = \frac{x - x_n}{h}.$$

So in one element, the number of local basis functions $N_{lb} = 2$.

Local assembly

- Then the only four non-zero local integrals become

$$\int_{x_n}^{x_{n+1}} c\psi'_{n1}\psi'_{n1}\,dx, \int_{x_n}^{x_{n+1}} c\psi'_{n2}\psi'_{n1}\,dx, \int_{x_n}^{x_{n+1}} c\psi'_{n1}\psi'_{n2}\,dx, \int_{x_n}^{x_{n+1}} c\psi'_{n2}\psi'_{n2}\,dx.$$

- That is, instead of the original four non-zero local integrals with the global basis functions ϕ_n and ϕ_{n+1}, we will compute the following four non-zero local integrals with the local basis functions ψ_{n1} and ψ_{n2}:

$$\int_{x_n}^{x_{n+1}} c\psi'_{n\alpha}\psi'_{n\beta}\,dx \ (\alpha,\beta=1,2).$$

- Question: how to compute these integrals?
- Gauss quadrature. The needed information is stored in the matrices P and T.

Local assembly

Assemble the non-zero local integrals into A:

- Based on Algorithm 1, when the trial function is ϕ_j and the test function is ϕ_i, the corresponding non-zero local integrals should be assembled to a_{ij}.
- For example, $\int_{x_n}^{x_{n+1}} c\phi_n'\phi_n' \, dx$ should be assemble to a_{nn}.
- $\int_{x_n}^{x_{n+1}} c\phi_{n+1}'\phi_n' \, dx$ should be assemble to $a_{n,n+1}$.
- $\int_{x_n}^{x_{n+1}} c\phi_n'\phi_{n+1}' \, dx$ should be assemble to $a_{n+1,n}$.
- $\int_{x_n}^{x_{n+1}} c\phi_{n+1}'\phi_{n+1}' \, dx$ should be assemble to $a_{n+1,n+1}$.

Local assembly

- Therefore, if we find the global node indices of the trial and test basis functions, we can easily locate where to assemble a non-zero local integral.
- Question: Since we compute

$$\int_{x_n}^{x_{n+1}} c\psi'_{n\alpha}\psi'_{n\beta}\,dx\ (\alpha,\beta=1,2)$$

instead of

$$\int_{x_n}^{x_{n+1}} c\phi'_j\phi'_i\,dx\ (i,j=n,n+1),$$

how do we obtain the corresponding global node indices of the local trial and test basis functions $\psi_{n\alpha}$ and $\psi_{n\beta}$ $(\alpha,\beta=1,2)$?

- Information matrix T_b!

Local assembly

- Recall that the n^{th} column of the matrix T_b stores the global node indices of the finite element nodes of the n^{th} mesh element:

$$T_b = \begin{pmatrix} 1 & 2 & \cdots & N-1 & N \\ 2 & 3 & \cdots & N & N+1 \end{pmatrix}.$$

- Hence $T_b(\alpha, n)$ and $T_b(\beta, n)$ give the global node indices of the local trial and test basis functions $\psi_{n\alpha}$ and $\psi_{n\beta}$ $(\alpha, \beta = 1, 2)$.
- That is, for $n = 1, \cdots, N$,

$$\int_{x_n}^{x_{n+1}} c\psi'_{n\alpha}\psi'_{n\beta}\, dx \ (\alpha, \beta = 1, 2)$$

should be assembled to a_{ij} where $i = T_b(\beta, n)$ and $j = T_b(\alpha, n)$.

Local assembly

Algorithm 4:

- Initialize the matrix: $A = sparse(N_b, N_b)$;
- Compute the integrals and assemble them into A:

$FOR\ n = 1, \cdots, N$:
  $FOR\ \alpha = 1, \cdots, N_{lb}$:
    $FOR\ \beta = 1, \cdots, N_{lb}$:
      Compute $r = \int_{x_n}^{x_{n+1}} c\psi'_{n\alpha}\psi'_{n\beta}\ dx$;
      Add r to $A(T_b(\beta, n), T_b(\alpha, n))$;
    END
  END
END

Local assembly

Algorithm 4 (alternative version):

- Initialize the matrix: $A = sparse(N_b, N_b)$ and $S = zeros(N_{lb}, N_{lb})$;
- Compute the integrals and assemble them into A:

$FOR\ n = 1, \cdots, N$:
    $FOR\ \alpha = 1, \cdots, N_{lb}$:
        $FOR\ \beta = 1, \cdots, N_{lb}$:
            Compute $S(\beta, \alpha) = \int_{x_n}^{x_{n+1}} c\psi'_{n\alpha}\psi'_{n\beta}\ dx$;
        END
    END
    $A(T_b(:, n), T_b(:, n)) = A(T_b(:, n), T_b(:, n)) + S$;
END

Local assembly

- If we follow Algorithm 4 to develop a subroutine to assemble the matrix arising from a more general integral

$$\int_{x_n}^{x_{n+1}} c\psi_{n\alpha}^{(r)}\psi_{n\beta}^{(s)}\, dx,$$

  then Algorithm 4 is equivalent to calling this subroutine with input parameters $r = s = 1$.
- Furthermore, in Petrov Galerkin method, the trial and test function spaces can be different.
- For example, consider the trial function space $span\{\varphi_{n\alpha}\}$ and test function space $span\{\psi_{n\beta}\}$. Then an even more general integral

$$\int_{x_n}^{x_{n+1}} c\varphi_{n\alpha}^{(r)}\psi_{n\beta}^{(s)}\, dx.$$

  Hence Algorithm 4 is equivalent to calling this subroutine with $\varphi_{n\alpha} = \psi_{n\alpha}$ and $r = s = 1$.

Local assembly

To make a general subroutine for different cases, more information needed for computing and assembling the integral should be treated as input parameters or input functions of this subroutine:

- the coefficient function c;
- the Gauss quadrature points and weights for numerical integrals;
- the mesh information matrices P and T, which can also provide the number of mesh elements $N = size(T, 2)$ and the number of mesh nodes $N_m = size(P, 2)$;

Local assembly

- the type of the basis functions, which can be different for the trial and test functions;

- the finite element information matrices P_b and T_b, which can also provide the number of local basis functions $N_{lb} = size(T_b, 1)$ and the number of the global basis functions $N_b = size(P_b, 2)$ (= the number of unknowns). (They can be different for the trial and test functions)

Local assembly

Algorithm 4 (upgraded version):

- Initialize the matrix: $A = sparse(N_b^{test}, N_b^{trial})$;
- Compute the integrals and assemble them into A:

$FOR\ n = 1, \cdots, N$:
  $FOR\ \alpha = 1, \cdots, N_{lb}^{trial}$:
    $FOR\ \beta = 1, \cdots, N_{lb}^{test}$:
      Compute $r = \int_{x_n}^{x_{n+1}} c\varphi_{n\alpha}^{(r)}\psi_{n\beta}^{(s)}\ dx$;
      Add r to $A(T_b^{test}(\beta, n), T_b^{trial}(\alpha, n))$;
    END
  END
END

Local assembly

Observation based on Algorithm 2 for the load vector $\vec{b}$:

- All the non-zero entries in the load vector $\vec{b}$ come from the non-zero local integrals defined on the mesh elements.
- In each non-zero local integral, the test basis functions are only corresponding to the nodes of the element which is the integral interval.
- On each element, all the local integrals, whose test basis functions are corresponding to the nodes of this element, have non-trivial contribution to some non-zero entries of the load vector $\vec{b}$.

Local assembly

New assembly idea for the load vector $\vec{b}$:

- Loop over all the elements;
- Compute all non-zero local integrals on each element for the load vector $\vec{b}$;
- Assemble these non-zero local integrals into the corresponding entries of the load vector $\vec{b}$.

Local assembly

Compute all non-zero local integrals on each element for $\vec{b}$:

- On the n^{th} element $E_n = [x_n, x_{n+1}]$, we get non-zero local integrals only when the test basis functions are corresponding to the finite element nodes of the element.
- That is, we only consider the test basis functions to be ϕ_n or ϕ_{n+1}.
- There are only two non-zero local integrals on E_n with the global basis functions ϕ_n and ϕ_{n+1}:

$$\int_{x_n}^{x_{n+1}} f\phi_n \, dx, \int_{x_n}^{x_{n+1}} f\phi_{n+1} \, dx.$$

- They can be rewritten as

$$\int_{x_n}^{x_{n+1}} f\phi_i \, dx \ (i = n, n+1).$$

Local assembly

- Using ψ_{n1} and ψ_{n2}, these two non-zero local integrals become

$$\int_{x_n}^{x_{n+1}} f\psi_{n1}\, dx, \quad \int_{x_n}^{x_{n+1}} f\psi_{n2}\, dx.$$

- That is, instead of the original two non-zero local integrals with the global basis functions ϕ_n and ϕ_{n+1}, we will compute the following two non-zero local integrals with the local basis functions ψ_{n1} and ψ_{n2}:

$$\int_{x_n}^{x_{n+1}} f\psi_{n\beta}\, dx \ (\beta = 1, 2).$$

- Question: how to compute these integrals?
- Gauss quadrature. The needed information is stored in the matrices P and T.

Local assembly

Assemble the non-zero local integrals into $\vec{b}$:

- Based on Algorithm 1, when the test function is ϕ_i, the corresponding non-zero local integrals should be assembled to b_i.
- For example, $\int_{x_n}^{x_{n+1}} f\phi_n \, dx$ should be assemble to b_n.
- $\int_{x_n}^{x_{n+1}} f\phi_{n+1} \, dx$ should be assemble to b_{n+1}.
- Therefore, if we find the global node indices of test basis functions, we can easily locate where to assemble a non-zero local integral.

Local assembly

- Question: Since we compute

$$\int_{x_n}^{x_{n+1}} f\psi_{n\beta}\ dx\ (\beta = 1, 2)$$

instead of

$$\int_{x_n}^{x_{n+1}} f\phi_i\ dx\ (i = n, n+1),$$

how do we obtain the corresponding global node indices of the local test basis functions $\psi_{n\beta}\ (\beta = 1, 2)$?

- Information matrix T_b!

Local assembly

- Recall that the n^{th} column of the matrix T_b stores the global node indices of the finite element nodes of the n^{th} mesh element:

$$T_b = \begin{pmatrix} 1 & 2 & \cdots & N-1 & N \\ 2 & 3 & \cdots & N & N+1 \end{pmatrix}.$$

- Hence $T_b(\beta, n)$ gives the global node indices of the local test basis functions $\psi_{n\beta}$ $(\beta = 1, 2)$.
- That is, for $n = 1, \cdots, N$,

$$\int_{x_n}^{x_{n+1}} f\psi_{n\beta}\, dx \ (\beta = 1, 2)$$

should be assembled to b_i where $i = T_b(\beta, n)$.

Local assembly

Algorithm 5:

- Initialize the vector: $b = zeros(N_b, 1)$;
- Compute the integrals and assemble them into $\vec{b}$:

$FOR\ n = 1, \cdots, N$:
  $FOR\ \beta = 1, \cdots, N_{lb}$:
    Compute $r = \int_{x_n}^{x_{n+1}} f\psi_{n\beta}\ dx$;
    $b(T_b(\beta, n), 1) = b(T_b(\beta, n), 1) + r$;
  END
END

Local assembly

Algorithm 5 (alternative version):

- Initialize the vector: $b = zeros(N_b, 1)$ and $d = zeros(N_{lb}, 1)$;
- Compute the integrals and assemble them into $\vec{b}$:

$FOR\ n = 1, \cdots, N$:
    $FOR\ \beta = 1, \cdots, N_{lb}$:
        Compute $d(\beta, 1) = \int_{x_n}^{x_{n+1}} f\psi_{n\beta}\ dx$;
    END
    $b(T_b(:, n), 1) = b(T_b(:, n), 1) + d$;
END

Local assembly

- If we follow Algorithm 5 to develop a subroutine to assemble the vector arising from a more general integral

$$\int_{x_n}^{x_{n+1}} f\psi_{n\beta}^{(s)}\, dx,$$

then Algorithm 5 is equivalent to calling this subroutine with parameter $s = 0$.

Local assembly

To make a general subroutine for different cases, more information needed for computing and assembling the integral should be treated as input parameters or input functions of this subroutine:

- the right hand side function f;
- the quadrature points and weights for numerical integrals;
- the mesh information matrices P and T, which can also provide the number of mesh elements $N = size(T, 2)$ and the number of mesh nodes $N_m = size(P, 2)$;
- the type of the basis function for the test functions.
- the finite element information matrices P_b and T_b for the test functions, which can also provide the number of local basis functions $N_{lb} = size(T_b, 1)$ and the number of the global basis functions $N_b = size(P_b, 2)$ (= the number of unknowns);

Local assembly

Algorithm 5 (upgraded version):

- Initialize the vector: $b = zeros(N_b, 1)$;
- Compute the integrals and assemble them into $\vec{b}$:

$FOR\ n = 1, \cdots, N$:
  $FOR\ \beta = 1, \cdots, N_{lb}$:
    Compute $r = \int_{x_n}^{x_{n+1}} f\psi_{n\beta}^{(s)}\ dx$;
    $b(T_b(\beta, n), 1) = b(T_b(\beta, n), 1) + r$;
  END
END

Treat boundary conditions

- Boundary information matrix $boundarynodes$:
- $boundarynodes(1,k)$ is the type of the k^{th} boundary finite element node: Dirichlet, Neumann, Robin.....
- $boundarynodes(2,k)$ is the global node index of the k^{th} boundary finite element node.
- Set nbn to be the number of boundary finite element nodes;
- Define $g(x)$ to be the boundary function which satisfies $g(a) = g_a$ and $g(b) = g_b$;
- Algorithm 3 can be reorganized into a more general framework by using the boundary information matrix $boundarynodes$.

Treat boundary conditions

Algorithm 6:

- Deal with the Dirichlet boundary conditions:

$FOR\ k = 1, \cdots, nbn$:
  $IF\ boundarynodes(1,k)$ shows Dirichlet condition,
$THEN$
    $i = boundarynodes(2,k);$
    $A(i,:) = 0;$
    $A(i,i) = 1;$
    $b(i) = g(P_b(i));$
  $ENDIF$
END

Enriched algorithm

Recall Algorithm 4 (upgraded version):

- Initialize the matrix: $A = sparse(N_b^{test}, N_b^{trial})$;
- Compute the integrals and assemble them into A:

$FOR\ n = 1, \cdots, N$:
  $FOR\ \alpha = 1, \cdots, N_{lb}^{trial}$:
    $FOR\ \beta = 1, \cdots, N_{lb}^{test}$:
      Compute $r = \int_{x_n}^{x_{n+1}} c\psi_{n\alpha}^{(r)}\psi_{n\beta}^{(s)}\ dx$;
      Add r to $A(T_b(\beta, n), T_b(\alpha, n))$;
    END
  END
END

Enriched algorithm

Recall Algorithm 5 (upgraded version):

- Initialize the vector: $b = zeros(N_b, 1)$;
- Compute the integrals and assemble them into $\vec{b}$:

$FOR\ n = 1, \cdots, N$:
    $FOR\ \beta = 1, \cdots, N_{lb}$:
        Compute $r = \int_{x_n}^{x_{n+1}} f\psi_{n\beta}^{(s)}\ dx$;
        $b(T_b(\beta, n), 1) = b(T_b(\beta, n), 1) + r$;
    END
END

Enriched algorithm

- Input a, b, and N. Generate the mesh information matrices P and T, the finite element information matrices P_b and T_b for the trial and test functions.
- Assemble the stiffness matrix A by using Algorithm 4.
- Assemble the load vector $\vec{b}$ by using Algorithm 5.
- Deal with the Drichlet boundary condition by using Algorithm 6.
- Solve $A\vec{X} = \vec{b}$ for $\vec{X}$ by using a direct or iterative method.

Numerical example

- Example 1: Use the 1D linear finite element method to solve the following equation:

$$-\frac{d}{dx}\left(e^x \frac{du(x)}{dx}\right) = -e^x[cos(x) - 2sin(x) - x\ cos(x) - x\ sin(x)]\ (0 \leq x \leq 1),$$
$$u(0) = 0, u(1) = cos(1).$$

- The analytic solution of this problem is $u = x\ cos(x)$, which can be used to compute the error of the numerical solution.
- Let's code for the linear finite element method for 1D elliptic equation together!
- Open your Matlab!

Numerical example

h	maximum absolute error at all nodes
$1/4$	2.3340×10^{-3}
$1/8$	5.8317×10^{-4}
$1/16$	1.4645×10^{-4}
$1/32$	3.6675×10^{-5}
$1/64$	9.1700×10^{-6}
$1/128$	2.2929×10^{-6}

Table: The maximum numerical errors at all mesh nodes.

- Any Observation?

Numerical example

- Second order convergence $O(h^2)$ since the error is reduced by $\frac{1}{4}$ when h is reduced by half.
- This matches the optimal approximation capability expected from piecewise linear functions.

Outline

Basic framework

- A "reference→ local → global" framework will be introduced to construct the finite element spaces.
- Since all the integrals in the discretization formulation are locally computed on the mesh elements, it is critical to have a convenient formulation of the local basis functions on all the mesh elements.
- But we still need the concept of the global basis functions theoretically.
- In the following, we will first introduce the "local → global" framework to construct the 1D linear finite element space by defining the local basis functions in a direct way. Later we will introduce the "reference→ local" framework for defining the local basis functions in another way.

Reconstruct 1D linear finite element space

Recall:

- Assume that we have a uniform partition of $[a, b]$ into N elements with mesh size $h = \frac{b-a}{N}$.
- Let $x_i = a + (i-1)h$ $(i = 1, \cdots, N+1)$ denote the mesh nodes, which are also the finite element nodes of the 1D linear finite elements.
- Let $E_n = [x_n, x_{n+1}]$ $(n = 1, \cdots, N)$ denote the mesh elements.
- Let N_m denote the number of mesh nodes. Here $N_m = N + 1$.
- Let N_b denote the number of global finite element basis functions. Here $N_b = N + 1$.
- Let N_{lb} denote the number of local finite element basis functions in one element. Here $N_{lb} = 2$.

Reconstruct 1D linear finite element space

- For the above mesh and 1D linear finite element, we recall

$$P_b = P = \begin{pmatrix} x_1 & x_2 & \cdots & x_N & x_{N+1} \end{pmatrix},$$
$$T_b = T = \begin{pmatrix} 1 & 2 & \cdots & N-1 & N \\ 2 & 3 & \cdots & N & N+1 \end{pmatrix}.$$

Reconstruct 1D linear finite element space

- On each mesh element $E_n = [x_n, x_{n+1}] = [A_{n1}, A_{n2}]$ $(n = 1, \cdots, N)$, we define two local linear basis functions

$$\psi_{n1}(x) = a_{n1}x + b_{n1} \text{ and } \psi_{n2}(x) = a_{n2}x + b_{n2}$$

such that

$$\psi_{nj}(A_{ni}) = \delta_{ij} = \begin{cases} 0, & \text{if } j \neq i, \\ 1, & \text{if } j = i. \end{cases}$$

for $i, j = 1, 2$.

- Then it's easy to obtain

$$\begin{aligned} \psi_{n1}(x_n) = 1 &\Rightarrow a_{n1}x_n + b_{n1} = 1, \\ \psi_{n1}(x_{n+1}) = 0 &\Rightarrow a_{n1}x_{n+1} + b_{n1} = 0, \\ \psi_{n2}(x_n) = 0 &\Rightarrow a_{n2}x_n + b_{n2} = 0, \\ \psi_{n2}(x_{n+1}) = 1 &\Rightarrow a_{n2}x_{n+1} + b_{n2} = 1. \end{aligned}$$

Reconstruct 1D linear finite element space

- Solve the 4×4 system to get

$$a_{n1} = \frac{-1}{x_{n+1} - x_n}, b_{n1} = \frac{x_{n+1}}{x_{n+1} - x_n}, a_{n2} = \frac{1}{x_{n+1} - x_n}, b_{n2} = \frac{-x_n}{x_{n+1} - x_n}.$$

- Hence

$$\psi_{n1}(x) = \frac{x_{n+1} - x}{x_{n+1} - x_n}, \quad \psi_{n2}(x) = \frac{x - x_n}{x_{n+1} - x_n}.$$

- Since $x_{n+1} - x_n = h$, then the two local linear basis functions are

$$\begin{aligned} \psi_{n1}(x) &= \frac{x_{n+1} - x}{h}, \\ \psi_{n2}(x) &= \frac{x - x_n}{h}, \end{aligned}$$

which match the non-zero pieces of the global linear basis functions obtained in Chapter 1.

Reconstruct 1D linear finite element space

"local $\rightarrow$ global" framework:

- Define the local finite element space

$$S_h(E_n) = span\{\psi_{n1}, \psi_{n2}\}.$$

- At each finite element node x_j $(j = 1, \cdots, N+1)$, define the corresponding global linear basis function ϕ_j such that $\phi_j|_{E_n} \in S_h(E_n)$ and

$$\phi_j(x_i) = \delta_{ij} = \begin{cases} 0, & \text{if } j \neq i, \\ 1, & \text{if } j = i, \end{cases}$$

  for $i, j = 1, \cdots, N+1$.

- Then define the global finite element space to be

$$U_h = span\{\phi_j\}_{j=1}^{N+1}.$$

Reconstruct 1D linear finite element space

- In fact,

$$\phi_j|_{E_n} = \begin{cases} \psi_{n1}, & \text{if } j = n, \\ \psi_{n2}, & \text{if } j = n+1, \\ 0, & otherwise, \end{cases}$$

for $j = 1, \cdots, N+1$ and $n = 1, \cdots, N$.

- That is,

$$\phi_1 = \begin{cases} \psi_{11}, & \text{on } E_1, \\ 0, & otherwise, \end{cases}$$

$$\phi_j = \begin{cases} \psi_{n1}, & \text{on } E_n \text{ such that } j = n, \\ \psi_{n2}, & \text{on } E_n \text{ such that } j = n+1, \\ 0, & otherwise, \end{cases} \quad j = 2, \cdots, N;$$

$$\phi_{N+1} = \begin{cases} \psi_{N2}, & \text{on } E_N, \\ 0, & otherwise, \end{cases}$$

Reconstruct 1D linear finite element space

- Hence

$$\phi_1 = \begin{cases} \psi_{11}, & \text{on } E_1 = [x_1, x_2], \\ 0, & otherwise, \end{cases}$$

$$\phi_j = \begin{cases} \psi_{j1}, & \text{on } E_j = [x_j, x_{j+1}], \\ \psi_{j-1,2}, & \text{on } E_{j-1} = [x_{j-1}, x_j], \quad j = 2, \cdots, N; \\ 0, & otherwise, \end{cases}$$

$$\phi_{N+1} = \begin{cases} \psi_{N2}, & \text{on } E_N = [x_N, x_{N+1}], \\ 0, & otherwise, \end{cases}$$

- For each x_j $(j = 2, \cdots, N)$, there are two local basis functions which are defined to be 1 at x_j. One is the ψ_{j1} defined on the element $E_j = [x_j, x_{j+1}]$. The other one is the $\psi_{j-1,2}$ defined on the element $E_{j-1} = [x_{j-1}, x_j]$. These two local basis functions form the non-zero part of ϕ_j on the elements $E_j = [x_j, x_{j+1}]$ and $E_{j-1} = [x_{j-1}, x_j]$ while ϕ_j is 0 everywhere else.

Reconstruct 1D linear finite element space

- Since

$$\psi_{n1}(x) = \frac{x_{n+1} - x}{h}, \quad \psi_{n2}(x) = \frac{x - x_n}{h},$$

then

$$\phi_1(x) = \begin{cases} \frac{x_2 - x}{h}, & if\ x_1 \le x \le x_2, \\ 0, & otherwise, \end{cases}$$

$$\phi_j(x) = \begin{cases} \frac{x - x_{j-1}}{h}, & if\ x_{j-1} \le x \le x_j, \\ \frac{x_{j+1} - x}{h}, & if\ x_j \le x \le x_{j+1}, \\ 0, & otherwise, \end{cases}$$

$$(j = 2, \cdots, N)$$

$$\phi_{N+1}(x) = \begin{cases} \frac{x - x_N}{h}, & if\ x_N \le x \le x_{N+1}, \\ 0, & otherwise, \end{cases}$$

which are the same as the global basis functions defined in Chapter 1.

Reconstruct 1D linear finite element space

- Recall that the n^{th} column of the information matrix T_b is

$$\begin{pmatrix} n \\ n+1 \end{pmatrix}.$$

  which are the global node indices of the two finite element nodes $A_{n1} = x_n$ and $A_{n2} = x_{n+1}$ in the element $[x_n, x_{n+1}]$.

- Since the local basis functions ψ_{n1} and ψ_{n2} are one-to-one corresponding to the finite element nodes $A_{n1} = x_n$ and $A_{n2} = x_{n+1}$ in the element $[x_n, x_{n+1}]$, the n^{th} column of the information matrix T_b also gives the global indices of the local basis functions ψ_{n1} and ψ_{n2}, which are n and $n+1$.

Reconstruct 1D linear finite element space

- Hence

$$\phi_j|_{E_n} = \begin{cases} \psi_{n1}, & \text{if } j = T_b(1,n), \\ \psi_{n2}, & \text{if } j = T_b(2,n), \\ 0, & \text{otherwise.} \end{cases}$$

  for $j = 1, \cdots, N+1$ and $n = 1, \cdots, N$.

- This is the reason why we use $T_b(\alpha, n)$ and $T_b(\beta, n)$ $(\alpha, \beta = 1, 2)$ to give the global node indices of the local trial and test basis functions $\psi_{n\alpha}$ and $\psi_{n\beta}$ $(\alpha, \beta = 1, 2)$ of the n^{th} mesh element in Chapter 1!

Reconstruct 1D linear finite element space

- Now let's turn to the "reference $\rightarrow$ local" framework for defining the local basis functions in another way.
- Consider the reference interval $[\hat{A}_1, \hat{A}_2] = [0, 1]$.
- Define two reference linear basis functions $\hat{\psi}_1(\hat{x}) = a_1\hat{x} + b_1$ and $\hat{\psi}_2(\hat{x}) = a_2\hat{x} + b_2$ such that

$$\hat{\psi}_j(\hat{A}_i) = \delta_{ij} = \begin{cases} 0, & \text{if } j \neq i, \\ 1, & \text{if } j = i, \end{cases}$$

  for $i, j = 1, 2$.
- Then it's easy to obtain

$$\begin{aligned} \hat{\psi}_1(\hat{A}_1) = 1 &\Rightarrow b_1 = 1, \\ \hat{\psi}_1(\hat{A}_2) = 0 &\Rightarrow a_1 + b_1 = 0, \\ \hat{\psi}_2(\hat{A}_1) = 0 &\Rightarrow b_2 = 0, \\ \hat{\psi}_2(\hat{A}_2) = 1 &\Rightarrow a_2 + b_2 = 1. \end{aligned}$$

Reconstruct 1D linear finite element space

- Hence $a_1 = -1, b_1 = 1, a_2 = 1, b_2 = 0$ and

$$\begin{aligned} \hat{\psi}_1(\hat{x}) &= 1 - \hat{x}, \\ \hat{\psi}_2(\hat{x}) &= \hat{x}. \end{aligned}$$

- Now we can use the affine mapping to construct the local basis functions from the reference ones.
- If $x \in [a, b]$, then

$$a \le x \le b \Rightarrow 0 \le x - a \le b - a \Rightarrow 0 \le \frac{x-a}{b-a} \le 1.$$

- Let $\hat{x} = \frac{x-a}{b-a}$. Then

$$\hat{x} \in [0, 1], \quad x = (b-a)\hat{x} + a.$$

Reconstruct 1D linear finite element space

- For a given function $\hat{\psi}(\hat{x})$ where $\hat{x} \in [0,1]$, we can define the corresponding function for $x \in [a,b]$ as follows:

$$\psi(x) = \hat{\psi}(\hat{x}) = \hat{\psi}(\frac{x-a}{b-a}).$$

- Consider $[a,b] = [x_n, x_{n+1}]$. Then $\hat{x} = \frac{x-x_n}{x_{n+1}-x_n} = \frac{x-x_n}{h}$.
- From the above affine mapping and the reference basis functions

$$\hat{\psi}_1(\hat{x}) = 1 - \hat{x} \text{ and } \hat{\psi}_2(\hat{x}) = \hat{x},$$

we can use the "reference $\rightarrow$ local" framework to obtain the same local basis functions as before:

$$\begin{aligned}
\psi_{n1}(x) &= \hat{\psi}_1(\hat{x}) = \hat{\psi}_1(\frac{x-x_n}{h}) \\
&= 1 - \frac{x-x_n}{h} = \frac{h-x+x_n}{h} = \frac{x_{n+1}-x}{h}, \\
\psi_{n2}(x) &= \hat{\psi}_2(\hat{x}) = \hat{\psi}_2(\frac{x-x_n}{h}) = \frac{x-x_n}{h}.
\end{aligned}$$

Reconstruct 1D linear finite element space

- The affine mapping actually maps

$$\begin{aligned} \hat{A}_1 = 0 \quad &\rightarrow \quad A_{n1} = x_n, \\ \hat{A}_2 = 1 \quad &\rightarrow \quad A_{n2} = x_{n+1}. \end{aligned}$$

- It is easy to verify that $\psi_{nj}(x)$ $(j = 1, 2)$ are linear functions and

$$\psi_{nj}(A_{ni}) = \delta_{ij} = \begin{cases} 0, & \text{if } j \neq i, \\ 1, & \text{if } j = i. \end{cases}$$

  for $i, j = 1, 2$.

- Remark: If you want to use the reference basis functions $\hat{\psi}_j$ and the affine mapping $\hat{x} = \frac{x - x_n}{h}$ to provide the local basis functions $\psi_{nj}(x) = \hat{\psi}_j(\hat{x})$ instead of directly using the local basis functions, you will need to use chain rule to obtain the derivative of the local basis functions. For example, $\frac{d\psi_{nj}(x)}{dx} = \frac{d\hat{\psi}_j(\hat{x})}{d\hat{x}} \frac{d\hat{x}}{dx}$.

Reconstruct 1D linear finite element space

- Once the local basis functions are obtained by using the "reference $\rightarrow$ local" framework, we can use the "local $\rightarrow$ global" framework discussed before to obtain the 1D linear finite element space.

- This is the so called "reference$\rightarrow$ local $\rightarrow$ global" framework.

Reconstruct 1D linear finite element space

Summary of three ways for the global finite element basis functions:

- Directly define the global finite element basis functions globally. This is not a general way.
- Define local finite element basis functions directly on the local elements and then use them to form the global basis functions. I will use this way in my solution of 1D equations.
- Define local finite element basis functions by using the reference element and affine mapping and then use them to form the global basis functions. I will use this way in my solution of 2D equations.

Reconstruct 1D linear finite element space

Two structures to represent the local basis functions in code:

- "function" style: Use a subroutine with different parameters as a function to describe all the local basis functions; then evaluate the subroutine when we need to evaluate the local basis functions at needed points. I will use this style in my solution.

- "coefficient" style: Only store the coefficients of all the local basis functions; then use these coefficients to evaluate the local basis functions at needed points.

1D quadratic finite element space

- We first consider the reference quadratic basis functions on the reference interval $[\hat{A}_1, \hat{A}_2] = [0, 1]$ with $\hat{A}_3 = \frac{1}{2}$.
- Define three reference quadratic basis functions

$$\begin{aligned} \hat{\psi}_1(\hat{x}) &= a_1\hat{x}^2 + b_1\hat{x} + c_1, \\ \hat{\psi}_2(\hat{x}) &= a_2\hat{x}^2 + b_2\hat{x} + c_2, \\ \hat{\psi}_3(\hat{x}) &= a_3\hat{x}^2 + b_3\hat{x} + c_3, \end{aligned}$$

such that

$$\hat{\psi}_j(\hat{A}_i) = \delta_{ij} = \begin{cases} 0, & \text{if } j \neq i, \\ 1, & \text{if } j = i, \end{cases}$$

for $i, j = 1, 2, 3$.

1D quadratic finite element space

- Then it's easy to obtain

$$
\begin{aligned}
\hat{\psi}_1(\hat{A}_1) = 1 \quad &\Rightarrow \quad c_1 = 1, \\
\hat{\psi}_1(\hat{A}_2) = 0 \quad &\Rightarrow \quad a_1 + b_1 + c_1 = 0, \\
\hat{\psi}_1(\hat{A}_3) = 0 \quad &\Rightarrow \quad \frac{1}{4}a_1 + \frac{1}{2}b_1 + c_1 = 0, \\
\hat{\psi}_2(\hat{A}_1) = 0 \quad &\Rightarrow \quad c_2 = 0, \\
\hat{\psi}_2(\hat{A}_2) = 1 \quad &\Rightarrow \quad a_2 + b_2 + c_2 = 1, \\
\hat{\psi}_2(\hat{A}_3) = 0 \quad &\Rightarrow \quad \frac{1}{4}a_2 + \frac{1}{2}b_2 + c_2 = 0, \\
\hat{\psi}_3(\hat{A}_1) = 0 \quad &\Rightarrow \quad c_3 = 0, \\
\hat{\psi}_3(\hat{A}_2) = 0 \quad &\Rightarrow \quad a_3 + b_3 + c_3 = 0, \\
\hat{\psi}_3(\hat{A}_3) = 1 \quad &\Rightarrow \quad \frac{1}{4}a_3 + \frac{1}{2}b_3 + c_3 = 1.
\end{aligned}
$$

1D quadratic finite element space

- Hence

$$a_1 = 2, b_1 = -3, c_1 = 1,$$
$$a_2 = 2, b_2 = -1, c_2 = 0,$$
$$a_3 = -4, b_3 = 4, c_3 = 0.$$

- Then the three reference quadratic basis functions are

$$\begin{aligned} \hat{\psi}_1(\hat{x}) &= 2\hat{x}^2 - 3\hat{x} + 1, \\ \hat{\psi}_2(\hat{x}) &= 2\hat{x}^2 - \hat{x}, \\ \hat{\psi}_3(\hat{x}) &= -4\hat{x}^2 + 4\hat{x}. \end{aligned}$$

1D quadratic finite element space

- Now we turn to the local quadratic basis functions based on the above reference quadratic basis functions.
- Assume that we have a uniform partition of $[a, b]$ into N elements with mesh size $h = \frac{b-a}{N}$.
- Let $x_i = a + (i-1)h$ $(i = 1, \cdots, N+1)$ denote the mesh nodes.
- Let $E_n = [x_n, x_{n+1}]$ $(n = 1, \cdots, N)$ denote the mesh elements.
- Let N_m denote the number of mesh nodes. Here $N_m = N + 1$.

1D quadratic finite element space

- For the above mesh, we recall

$$
\begin{aligned}
P &= \begin{pmatrix} x_1 & x_2 & \cdots & x_{N_m-1} & x_{N_m} \end{pmatrix} \\
&= \begin{pmatrix} x_1 & x_2 & \cdots & x_N & x_{N+1} \end{pmatrix}, \\
T &= \begin{pmatrix} 1 & 2 & \cdots & N_m-2 & N_m-1 \\ 2 & 3 & \cdots & N_m-1 & N_m \end{pmatrix} \\
&= \begin{pmatrix} 1 & 2 & \cdots & N-1 & N \\ 2 & 3 & \cdots & N & N+1 \end{pmatrix}.
\end{aligned}
$$

1D quadratic finite element space

- The finite element nodes of 1D quadratic finite elements include all the mesh nodes and the middle points of all the mesh elements.
- Let $y_k = a + (k-1)h/2$ $(k = 1, \cdots, N_b)$ denote the finite element nodes where $N_b = 2N + 1$ is the number of global finite element basis functions.
- It is easy to see

$$x_i = y_{2i-1}.$$

- Also, each mesh element $E_n = [x_n, x_{n+1}]$ includes three finite element nodes:

$$y_{2n-1},\ y_{2n+1},\ y_{2n}.$$

- Let N_{lb} denote the number of local finite element basis functions in one element. Here $N_{lb} = 3$.

1D quadratic finite element space

- For the 1D quadratic finite elements, we use the j^{th} column of the matrix P_b to store the coordinates of the j^{th} finite element node and the n^{th} column of the matrix T_b to store the global node indices of the finite element nodes of the n^{th} mesh element

$$
\begin{aligned}
P_b &= \begin{pmatrix} y_1 & y_2 & \cdots & y_{N_b-1} & y_{N_b} \end{pmatrix} \\
&= \begin{pmatrix} y_1 & y_2 & \cdots & y_{2N} & y_{2N+1} \end{pmatrix}, \\
T_b &= \begin{pmatrix} 1 & 3 & \cdots & N_b-3 & N_b-2 \\ 3 & 5 & \cdots & N_b-1 & N_b \\ 2 & 4 & \cdots & N_b-2 & N_b-1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 3 & \cdots & 2N-3 & 2N-1 \\ 3 & 5 & \cdots & 2N-1 & 2N+1 \\ 2 & 4 & \cdots & 2N-2 & 2N \end{pmatrix}.
\end{aligned}
$$

1D quadratic finite element space

- Recall the affine mapping between $x \in [x_n, x_{n+1}]$ and $\hat{x} \in [0, 1]$:

$$\hat{x} = \frac{x - x_n}{x_{n+1} - x_n} = \frac{x - x_n}{h}, \quad \psi(x) = \hat{\psi}(\hat{x}) = \hat{\psi}(\frac{x - x_n}{h}).$$

- Then the three local quadratic basis functions on the element $E_n = [x_n, x_{n+1}]$ are

$$\begin{aligned}
\psi_{n1}(x) &= \hat{\psi}_1(\hat{x}) = \hat{\psi}_1(\frac{x - x_n}{h}) = 2\left(\frac{x - x_n}{h}\right)^2 - 3\frac{x - x_n}{h} + 1, \\
\psi_{n2}(x) &= \hat{\psi}_2(\hat{x}) = \hat{\psi}_2(\frac{x - x_n}{h}) = 2\left(\frac{x - x_n}{h}\right)^2 - \frac{x - x_n}{h}, \\
\psi_{n3}(x) &= \hat{\psi}_3(\hat{x}) = \hat{\psi}_3(\frac{x - x_n}{h}) = -4\left(\frac{x - x_n}{h}\right)^2 + 4\frac{x - x_n}{h}.
\end{aligned}$$

1D quadratic finite element space

- The affine mapping actually maps

$$\begin{aligned}
\hat{A}_1 = 0 \quad &\rightarrow \quad A_{n1} = y_{2n-1} = x_n, \\
\hat{A}_2 = 1 \quad &\rightarrow \quad A_{n2} = y_{2n+1} = x_{n+1}, \\
\hat{A}_3 = \frac{1}{2} \quad &\rightarrow \quad A_{n3} = y_{2n} = \frac{x_n + x_{n+1}}{2}.
\end{aligned}$$

- It's also easy to verify that $\psi_{nj}(x)$ $(j = 1, 2, 3)$ are quadratic functions and

$$\psi_{nj}(A_{ni}) = \delta_{ij} = \begin{cases} 0, & \text{if } j \neq i, \\ 1, & \text{if } j = i, \end{cases}$$

for $i, j = 1, 2, 3$.

1D quadratic finite element space

- Remark: If you want to use the reference basis functions $\hat{\psi}_j$ and the affine mapping $\hat{x} = \frac{x - x_n}{h}$ to provide the local basis functions $\psi_{nj}(x) = \hat{\psi}_j(\hat{x})$ instead of directly using the local basis functions, you will need to use chain rule to obtain the derivative of the local basis functions. For example, $\frac{d\psi_{nj}(x)}{dx} = \frac{d\hat{\psi}_j(\hat{x})}{d\hat{x}} \frac{d\hat{x}}{dx}$.

1D quadratic finite element space

- Define the local finite element space

$$S_h(E_n) = span\{\psi_{n1}, \psi_{n2}, \psi_{n3}\}.$$

- At each finite element node y_j $(j = 1, \cdots, 2N+1)$, define the corresponding global linear basis function ϕ_j such that $\phi_j|_{E_n} \in S_h(E_n)$ and

$$\phi_j(y_i) = \delta_{ij} = \begin{cases} 0, & \text{if } j \neq i, \\ 1, & \text{if } j = i. \end{cases}$$

  for $i, j = 1, \cdots, 2N+1$.

- Then define the global finite element space to be

$$U_h = span\{\phi_j\}_{j=1}^{2N+1}.$$

1D quadratic finite element space

- In fact,

$$\phi_j|_{E_n} = \begin{cases} \psi_{n1}, & \text{if } j = 2n-1, \\ \psi_{n2}, & \text{if } j = 2n+1, \\ \psi_{n3}, & \text{if } j = 2n, \\ 0, & otherwise, \end{cases}$$

for $j = 1, \cdots, 2N+1$ and $n = 1, \cdots, N$.

1D quadratic finite element space

- Recall that the n^{th} column of the information matrix T_b is

$$\begin{pmatrix} 2n-1 \\ 2n+1 \\ 2n \end{pmatrix}.$$

  which are the global node indices of the two finite element nodes $A_{n1} = y_{2n-1}$, $A_{n2} = y_{2n+1}$, and $A_{n3} = y_{2n}$ in the element $[x_n, x_{n+1}]$.

- Since the local basis functions ψ_{n1}, ψ_{n2}, and ψ_{n3} are one-to-one corresponding to the finite element nodes $A_{n1} = y_{2n-1}$, $A_{n2} = y_{2n+1}$, and $A_{n3} = y_{2n}$ in the element $[x_n, x_{n+1}]$, the n^{th} column of the information matrix T_b also gives the global indices of the local basis functions ψ_{n1}, ψ_{n2}, and ψ_{n3} which are $2n-1$, $2n+1$ and $2n$.

1D quadratic finite element space

- Hence

$$\phi_j|_{E_n} = \begin{cases} \psi_{n1}, & \text{if } j = T_b(1,n), \\ \psi_{n2}, & \text{if } j = T_b(2,n), \\ \psi_{n3}, & \text{if } j = T_b(3,n), \\ 0, & otherwise, \end{cases}$$

for $j = 1, \cdots, 2N+1$ and $n = 1, \cdots, N$.

General 1D finite element method

- Question: Can we dynamically incorporate linear, quadratic and even more 1D finite elements on different meshes into one code of a general framework since they are so similar?
- Answer: Yes! We have actually done so when we coded for the 1D linear finite element method!
- This is because in our 1D linear finite element code we have designed many flexible input parameters and input functions: r, s, c, N, N_m, N_b, N_{lb}, P, T, P_b, T_b, and type of finite elements!

General 1D finite element method

- r, s and c depend on the equation only;
- P, T, N and N_m depend on the mesh only;
- P_b, T_b, N_b and N_{lb} depend on the type of finite elements and the mesh;
- For a new type of finite elements, we need to add the basis functions into the code!

General 1D finite element method

- Example 2: Use the 1D quadratic finite element method to solve the following equation:

$$
\begin{aligned}
&-\frac{d}{dx}\left(e^x \frac{du(x)}{dx}\right) \\
&= -e^x[cos(x) - 2sin(x) - x\ cos(x) - x\ sin(x)]\ (0 \leq x \leq 1), \\
&u(0) = 0, u(1) = cos(1).
\end{aligned}
$$

- The analytic solution of this problem is $u = x\ cos(x)$, which can be used to compute the error of the numerical solution.
- Let's code for the quadratic finite element method for 1D elliptic equation together!
- Open your Matlab!

General 1D finite element method

h	maximum absolute error at all nodes
$1/4$	4.6597×10^{-5}
$1/8$	2.9918×10^{-6}
$1/16$	1.8901×10^{-7}
$1/32$	1.1869×10^{-8}
$1/64$	7.4356×10^{-10}
$1/128$	4.6623×10^{-11}

Table: The maximum numerical errors at all mesh nodes.

- Any Observation?

General 1D finite element method

- Third order convergence $O(h^3)$ since the error is reduced by at least $\frac{1}{8}$ when h is reduced by half.
- This matches the optimal approximation capability expected from piecewise quadratic functions.
- In fact, we observe superconvergence since the convergence order is almost $O(h^4)$.

1D cubic finite element space

- We consider the reference cubic basis functions on the reference interval $[\hat{A}_1, \hat{A}_2] = [0, 1]$.
- Define four reference quadratic basis functions

$$\begin{aligned}
\hat{\psi}_1(\hat{x}) &= a_1\hat{x}^3 + b_1\hat{x}^2 + c_1\hat{x} + d_1, \\
\hat{\psi}_2(\hat{x}) &= a_2\hat{x}^3 + b_2\hat{x}^2 + c_2\hat{x} + d_2, \\
\hat{\psi}_3(\hat{x}) &= a_3\hat{x}^3 + b_3\hat{x}^2 + c_3\hat{x} + d_3, \\
\hat{\psi}_4(\hat{x}) &= a_4\hat{x}^3 + b_4\hat{x}^2 + c_4\hat{x} + d_4,
\end{aligned}$$

such that

$$\begin{aligned}
&\hat{\psi}_1(\hat{A}_1) = 1, \hat{\psi}_1(\hat{A}_2) = 0, \hat{\psi}_1'(\hat{A}_1) = 0, \hat{\psi}_1'(\hat{A}_2) = 0; \\
&\hat{\psi}_2(\hat{A}_1) = 0, \hat{\psi}_2(\hat{A}_2) = 1, \hat{\psi}_2'(\hat{A}_1) = 0, \hat{\psi}_2'(\hat{A}_2) = 0; \\
&\hat{\psi}_3(\hat{A}_1) = 0, \hat{\psi}_3(\hat{A}_2) = 0, \hat{\psi}_3'(\hat{A}_1) = 1, \hat{\psi}_3'(\hat{A}_2) = 0; \\
&\hat{\psi}_4(\hat{A}_1) = 0, \hat{\psi}_4(\hat{A}_2) = 0, \hat{\psi}_4'(\hat{A}_1) = 0, \hat{\psi}_4'(\hat{A}_2) = 1.
\end{aligned}$$

1D cubic finite element space

- This is a Hermite type of finite elements since the definition of the basis functions involves with the derivatives. The linear and quadratic elements discussed before are Lagrange type of finite elements since the definition of the basis functions involves with the nodal values only.

Neumann/Robin boundary conditions

- Consider

$$-\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x)\ (a \leq x \leq b), u'(a) = r_a, u(b) = g_b.$$

- Recall

$$-c(b)u'(b)v(b) + c(a)u'(a)v(a) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx.$$

- Since the solution at $x = b$ is given by $u(b) = g_b$, then we can choose the test function $v(x)$ such that $v(b) = 0$.

- Hence

$$r_a c(a)v(a) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx$$

$$\Rightarrow \quad \int_a^b cu'v'\ dx = \int_a^b fv\ dx - r_a c(a)v(a).$$

- Code? Just add $-r_a c(a)$ to the corresponding entry of the load vector $\vec{b}$! You can find the corresponding entry by repeating the derivation of the matrix formulation from the above weak formulation.

Neumann/Robin boundary conditions

- Consider

$$-\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x)\ (a \le x \le b), u(a) = g_a, u'(b) = r_b.$$

- Recall

$$-c(b)u'(b)v(b) + c(a)u'(a)v(a) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx.$$

- Since the solution at $x = a$ is given by $u(a) = g_a$, then we can choose the test function $v(x)$ such that $v(a) = 0$.

- Hence

$$-r_b c(b)v(b) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx$$

$$\Rightarrow \quad \int_a^b cu'v'\ dx = \int_a^b fv\ dx + r_b c(b)v(b).$$

- Code? Just add $r_b c(b)$ to the corresponding entry of the load vector $\vec{b}$! You can find the corresponding entry by repeating the derivation of the matrix formulation from the above weak formulation.

Neumann/Robin boundary conditions

- Consider

$$\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x)\ (a \le x \le b), u'(a) = r_a, u'(b) = r_b.$$

- Recall

$$-c(b)u'(b)v(b) + c(a)u'(a)v(a) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx.$$

- Hence

$$-r_b c(b)v(b) + r_a c(a)v(a) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx.$$

- Is there anything wrong? The solution is not unique!
- If u is a solution, then $u + c$ is also a solution where c is a constant.

General 1D finite element method

- Example 3: Use the 1D linear and quadratic finite element methods to solve the following equation:

$$
\begin{aligned}
&-\frac{d}{dx}\left(e^x \frac{du(x)}{dx}\right) \\
&= -e^x[cos(x) - 2sin(x) - x\ cos(x) - x\ sin(x)]\ (0 \leq x \leq 1), \\
&u(0) = 0, u'(1) = cos(1) - sin(1).
\end{aligned}
$$

- The analytic solution of this problem is $u = x\ cos(x)$, which can be used to compute the error of the numerical solution.

Neumann/Robin boundary conditions

- Consider
$$-\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x)\ (a \le x \le b), u(a) = g_a, u'(b) + q_b u(b) = p_b.$$
- Recall
$$-c(b)u'(b)v(b) + c(a)u'(a)v(a) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx.$$
- Since the solution at $x = a$ is given by $u(a) = g_a$, then we can choose the test function $v(x)$ such that $v(a) = 0$.
- Hence
$$-[p_b - q_b u(b)]c(b)v(b) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx$$
$$\Rightarrow \quad q_b c(b)u(b)v(b) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx + p_b c(b)v(b).$$
- Code? Just add $p_b c(b)$ to the corresponding entry of the load vector $\vec{b}$ and $q_b c(b)$ to the corresponding entry of the stiffness matrix A! You can find the corresponding entries by repeating the derivation of the matrix formulation from the above weak formulation.

Neumann/Robin boundary conditions

- Consider

$$-\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x)\ (a \le x \le b), u'(a) = r_a, u'(b) + q_b u(b) = p_b.$$

- Recall

$$-c(b)u'(b)v(b) + c(a)u'(a)v(a) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx.$$

- Hence

$$-[p_b - q_b u(b)]c(b)v(b) + r_a c(a)v(a) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx$$

$$\Rightarrow \quad q_b c(b)u(b)v(b) + \int_a^b cu'v'\ dx = \int_a^b fv\ dx - r_a c(a)v(a) + p_b c(b)v(b).$$

- Code? Just add $-r_a c(a)$ and $p_b c(b)$ to the corresponding entries of the load vector $\vec{b}$ and $q_b c(b)$ to the corresponding entry of the stiffness matrix A! You can find the corresponding entries by repeating the derivation of the matrix formulation from the above weak formulation.

Neumann/Robin boundary conditions

- Consider

$$-\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x)\ (a \le x \le b), u'(a) + q_a u(a) = p_a, u(b) = g_b.$$

- Consider

$$-\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x)\ (a \le x \le b), u'(a) + q_a u(a) = p_a, u'(b) = r_b.$$

- Consider

$$-\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x)\ (a \le x \le b),$$
$$u'(a) + q_a u(a) = p_a, u'(b) + q_b u(b) = p_b.$$

General 1D finite element method

- Example 4: Use the 1D linear and quadratic finite element methods to solve the following equation:

$$-\frac{d}{dx}\left(e^x \frac{du(x)}{dx}\right)$$
$$= -e^x[cos(x) - 2sin(x) - x\ cos(x) - x\ sin(x)]\ (0 \leq x \leq 1),$$
$$u'(0) + u(0) = 1, u(1) = cos(1).$$

- The analytic solution of this problem is $u = x\ cos(x)$, which can be used to compute the error of the numerical solution.

More measurements for errors

Recall

Definition (L^2 space)

$$L^2(I) = \{v : I \to \mathbf{R} : \int_a^b v^2 \, dx < \infty\}$$

where $I = (a, b)$.

Definition (H^1 space)

$$H^1(I) = \{v \in L^2(I) : v' \in L^2(I)\}$$

where $I = (a, b)$.

More measurements for errors

Definition (L^∞ space)

$$L^\infty(I) = \{v : I \to \mathbf{R} : \sup_{x \in I} |u(x)| < \infty\}$$

where $I = (a, b)$.

More measurements for errors

- L^∞ norm: $\|u\|_\infty = \sup_{x\in I} |u(x)|$ for $u \in L^\infty(I)$.
- L^∞ norm error: $\|u - u_h\|_\infty = \sup_{x\in I} |u(x) - u_h(x)|$.
- L^2 norm: $\|u\|_0 = \sqrt{\int_I u^2 dx}$ for $u \in L^2(I)$.
- L^2 norm error: $\|u - u_h\|_0 = \sqrt{\int_I (u - u_h)^2 dx}$.
- H^1 semi-norm: $|u|_1 = \sqrt{\int_I u'^2 dx}$ for $u \in H^1(I)$.
- H^1 semi-norm error: $|u - u_h|_1 = \sqrt{\int_I (u' - u_h')^2 dx}$.
- H^1 norm: $\|u\|_1 = \sqrt{\int_I u^2 dx + \int_I u'^2 dx}$ for $u \in H^1(I)$.
- H^1 norm error: $\|u - u_h\|_1 = \sqrt{\int_I (u - u_h)^2 dx + \int_I (u' - u_h')^2 dx}$.

More measurements for errors

- By using $u_h = \sum_{j=1}^{N_b} u_j \phi_j$, the definition of T_b, and the definition of the local basis functions ψ_{nk}, we get

$$\begin{aligned}
\|u - u_h\|_\infty &= \sup_{x \in I} |u(x) - u_h(x)| \\
&= \max_{1 \le n \le N} \max_{x_n \le x \le x_{n+1}} |u(x) - u_h(x)| \\
&= \max_{1 \le n \le N} \max_{x_n \le x \le x_{n+1}} \left| u(x) - \sum_{j=1}^{N_b} u_j \phi_j \right| \\
&= \max_{1 \le n \le N} \max_{x_n \le x \le x_{n+1}} \left| u(x) - \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \psi_{nk}(x) \right| .
\end{aligned}$$

More measurements for errors

- Define

$$w_n(x) = \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \psi_{nk}(x).$$

Then

$$\|u - u_h\|_\infty = \max_{1 \le n \le N} \max_{x_n \le x \le x_{n+1}} |u(x) - w_n(x)| .$$

- $\max\limits_{x_n \le x \le x_{n+1}} |u(x) - w_n(x)|$ can be approximated by choosing the maximum values of $|u(x) - w_n(x)|$ on a group of chosen points in $[x_n, x_{n+1}]$, such as some Gauss quadrature nodes in this element. We denote the approximation by r_n.

More measurements for errors

Algorithm 7:

- Approximate the maximum absolute errors on all elements and then choose the largest one as the final approximation:

$FOR\ n = 1, \cdots, N$:

Compute $r_n \approx \max\limits_{x_n \le x \le x_{n+1}} |u(x) - w_n(x)|$;

END

$error = \max\limits_{1 \le n \le N} r_n.$

More measurements for errors

- By using $u_h = \sum_{j=1}^{N_b} u_j \phi_j$, the definition of T_b, and the definition of the local basis functions ψ_{nk}, we get

$$\begin{aligned}
\|u - u_h\|_0 &= \sqrt{\int_I (u - u_h)^2 dx} \\
&= \sqrt{\sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} (u - u_h)^2 dx} \\
&= \sqrt{\sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} \left(u - \sum_{j=1}^{N_b} u_j \phi_j \right)^2 dx} \\
&= \sqrt{\sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} \left(u - \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \psi_{nk} \right)^2 dx}.
\end{aligned}$$

More measurements for errors

- Define

$$w_n = \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \psi_{nk}.$$

Then

$$\|u - u_h\|_0 = \sqrt{\sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} (u - w_n)^2 dx}.$$

- Each integral $\int_{x_n}^{x_{n+1}} (u - w_n)^2 dx$ can be computed by numerical integration.

More measurements for errors

- By using $u_h = \sum_{j=1}^{N_b} u_j \phi_j$, the definition of T_b, and the definition of the local basis functions ψ_{nk}, we get

$$
\begin{aligned}
|u - u_h|_1 &= \sqrt{\int_I (u' - u_h')^2 dx} \\
&= \sqrt{\sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} (u' - u_h')^2 dx} \\
&= \sqrt{\sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} \left(u' - \sum_{j=1}^{N_b} u_j \phi_j' \right)^2 dx} \\
&= \sqrt{\sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} \left(u' - \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \psi_{nk}' \right)^2 dx}.
\end{aligned}
$$

More measurements for errors

- Define

$$w_n = \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \psi'_{nk}.$$

Then

$$|u - u_h|_1 = \sqrt{\sum_{n=1}^{N} \int_{x_n}^{x_{n+1}} (u' - w_n)^2 dx}.$$

- Each integral $\int_{x_n}^{x_{n+1}} (u' - w_n)^2 dx$ can be computed by numerical integration.

More measurements for errors

- Develop a subroutine for a more general formulation

$$\sqrt{\sum_{n=1}^{N}\int_{x_n}^{x_{n+1}}\left(u^{(s)}-\sum_{k=1}^{N_{lb}}u_{T_b(k,n)}\psi_{nk}^{(s)}\right)^2 dx}.$$

- That is,

$$\sqrt{\sum_{n=1}^{N}\int_{x_n}^{x_{n+1}}(u^{(s)}-w_{n,s})^2 dx},\quad w_{n,s}=\sum_{k=1}^{N_{lb}}u_{T_b(k,n)}\psi_{nk}^{(s)}.$$

- The L^2 norm error is equivalent to calling this subroutine with parameter $s=0$.
- The H^1 norm error is equivalent to calling this subroutine with parameter $s=1$.

More measurements for errors

Algorithm 8:

- Initialize the error $error = 0$; input the parameter s;
- Compute the integrals and add them into the total error:

  $FOR\ n = 1, \cdots, N$:

  $$error = error + \int_{x_n}^{x_{n+1}} \left(u^{(s)} - \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \psi_{nk}^{(s)} \right)^2 dx;$$

  END

  $error = \sqrt{error}$;

More measurements for errors

More numerical results for Example 1:

h	$\|u - u_h\|_\infty$	$\|u - u_h\|_0$	$\|u - u_h\|_1$
$1/4$	1.4041×10^{-2}	7.1969×10^{-3}	1.0528×10^{-1}
$1/8$	3.6803×10^{-3}	1.7951×10^{-3}	5.2731×10^{-2}
$1/16$	9.4048×10^{-4}	4.4854×10^{-4}	2.6376×10^{-2}
$1/32$	2.3760×10^{-4}	1.1212×10^{-4}	1.3189×10^{-2}
$1/64$	5.9704×10^{-5}	2.8029×10^{-5}	6.5949×10^{-3}
$1/128$	1.4964×10^{-5}	7.0072×10^{-6}	3.2975×10^{-3}

Table: L^∞ norm error, L^2 norm error and H^1 semi-norm error

- Any Observation?

More measurements for errors

- Second order convergence $O(h^2)$ in L^2/L^∞ norm since the error is reduced by $\frac{1}{4}$ when h is reduced by half.
- First order convergence $O(h)$ in H^1 semi-norm since the error is reduced by half when h is reduced by half.
- This matches the optimal approximation capability expected from piecewise linear functions.

More measurements for errors

More numerical results for Example 2:

h	$\|u - u_h\|_\infty$	$\|u - u_h\|_0$	$\|u - u_h\|_1$
$1/4$	3.3128×10^{-4}	2.1041×10^{-4}	5.4212×10^{-3}
$1/8$	3.9240×10^{-5}	2.6144×10^{-5}	1.3534×10^{-3}
$1/16$	4.7518×10^{-6}	3.2631×10^{-6}	3.3823×10^{-4}
$1/32$	5.8390×10^{-7}	4.0774×10^{-7}	8.4550×10^{-5}
$1/64$	7.2343×10^{-8}	5.0962×10^{-8}	2.1137×10^{-5}
$1/128$	9.0022×10^{-9}	6.3702×10^{-9}	5.2842×10^{-6}

Table: L^∞ norm error, L^2 norm error and H^1 semi-norm error

- Any Observation?

More measurements for errors

- Third order convergence $O(h^3)$ in L^2/L^∞ norm since the error is reduced by $\frac{1}{8}$ when h is reduced by half.
- Second order convergence $O(h^2)$ in H^1 semi-norm since the error is reduced by $\frac{1}{4}$ when h is reduced by half.
- This matches the optimal approximation capability expected from piecewise quadratic functions.

Outline

Advantages of the finite element method

- The framework of finite element methods are universal for all partial differential equations.
- If the original equations are symmetric positive definite, the linear systems arising from finite element methods are also symmetric positive definite, which is important for many fast matrix solvers.
- It is natural for finite element methods to deal with problem domains with curved boundary, interface or singularities once the mesh is properly constructed.
- It is natural for many finite element methods to keep the conservation law.
- The finite element methods provide piecewise functions defined on the whole problem domain as numerical solutions, not just the numerical solutions at mesh nodes.
- The finite element methods have mature frameworks for mathematical analysis.

Advantages of the finite element method

The framework of finite element methods are universal for all partial differential equations due to the following reasons:

- The weak formulations of all partial differential equations consist of integrals in similar formats. This unifies different equations into a universal formation.
- Due to the "local assembly" idea of the general implementation framework of finite elements, all the processes in finite element methods are completely based on the information matrices, including the construction of finite element spaces, the finite element discretization, the assembly of the matrices and vectors, and the treatment of the boundary conditions.
- Each analysis framework for finite element methods can be applied to a wide range of problems.

Higher dimensional equations

- Global indices of the elements and nodes: more complicated but similar.
- Mesh information matrices P, T: more complicated but similar.
- Mesh information matrix E: mesh information for the element edges/surfaces.
- Integrals: higher dimensional and more complicated but similar.
- Local Assembly: similar with the new indices.
- Boundary information matrix $boundarynodes$: more complicated but similar.
- Boundary information matrix $boundarysurfaces$: information for the element edges/surfaces on the problem domain boundary.

Different types of finite elements

- Linear finite elements, quadratic finite elements.....
- Bilinear finite elements, biquadratic finite elements.....
- Crouzeix-Raviart finite elements
- Mixed finite elements: Taylor-Hood elements, Raviart-Thomas elements.....
- Nonconforming finite elements
- More.....