01 Aug 2019

Chapter 3: Finite Elements for 2D second order elliptic equation

Xiaoming He
Missouri University of Science and Technology, hex@mst.edu

Mathematical Foundation of Finite Element Methods

Chapter 3: Finite Elements for 2D second order elliptic equation

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

Outline

Outline

Target problem

- Consider the 2D second order elliptic equation

$$-\nabla \cdot (c\nabla u) = f, \quad \text{in } \Omega$$
$$u = g, \quad \text{on } \partial\Omega.$$

  where Ω is a 2D domain, $f(x, y)$ and $c(x, y)$ are given functions on Ω, $g(x, y)$ is a given function on $\partial\Omega$ and $u(x, y)$ is the unknown function.

- The gradient of a 2D function u is defined by

$$\nabla u = (u_x, u_y).$$

- The divergence of a 2×1 vector $\overrightarrow{v}$ is defined by

$$\nabla \cdot \overrightarrow{v} = \frac{\partial v_1}{\partial x} + \frac{\partial v_2}{\partial y}.$$

Weak formulation

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

$$\begin{aligned} & -\nabla \cdot (c\nabla u) = f \quad \text{in } \Omega \\ \Rightarrow \quad & -\nabla \cdot (c\nabla u)v = fv \quad \text{in } \Omega \\ \Rightarrow \quad & -\int_\Omega \nabla \cdot (c\nabla u)v \; dxdy = \int_\Omega fv \; dxdy. \end{aligned}$$

- $u(x, y)$ is called a trail function and $v(x, y)$ is called a test function.

Weak formulation

- Second, using Green's formula (divergence theory, integration by parts in multi-dimension)

$$\int_\Omega \nabla \cdot (c\nabla u) v \; dxdy = \int_{\partial\Omega} (c\nabla u \cdot \vec{n}) \, v \; ds - \int_\Omega c\nabla u \cdot \nabla v \; dxdy,$$

we obtain

$$\int_\Omega c\nabla u \cdot \nabla v \; dxdy - \int_{\partial\Omega} (c\nabla u \cdot \vec{n}) \, v \; ds = \int_\Omega fv \; dxdy.$$

Weak formulation

- Since the solution on the domain boundary $\partial\Omega$ are given by $u(x,y) = g(x,y)$, then we can choose the test function $v(x,y)$ such that $v = 0$ on $\partial\Omega$.
- Hence

$$\int_\Omega c\nabla u \cdot \nabla v \; dxdy = \int_\Omega fv \; dxdy.$$

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

Sobolev spaces

Definition (Support)

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

Lemma (I)

A function compactly supported in Ω is zero on and near the boundary of Ω.

Sobolev spaces

Definition

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

- Recall integration by parts:

$$\int_\Omega \frac{\partial u}{\partial x} v \, dxdy = \int_{\partial\Omega} u v n_x \, ds - \int_\Omega u \frac{\partial v}{\partial x} \, dxdy.$$

- For $v \in C_0^\infty(\Omega)$, we have $v = 0$ on $\partial\Omega$. Then

$$\int_\Omega \frac{\partial u}{\partial x} v \, dxdy = - \int_\Omega u \frac{\partial v}{\partial x} \, dxdy.$$

Sobolev spaces

Definition (weak derivative with respect to x in 2D)

Suppose u is a real-valued function defined on a domain Ω and that u is integrable over every compact subset of Ω. If there exists another locally integrable function w defined on Ω such that

$$\int_\Omega wv \; dxdy = -\int_\Omega u\frac{\partial v}{\partial x} \; dxdy.$$

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

Sobolev spaces

Definition (general weak derivative in 2D)

Let $\alpha = (\alpha_1, \alpha_2)$. Suppose u is a real-valued function defined on a domain Ω and that u is integrable over every compact subset of Ω. If there exists another locally integrable function w defined on Ω such that

$$\int_\Omega wv \; dxdy = (-1)^{\alpha_1+\alpha_2} \int_\Omega u \frac{\partial^{\alpha_1+\alpha_2} v}{\partial x^{\alpha_1} \partial y^{\alpha_2}} \; dxdy.$$

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

Sobolev spaces

Lemma (II)

If u is differentiable, then u is weakly differentiable and its weak derivative of order $\alpha = (\alpha_1, \alpha_2)$ *is* $\frac{\partial^{\alpha_1+\alpha_2} u}{\partial x^{\alpha_1} \partial y^{\alpha_2}}$.

Remark

In the Sobolev spaces, which will be defined below, $\frac{\partial^{\alpha_1+\alpha_2} u}{\partial x^{\alpha_1} \partial y^{\alpha_2}}$ *is used to represent the weak derivative of order* $\alpha = (\alpha_1, \alpha_2)$.

Sobolev spaces

Definition (L^p space)

$$L^p(\Omega) = \{v : \Omega \to \mathbf{R} : \int_\Omega v^p \, dxdy < \infty\}.$$

Definition (L^2 space)

$$L^2(\Omega) = \{v : \Omega \to \mathbf{R} : \int_\Omega v^2 \, dxdy < \infty\}.$$

Definition (L^∞ space)

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

Sobolev spaces

Definition (H^m space)

$$H^m(\Omega) = \{v \in L^2(\Omega) : \frac{\partial^{\alpha_1+\alpha_2} v}{\partial x^{\alpha_1} \partial y^{\alpha_2}} \in L^2(\Omega),\ \forall \alpha_1 + \alpha_2 = 1, \cdots, m\}.$$

Definition (H^1 space)

$$H^1(\Omega) = \{v \in L^2(\Omega) : \frac{\partial^{\alpha_1+\alpha_2} v}{\partial x^{\alpha_1} \partial y^{\alpha_2}} \in L^2(\Omega),\ \forall \alpha_1 + \alpha_2 = 1\}.$$

Definition (H_0^1 space)

$$H_0^1(\Omega) = \{v \in H^1(\Omega) : v = 0 \text{ on } \partial\Omega\}.$$

Sobolev spaces

Definition (W_p^m space)

$$W_p^m(\Omega) = \{v : \Omega \to \mathbf{R} : \int_\Omega \left[\frac{\partial^{\alpha_1+\alpha_2} v}{\partial x^{\alpha_1} \partial y^{\alpha_2}}\right]^p dxdy < \infty, \forall \alpha_1 + \alpha_2 = 0, \cdots, m\}.$$

Remark

- $L^p(\Omega) = W_p^0(\Omega)$;
- $L^2(\Omega) = W_2^0(\Omega)$;
- $H^m(\Omega) = W_2^m(\Omega)$;
- $H^1(\Omega) = W_2^1(\Omega)$.

Weak formulation

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

$$\int_\Omega c\nabla u \cdot \nabla v \ dxdy = \int_\Omega fv \ dxdy.$$

  for any $v \in H_0^1(\Omega)$.

- Let $a(u, v) = \int_\Omega c\nabla u \cdot \nabla v dxdy$ and $(f, v) = \int_\Omega fvdxdy$.

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

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

  for any $v \in H_0^1(\Omega)$.

Galerkin formulation

- Assume there is a finite dimensional subspace $U_h \subset H^1(\Omega)$.
- Then the Galerkin formulation is to find $u_h \in U_h$ such that

$$\begin{aligned} & a(u_h, v_h) = (f, v_h) \\ \Leftrightarrow \quad & \int_\Omega c\nabla u_h \cdot \nabla v_h \, dxdy = \int_\Omega f v_h \, dxdy \end{aligned}$$

  for any $v_h \in U_h$.
- Basic idea of Galerkin formulation: use finite dimensional space to approximate infinite dimensional space.
- Here $U_h = span\{\phi_j\}_{j=1}^{N_b}$ is chosen to be a finite element space where $\{\phi_j\}_{j=1}^{N_b}$ are the global finite element basis functions.

Outline

Discretization formulation

Recall the following definitions from Chapter 2:

- N: number of mesh elements.
- N_m: number of mesh nodes.
- E_n $(n = 1, \cdots, N)$: mesh elements.
- Z_k $(k = 1, \cdots, N_m)$: mesh nodes.
- N_l: number of local mesh nodes in a mesh element.
- P:information matrix consisting of the coordinates of all mesh nodes.
- T: information matrix consisting of the global node indices of the mesh nodes of all the mesh elements.

Discretization formulation

- We only consider the nodal basis functions (Lagrange type) in this course.
- N_{lb}: number of local finite element nodes (=number of local finite element basis functions) in a mesh element.
- N_b: number of the finite element nodes (= the number of unknowns = the total number of the finite element basis functions).
- X_j $(j = 1, \cdots, N_b)$: finite element nodes.
- P_b: information matrix consisting of the coordinates of all finite element nodes.
- T_b: information matrix consisting of the global node indices of the finite element nodes of all the mesh elements.

Discretization formulation

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

$$\begin{aligned} & a(u_h, v_h) = (f, v_h) \\ \Leftrightarrow \quad & \int_\Omega c \nabla u_h \cdot \nabla v_h \, dxdy = \int_\Omega f v_h \, dxdy \end{aligned}$$

  for any $v_h \in U_h$.

- Here $U_h = span\{\phi_j\}_{j=1}^{N_b}$ is chosen to be a finite element space where $\{\phi_j\}_{j=1}^{N_b}$ are the global finite element basis functions defined in Chapter 2.
- Since $u_h \in U_h = span\{\phi_j\}_{j=1}^{N_b}$, then

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

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

Discretization formulation

- 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

$$u_h(X_k) = \sum_{j=1}^{N_b} u_j \phi_j(A_k) = u_k.$$

- Hence the coefficient u_j is actually the numerical solution at the node X_j $(j = 1, \cdots, N_b)$.

Discretization formulation

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

$$\int_\Omega c \nabla \left(\sum_{j=1}^{N_b} u_j \phi_j \right) \cdot \nabla \phi_i \; dxdy = \int_\Omega f \phi_i \; dxdy,$$

$$\Rightarrow \quad \sum_{j=1}^{N_b} u_j \left[\int_\Omega c \nabla \phi_j \cdot \nabla \phi_i \; dxdy \right] = \int_\Omega f \phi_i \; dxdy, \;\; i = 1, \cdots, N_b.$$

Matrix formulation

- Define the stiffness matrix

$$A = [a_{ij}]_{i,j=1}^{N_b} = \left[\int_\Omega c\nabla\phi_j \cdot \nabla\phi_i \ dxdy\right]_{i,j=1}^{N_b}.$$

- Define the load vector

$$\vec{b} = [b_i]_{i=1}^{N_b} = \left[\int_\Omega f\phi_i \ dxdy\right]_{i=1}^{N_b}.$$

- Define the unknown vector

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

- Then we obtain the linear algebraic system

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

Assembly of the stiffness matrix

- Once $\vec{X}$ is obtained, the finite element solution u_h and the numerical solutions at all the mesh nodes are obtained.
- From the definition of ϕ_j $(j = 1, \cdots, N_b)$, 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

$$a_{ij} = \int_{\Omega} c \nabla \phi_j \cdot \nabla \phi_i \; dxdy = \sum_{n=1}^{N} \int_{E_n} c \nabla \phi_j \cdot \nabla \phi_i \; dxdy.$$

- It is easy to see that most of $\int_{E_n} c \nabla \phi_j \cdot \nabla \phi_i \; dxdy$ will be 0.
- So we only need to use numerical integration to compute those nonzero integrals.

Assembly of the stiffness matrix

General local assembly idea for A:

- Loop over all the 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.

Assembly of the stiffness matrix

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

- On the n^{th} element E_n, we get non-zero local integrals only when the trial and test basis functions are corresponding to the finite element nodes of this element.
- Let $p_s = T_b(s,n)$ $(s = 1, \cdots, N_{lb})$.
- Then we only consider the trial and test basis functions to be ϕ_{p_s} $(s = 1, \cdots, N_{lb})$.
- There are only N_{lb}^2 non-zero local integrals on E_n with the global basis functions ϕ_{p_s} $(s = 1, \cdots, N_{lb})$:

$$\int_{E_n} c \nabla \phi_j \cdot \nabla \phi_i \; dxdy \;\; (i, j = p_1, \cdots, p_{N_{lb}}).$$

- In fact, we have

$$\psi_{ns} = \phi_{p_s}|_{E_n} \;\; (s = 1, \cdots, N_{lb}).$$

Assembly of the stiffness matrix

- That is, instead of the original non-zero local integrals with the global basis functions ϕ_{p_s} $(s = 1, \cdots, N_{lb})$, we will compute the following non-zero local integrals with the local basis functions ψ_{ns} $(s = 1, \cdots, N_{lb})$:

$$\int_{E_n} c \nabla \psi_{n\alpha} \cdot \nabla \psi_{n\beta} \, dxdy \ (\alpha, \beta = 1, \cdots, N_{lb}).$$

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

Assembly of the stiffness matrix

Assemble the non-zero local integrals into A:

- When the trial function is ϕ_i and the test function is ϕ_j, the corresponding non-zero local integrals should be assembled to a_{ij}.
- 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.

Assembly of the stiffness matrix

- Question: Since we compute

$$\int_{E_n} c\nabla\psi_{n\alpha}\cdot\nabla\psi_{n\beta}\, dxdy \ (\alpha,\beta=1,\cdots,N_{lb})$$

instead of

$$\int_{E_n} c\nabla\phi_j\cdot\nabla\phi_i\, dxdy \ (i,j=p_1,\cdots,p_{N_{lb}}),$$

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

- Information matrix T_b!

Assembly of the stiffness matrix

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

$$\int_{E_n} c\nabla\psi_{n\alpha} \cdot \nabla\psi_{n\beta}\, dxdy \ (\alpha, \beta = 1, \cdots, N_{lb})$$

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

Assembly of the stiffness matrix

Algorithm I-1:

- 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 $r = \int_{E_n} c \nabla \psi_{n\alpha} \cdot \nabla \psi_{n\beta} \, dxdy$;
            Add $r$ to $A(T_b(\beta, n), T_b(\alpha, n))$.
        END
    END
END
```

Assembly of the stiffness matrix

Algorithm I-2:

- 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 $S(\beta, \alpha) = \int_{E_n} c \nabla \psi_{n\alpha} \cdot \nabla \psi_{n\beta}\, dxdy$;
        $END$
    $END$
    $A(T_b(:, n), T_b(:, n)) = A(T_b(:, n), T_b(:, n)) + S$;
$END$
```

Assembly of the stiffness matrix

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 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 finite element information matrices P_b and T_b for the trial and test functions respectively, 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);
- the type of the basis function for the trial and test functions respectively;

Assembly of the stiffness matrix

- Note that

$$\int_{E_n} c\nabla\psi_{n\alpha}\cdot\nabla\psi_{n\beta}\, dxdy = \int_{E_n} c\frac{\partial\psi_{n\alpha}}{\partial x}\frac{\partial\psi_{n\beta}}{\partial x}\, dxdy + \int_{E_n} c\frac{\partial\psi_{n\alpha}}{\partial y}\frac{\partial\psi_{n\beta}}{\partial y}\, dxdy.$$

- Hence we can consider to develop an algorithm to assemble the matrix arising from a more general integral

$$\int_{E_n} c\frac{\partial^{r+s}\psi_{n\alpha}}{\partial x^r\partial y^s}\frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p\partial y^q}\, dxdy.$$

with parameters r, s, p, and q.

Assembly of the stiffness matrix

Algorithm I-3:

- 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 $r = \int_{E_n} c \frac{\partial^{r+s}\psi_{n\alpha}}{\partial x^r \partial y^s} \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q} \, dxdy$;
            Add $r$ to $A(T_b(\beta, n), T_b(\alpha, n))$.
        END
    END
END
```

Assembly of the stiffness matrix

Algorithm I-4:

- 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 $S(\beta, \alpha) = \int_{E_n} c \frac{\partial^{r+s}\psi_{n\alpha}}{\partial x^r \partial y^s} \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q}\ dxdy$;
        $END$
    $END$
    $A(T_b(:, n), T_b(:, n)) = A(T_b(:, n), T_b(:, n)) + S$;
$END$
```

Assembly of the stiffness matrix

- First, we call Algorithm I-3 with $r = p = 1$ and $s = q = 0$ to obtain $A1$.
- Second, we call Algorithm I-3 with $r = p = 0$ and $s = q = 1$ to obtain $A2$.
- Then the stiffness matrix $A = A1 + A2$.
- That is, Algorithm I-1 is equivalent to calling Algorithm I-3 twice with two different groups of parameters $(r = p = 1, s = q = 0$ and $r = p = 0, s = q = 1)$ and then adding the two resulted matrices together.
- Algorithm I-2 and Algorithm I-4 have a similar relationship.

Assembly of the load vector

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

$$b_i = \int_\Omega f \phi_i \, dxdy \quad = \quad \sum_{n=1}^{N} \int_{E_n} f \phi_i \, dxdy, \;\; i = 1, \cdots, N_b.$$

- 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 $\vec{b}$.

Assembly of the load vector

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

- On the n^{th} element E_n, we get non-zero local integrals only when the test basis functions are corresponding to the finite element nodes of the element.
- Let $p_s = T_b(s, n)$ $(s = 1, \cdots, N_{lb})$.
- Then we only consider the test basis functions to be ϕ_{p_s} $(s = 1, \cdots, N_{lb})$.
- There are only N_{lb} non-zero local integrals on E_n with the global basis functions ϕ_{p_s} $(s = 1, \cdots, N_{lb})$:

$$\int_{E_n} f \phi_i \, dxdy \ (i = p_1, \cdots, p_{N_{lb}}).$$

- In fact, we have

$$\psi_{ns} = \phi_{p_s}|_{E_n} \ (s = 1, \cdots, N_{lb}).$$

Assembly of the load vector

- That is, instead of the original non-zero local integrals with the global basis functions ϕ_{p_s} $(s = 1, \cdots, N_{lb})$, we will compute the following non-zero local integrals with the local basis functions ψ_{ns} $(s = 1, \cdots, N_{lb})$:

$$\int_{E_n} f \psi_{n\beta} \, dxdy \ (\beta = 1, \cdots, N_{lb}).$$

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

Assembly of the load vector

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

- When the test function is ϕ_i, the corresponding non-zero local integrals should be assembled to b_i.
- Therefore, if we find the global node indices of the test basis functions, we can easily locate where to assemble a non-zero local integral.
- Question: Since we compute

$$\int_{E_n} f\psi_{n\beta}\, dxdy \ (\beta = 1, \cdots, N_{lb})$$

  instead of

$$\int_{E_n} f\phi_i\, dxdy \ (i = p_1, \cdots, p_{N_{lb}}),$$

  how do we obtain the corresponding global node indices of the local test basis functions $\psi_{n\beta}$ $(\beta = 1, \cdots, N_{lb})$?
- Information matrix T_b!

Assembly of the load vector

- Recall that $T_b(\beta, n)$ give the global node indices of the local test basis functions $\psi_{n\beta}$ $(\beta = 1, \cdots, N_{lb})$.
- That is, for $n = 1, \cdots, N$,

$$\int_{E_n} f\psi_{n\beta}\, dxdy \ (\beta = 1, \cdots, N_{lb})$$

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

Assembly of the load vector

Algorithm II-1:

- Initialize the matrix: $b = sparse(N_b, 1)$;
- Compute the integrals and assemble them into b:

```
FOR $n = 1, \cdots, N$:
    FOR $\beta = 1, \cdots, N_{lb}$:
        Compute $r = \int_{E_n} f \psi_{n\beta}\ dxdy$;
        $b(T_b(\beta, n), 1) = b(T_b(\beta, n), 1) + r$;
    END
END
```

Assembly of the load vector

Algorithm II-2:

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

```
$FOR\ n = 1, \cdots, N$:
    $FOR\ \beta = 1, \cdots, N_{lb}$:
        Compute $d(\beta, 1) = \int_{E_n} f \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q}\, dxdy$;
    $END$
    $b(T_b(:, n), 1) = b(T_b(:, n), 1) + d$;
$END$
```

Assembly of the load vector

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 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);
- the type of the basis function for the test functions.

Assembly of the load vector

- We can also consider to develop an algorithm to assemble the vector arising from

$$\int_{E_n} f \frac{\partial^{p+q} \psi_{n\beta}}{\partial x^p \partial y^q} \, dxdy.$$

Assembly of the load vector

Algorithm II-3:

- Initialize the matrix: $b = sparse(N_b, 1)$;
- Compute the integrals and assemble them into b:

```
FOR $n = 1, \cdots, N$:
    FOR $\beta = 1, \cdots, N_{lb}$:
        Compute $r = \int_{E_n} f \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q}\, dxdy$;
        $b(T_b(\beta, n), 1) = b(T_b(\beta, n), 1) + r$;
    END
END
```

Assembly of the load vector

Algorithm II-4:

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

```
$FOR\ n = 1, \cdots, N$:
    $FOR\ \beta = 1, \cdots, N_{lb}$:
        Compute $d(\beta, 1) = \int_{E_n} f \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q}\ dxdy$;
    $END$
    $b(T_b(:, n), 1) = b(T_b(:, n), 1) + d$;
$END$
```

Assembly of the load vector

- We call Algorithm I-3 with $p = q = 0$ to obtain b.
- That is, Algorithm II-3 is equivalent to Algorithm II-1 with $p = q = 0$.
- Algorithm II-2 and Algorithm II-4 have a similar relationship.

Outline

Dirichlet boundary condition

- Basically, the Dirichlet boundary condition $u = g$ give the solutions at all boundary finite element nodes.
- Since the coefficient u_j in the finite element solution $u_h = \sum_{j=1}^{N_b} u_j \phi_j$ is actually the numerical solution at the finite element node X_j ($j = 1, \cdots, N_b$), we actually know those u_j which are corresponding to the boundary finite element nodes.
- Recall that boundarynodes(2,:) store the global node indices of all boundary finite element nodes.
- If $m \in \ boundarynodes(2, :)$, then the m^{th} equation is called a boundary node equation.
- Set *nbn* to be the number of boundary nodes;

Dirichlet boundary condition

- One way to impose the Dirichlet boundary condition is to replace the boundary node equations in the linear system by the following equations

$$u_m = g(X_m).$$

for all $m \in \ boundarynodes(2,:)$.

Dirichlet boundary condition

Algorithm III:

- 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
```

Outline

Universal framework of the finite element method

- Generate the mesh information: matrices P and T;
- Assemble the matrices and vectors: local assembly based on P and T only;
- Deal with the boundary conditions: boundary information matrix and local assembly;
- Solve linear systems: numerical linear algebra (Math 6601: Numerical Analysis).

Algorithm

- Generate the mesh information matrices P and T.
- Assemble the stiffness matrix A by using Algorithm I. (We will choose Algorithm I-3 in class)
- Assemble the load vector $\vec{b}$ by using Algorithm II. (We will choose Algorithm II-3 in class)
- Deal with the Drichlet boundary condition by using Algorithm III.
- Solve $A\vec{X} = \vec{b}$ for $\vec{X}$ by using a direct or iterative method.

Algorithm

Recall Algorithm I-3:

- 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 $r = \int_{E_n} c \frac{\partial^{r+s}\psi_{n\alpha}}{\partial x^r \partial y^s} \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q}\, dxdy$;
            Add $r$ to $A(T_b(\beta, n), T_b(\alpha, n))$.
        END
    END
END
```

Algorithm

Recall

- First, we call Algorithm I-3 with $r = p = 1$ and $s = q = 0$ to obtain $A1$.
- Second, we call Algorithm I-3 with $r = p = 0$ and $s = q = 1$ to obtain $A2$.
- Then the stiffness matrix $A = A1 + A2$.

Algorithm

Recall Algorithm II-3:

- Initialize the matrix: $b = sparse(N_b, 1)$;
- Compute the integrals and assemble them into b:

```
FOR $n = 1, \cdots, N$:
    FOR $\beta = 1, \cdots, N_{lb}$:
        Compute $r = \int_{E_n} f \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q}\, dxdy$;
        $b(T_b(\beta, n), 1) = b(T_b(\beta, n), 1) + r$;
    END
END
```

- Recall: We call Algorithm I-3 with $p = q = 0$ to obtain b.

Algorithm

Recall Algorithm III:

- Deal with the Dirichlet boundary conditions:

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

Measurements for errors

Recall

Definition (L^2 space)

$$L^2(\Omega) = \{v : \Omega \to \mathbf{R} : \int_\Omega v^2 \, dxdy < \infty\}.$$

Definition (H^1 space)

$$H^1(\Omega) = \{v \in L^2(\Omega) : \frac{\partial^{\alpha_1+\alpha_2} v}{\partial x^{\alpha_1} \partial y^{\alpha_2}} \in L^2(\Omega), \ \forall \alpha_1 + \alpha_2 = 1\}.$$

Definition (L^∞ space)

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

Measurements for errors

- L^∞ norm: $\|u\|_\infty = \sup\limits_{(x,y)\in\Omega} |u(x,y)|$ for $u \in L^\infty(\Omega)$.
- L^∞ norm error: $\|u-u_h\|_\infty = \sup\limits_{(x,y)\in\Omega} |u(x,y)-u_h(x,y)|$.
- L^2 norm: $\|u\|_0 = \sqrt{\int_\Omega u^2 dxdy}$ for $u \in L^2(\Omega)$.
- L^2 norm error: $\|u-u_h\|_0 = \sqrt{\int_\Omega (u-u_h)^2 dxdy}$.
- H^1 semi-norm: $|u|_1 = \sqrt{\int_\Omega \left(\frac{\partial u}{\partial x}\right)^2 dxdy + \int_\Omega \left(\frac{\partial u}{\partial y}\right)^2 dxdy}$ for $u \in H^1(\Omega)$.
- H^1 semi-norm error: $|u-u_h|_1 = \sqrt{\int_\Omega \left(\frac{\partial(u-u_h)}{\partial x}\right)^2 dxdy + \int_\Omega \left(\frac{\partial(u-u_h)}{\partial y}\right)^2 dxdy}$.

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

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

Measurements for errors

- Define
$$w_n(x,y)=\sum_{k=1}^{N_{lb}} u_{T_b(k,n)}\psi_{nk}(x,y).$$
Then
$$\|u-u_h\|_\infty \quad = \quad \max_{1\le n\le N}\max_{(x,y)\in E_n}|u(x,y)-w_n(x,y)|\,.$$

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

Measurements for errors

Algorithm IV:

- Initialize the error $error = 0$;
- Approximate the maximum absolute errors on all elements and then choose the largest one as the final approximation:

```
FOR $n = 1, \cdots, N$:
    Compute $r_n \approx \max_{(x,y) \in E_n} |u(x,y) - w_n(x,y)|$;
    IF $r_n > error$, THEN
        $error = r_n$;
    END
END
```

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

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

Measurements for errors

- Define

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

Then

$$\|u - u_h\|_0 = \sqrt{\sum_{n=1}^{N} \int_{E_n} (u - w_n)^2 dxdy}.$$

- Each integral $\int_{E_n} (u - w_n)^2 dxdy$ can be computed by numerical integration.

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

$$\begin{aligned}
|u-u_h|_{1,x} &= \sqrt{\int_\Omega \left(\frac{\partial(u-u_h)}{\partial x}\right)^2} \\
&= \sqrt{\sum_{n=1}^{N} \int_{E_n} \left(\frac{\partial(u-u_h)}{\partial x}\right)^2 dxdy} \\
&= \sqrt{\sum_{n=1}^{N} \int_{E_n} \left(\frac{\partial u}{\partial x} - \sum_{j=1}^{N_b} u_j \frac{\partial \phi_j}{\partial x}\right)^2 dxdy} \\
&= \sqrt{\sum_{n=1}^{N} \int_{E_n} \left(\frac{\partial u}{\partial x} - \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \frac{\partial \psi_{nk}}{\partial x}\right)^2 dxdy}.
\end{aligned}$$

Measurements for errors

- Similarly,

$$
\begin{aligned}
|u-u_h|_{1,y} &= \sqrt{\int_\Omega \left(\frac{\partial(u-u_h)}{\partial y}\right)^2 dxdy} \\
&= \sqrt{\sum_{n=1}^{N} \int_{E_n} \left(\frac{\partial(u-u_h)}{\partial y}\right)^2 dxdy} \\
&= \sqrt{\sum_{n=1}^{N} \int_{E_n} \left(\frac{\partial u}{\partial y} - \sum_{j=1}^{N_b} u_j \frac{\partial \phi_j}{\partial y}\right)^2 dxdy} \\
&= \sqrt{\left(\frac{\partial u}{\partial y} - \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \frac{\partial \psi_{nk}}{\partial y}\right)^2 dxdy}.
\end{aligned}
$$

Measurements for errors

- Then

$$
\begin{aligned}
& |u-u_h|_1^2 \\
= & |u-u_h|_{1,x}^2 + |u-u_h|_{1,y}^2 \\
= & \sum_{n=1}^{N} \int_{E_n} \left(\frac{\partial u}{\partial x} - \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \frac{\partial \psi_{nk}}{\partial x} \right)^2 dxdy \\
& + \sum_{n=1}^{N} \int_{E_n} \left(\frac{\partial u}{\partial y} - \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \frac{\partial \psi_{nk}}{\partial y} \right)^2 dxdy.
\end{aligned}
$$

Measurements for errors

- Define

$$\begin{aligned} w_{n1} &= \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \frac{\partial \psi_{nk}}{\partial x}, \\ w_{n2} &= \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \frac{\partial \psi_{nk}}{\partial y}. \end{aligned}$$

  Then

$$\begin{aligned} & |u - u_h|_1 \\ = & \sqrt{\sum_{n=1}^{N} \int_{E_n} \left(\frac{\partial u}{\partial x} - w_{n1} \right)^2 dxdy + \sum_{n=1}^{N} \int_{E_n} \left(\frac{\partial u}{\partial y} - w_{n2} \right)^2 dxdy}. \end{aligned}$$

- Each integral $\int_{E_n} \left(\frac{\partial u}{\partial x} - w_{n1} \right)^2 dxdy$ or $\int_{E_n} \left(\frac{\partial u}{\partial y} - w_{n2} \right)^2 dxdy$ can be computed by numerical integration.

Measurements for errors

- Develop a subroutine for a more general formulation

$$\sqrt{\sum_{n=1}^{N}\int_{E_n}\left(\frac{\partial^{\alpha_1+\alpha_2}u}{\partial x^{\alpha_1}\partial y^{\alpha_2}}-\sum_{k=1}^{N_{lb}}u_{T_b(k,n)}\frac{\partial^{\alpha_1+\alpha_2}\psi_{nk}}{\partial x^{\alpha_1}\partial y^{\alpha_2}}\right)^2 dxdy}.$$

- $\|u-u_h\|_0$ is equivalent to calling this subroutine with $\alpha_1=0$ and $\alpha_2=0$.
- $|u-u_h|_{1,x}$ is equivalent to calling this subroutine with $\alpha_1=1$ and $\alpha_2=0$.
- $|u-u_h|_{1,y}$ is equivalent to calling this subroutine with $\alpha_1=0$ and $\alpha_2=1$.

Measurements for errors

Algorithm V:

- Initialize the error $error = 0$; input the parameters α_1 and α_2;
- Compute the integrals and add them into the total error:

```
$FOR\ n = 1, \cdots, N$:
    $$error = error + \int_{E_n} \left( \frac{\partial^{\alpha_1+\alpha_2} u}{\partial x^{\alpha_1} \partial y^{\alpha_2}} - \sum_{k=1}^{N_{lb}} u_{T_b(k,n)} \frac{\partial^{\alpha_1+\alpha_2} \psi_{nk}}{\partial x^{\alpha_1} \partial y^{\alpha_2}} \right)^2 dxdy;$$
$END$
$error = \sqrt{error};$
```

Numerical example

- Example 1: Use the finite element method to solve the following equation on the domain $\Omega = [-1,1] \times [-1,1]$:

$$\begin{aligned} -\nabla \cdot (\nabla u) &= -y(1-y)(1-x-\frac{x^2}{2})e^{x+y} \\ &\quad -x(1-\frac{x}{2})(-3y-y^2)e^{x+y}, \\ u &= -1.5y(1-y)e^{-1+y} \text{ on } x=-1, \\ u &= 0.5y(1-y)e^{1+y} \text{ on } x=1, \\ u &= -2x(1-\frac{x}{2})e^{x-1} \text{ on } y=-1, \\ u &= 0 \text{ on } y=1. \end{aligned}$$

- The analytic solution of this problem is $u = xy(1-\frac{x}{2})(1-y)e^{x+y}$, which can be used to compute the error of the numerical solution.

Numerical example

- Let's code for the linear and quadratic finite element method of the 2D second order elliptic equation together!
- Open your Matlab!

Numerical example

h	$\|u - u_h\|_\infty$	$\|u - u_h\|_0$	$\|u - u_h\|_1$
$1/8$	2.3620×10^{-2}	6.8300×10^{-3}	1.8774×10^{-1}
$1/16$	6.3421×10^{-3}	1.7189×10^{-3}	9.4167×10^{-2}
$1/32$	1.6430×10^{-3}	4.3049×10^{-4}	4.7121×10^{-2}
$1/64$	4.1810×10^{-4}	1.0767×10^{-4}	2.3565×10^{-2}
$1/128$	1.0546×10^{-4}	2.6922×10^{-5}	1.1783×10^{-2}

Table : The numerical errors for linear finite element.

- Any Observation?
- Second order convergence $O(h^2)$ in L^2/L^∞ norm and first order convergence $O(h)$ in H^1 semi-norm, which match the optimal approximation capability expected from piecewise linear functions.

Numerical example

h	$\|u-u_h\|_\infty$	$\|u-u_h\|_0$	$\|u-u_h\|_1$
$1/8$	3.3678×10^{-4}	1.1705×10^{-4}	8.9192×10^{-3}
$1/16$	4.4273×10^{-5}	1.4637×10^{-5}	2.2414×10^{-3}
$1/32$	5.6752×10^{-6}	1.8289×10^{-6}	5.6131×10^{-4}
$1/64$	7.1839×10^{-7}	2.2853×10^{-7}	1.4042×10^{-4}
$1/128$	9.0366×10^{-8}	2.8560×10^{-8}	3.5114×10^{-5}

Table : The numerical errors for quadratic finite element.

- Any Observation?
- Third order convergence $O(h^3)$ in L^2/L^∞ norm and second order convergence $O(h^2)$ in H^1 semi-norm, which match the optimal approximation capability expected from piecewise quadratic functions.

Outline

Neumann boundary conditions

- Consider

$$-\nabla \cdot (c\nabla u) = f \quad \text{in } \Omega, \quad \nabla u \cdot \vec{n} = p \quad \text{on } \partial\Omega.$$

- Recall

$$\int_\Omega c\nabla u \cdot \nabla v \; dxdy - \int_{\partial\Omega} (c\nabla u \cdot \vec{n})\, v \; ds = \int_\Omega fv \; dxdy.$$

- Hence

$$\int_\Omega c\nabla u \cdot \nabla v \; dxdy = \int_\Omega fv \; dxdy + \int_{\partial\Omega} cpv \; ds.$$

- 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.

Neumann boundary condition

- Consider

$$\begin{aligned}
&-\nabla \cdot (c\nabla u) = f \quad \text{in } \Omega, \\
&\nabla u \cdot \vec{n} = p \quad \text{on } \Gamma_1 \subset \partial\Omega, \\
&u = g \quad \text{on } \partial\Omega/\Gamma_1.
\end{aligned}$$

- Recall

$$\int_\Omega c\nabla u \cdot \nabla v \; dxdy - \int_{\partial\Omega} (c\nabla u \cdot \vec{n})\, v \; ds = \int_\Omega fv \; dxdy.$$

- Since the solution on $\partial\Omega/\Gamma_1$ is given by $u = g$, then we can choose the test function $v(x)$ such that $v = 0$ on $\partial\Omega/\Gamma_1$.

Neumann boundary condition

- Since

$$\begin{aligned}\int_{\partial\Omega}(c\nabla u\cdot\vec{n})\,v\;ds &= \int_{\Gamma_1}(c\nabla u\cdot\vec{n})\,v\;ds+\int_{\partial\Omega/\Gamma_1}(c\nabla u\cdot\vec{n})\,v\;ds\\ &= \int_{\Gamma_1} cpv\;ds,\end{aligned}$$

  then

$$\int_{\Omega} c\nabla u\cdot\nabla v\;dxdy-\int_{\Gamma_1} cpv\;ds=\int_{\Omega} fv\;dxdy.$$

- Hence the weak formulation is

$$\int_{\Omega} c\nabla u\cdot\nabla v\;dxdy=\int_{\Omega} fv\;dxdy+\int_{\Gamma_1} cpv\;ds.$$

Neumann boundary condition

- Then the Galerkin formulation is to find $u_h \in U_h$ such that

$$\int_\Omega c\nabla u_h \cdot \nabla v_h \, dxdy = \int_\Omega f v_h \, dxdy + \int_{\Gamma_1} c p v_h \, ds$$

for any $v_h \in U_h$.

- Recall: Since $u_h \in U_h = span\{\phi_j\}_{j=1}^{N_b}$, then

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

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

- Recall: Choose $v_h = \phi_i$ $(i = 1, \cdots, N_b)$.

Neumann boundary condition

- Then for $i = 1, \cdots, N_b$, the finite element formulation gives

$$\int_\Omega c\nabla(\sum_{j=1}^{N_b} u_j\phi_j) \cdot \nabla\phi_i \ dxdy = \int_\Omega f\phi_i \ dxdy + \int_{\Gamma_1} cp\phi_i \ ds,$$

$$\Rightarrow \sum_{j=1}^{N_b} u_j \left[\int_\Omega c\nabla\phi_j \cdot \nabla\phi_i \ dxdy\right] = \int_\Omega f\phi_i \ dxdy + \int_{\Gamma_1} cp\phi_i \ ds.$$

Neumann boundary condition

Recall

- Define the stiffness matrix

$$A = [a_{ij}]_{i,j=1}^{N_b} = \left[\int_\Omega c \nabla \phi_j \cdot \nabla \phi_i \; dxdy \right]_{i,j=1}^{N_b}.$$

- Define the load vector

$$\vec{b} = [b_i]_{i=1}^{N_b} = \left[\int_\Omega f \phi_i \; dxdy \right]_{i=1}^{N_b}.$$

- Define the unknown vector

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

Neumann boundary condition

- Define the additional vector from the Neumann boundary condition

$$\vec{v} = [v_i]_{i=1}^{N_b} = \left[\int_{\Gamma_1} cp\phi_i \, ds \right]_{i=1}^{N_b}.$$

- Define the new vector $\widetilde{\vec{b}} = \vec{b} + \vec{v}$.
- Then we obtain the linear algebraic system

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

- Code?
- Add one more subroutine for $\vec{v}$ to the existing code!

Neumann boundary condition

Recall

- Matrix *boundaryedges*:
- $boundaryedges(1, k)$ is the type of the k^{th} boundary edge e_k: Dirichlet (-1), Neumann (-2), Robin (-3)......
- $boundaryedges(2, k)$ is the index of the element which contains the k^{th} boundary edge e_k.
- Each boundary edge has two end nodes. We index them as the first and the second counterclock wise along the boundary.
- $boundaryedges(3, k)$ is the global node index of the first end node of the k^{th} boundary boundary edge e_k.
- $boundaryedges(4, k)$ is the global node index of the second end node of the k^{th} boundary boundary edge e_k.
- Set $nbe = size(boundaryedges, 2)$ to be the number of boundary edges;

Neumann boundary condition

- The idea for the assembly of the vector $\vec{v}$ is similar to that of the load vector. We have

$$v_i = \int_{\Gamma_1} cp\phi_i \, ds \quad = \sum_{\substack{e_k \subset \Gamma_1 \\ 1 \leq k \leq nbe}} \int_{e_k} cp\phi_i \, ds, \ i = 1, \cdots, N_b.$$

- Loop over all the boundary edges;
- Compute all non-zero local integrals on each Neumann boundary edge for the vector $\vec{v}$;
- Assemble these non-zero local integrals into the corresponding entries of the vector $\vec{v}$.

Neumann boundary condition

Compute all non-zero local integrals on each Neumann boundary edge for $\vec{v}$:

- The index of the element which contains the k^{th} boundary edge e_k is $n_k = boundaryedges(2, k)$. Then on e_k, we get non-zero local integrals only when the test basis functions are corresponding to the finite element nodes of the n_k^{th} element E_{n_k}.
- Let $p_s = T_b(s, n_k)$ $(s = 1, \cdots, N_{lb})$.
- Then we only consider the test basis functions to be ϕ_{p_s} $(s = 1, \cdots, N_{lb})$.
- There are only N_{lb} non-zero local integrals on e_k with the global basis functions ϕ_{p_s} $(s = 1, \cdots, N_{lb})$:

$$\int_{e_k} cp\phi_i \, ds \; (i = p_1, \cdots, p_{N_{lb}}).$$

Neumann boundary condition

- In fact, we have

$$\psi_{n_k s} = \phi_{p_s}|_{E_{n_k}} \ (s = 1, \cdots, N_{lb}).$$

- That is, instead of the original non-zero local integrals with the global basis functions ϕ_{p_s} $(s = 1, \cdots, N_{lb})$, we will compute the following non-zero local integrals with the local basis functions $\psi_{n_k s}$ $(s = 1, \cdots, N_{lb})$:

$$\int_{e_k} cp\psi_{n_k\beta} \ ds \ (\beta = 1, \cdots, N_{lb}).$$

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

Neumann boundary condition

- $P(:, boundaryedges(3:4, k))$ provides the coordinates of the two end points of the k^{th} boundary edge. We discuss three cases based on these coordinates.
- Case 1: If a boundary edge is vertical, then it can be described as $x = c$ $(y_1 \leq y \leq y_2)$. The $y-$coordinates of the Gauss quadrature nodes on this boundary edge and the Gauss quadrature weights can be obtained from the 1D local Gauss quadrature on $[y_1, y_2]$. And the $x-$coordinates of the Gauss quadrature nodes are fixed to be c.

Neumann boundary condition

- Case 2: If a boundary edge is horizontal, then it can be described as $y = c$ $(x_1 \leq x \leq x_2)$. The $x-$coordinates of the Gauss quadrature nodes on this boundary edge and the Gauss quadrature weights can be obtained from the 1D local Gauss quadrature on $[x_1, x_2]$. And the $y-$coordinates of the Gauss quadrature nodes are fixed to be c.
- Case 3: Otherwise, a boundary edge can be described as $y = ax + b$ $(x_1 \leq x \leq x_2)$. The $x-$coordinates of the Gauss quadrature nodes on this boundary edge and the Gauss quadrature weights can be obtained from the 1D local Gauss nodes in $[x_1, x_2]$. And the $y-$coordinates of the Gauss quadrature nodes are obtained from $y = ax + b$.
- The case 3 with $a = 0$ and $b = c$ is equivalent to case 2. Hence case 2 and case 3 can be combined into one case.

Neumann boundary condition

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

- When the test function is ϕ_i, the corresponding non-zero local integrals should be assembled to v_i.
- Therefore, if we find the global node indices of the test basis functions, we can easily locate where to assemble a non-zero local integral.
- Question: Since we compute

$$\int_{e_k} cp\psi_{n_k\beta}\, ds \ (\beta = 1, \cdots, N_{lb})$$

instead of

$$\int_{e_k} cp\phi_i\, ds \ (i = p_1, \cdots, p_{N_{lb}}),$$

how do we obtain the corresponding global node indices of the local test basis functions $\psi_{n_k\beta}$ $(\beta = 1, \cdots, N_{lb})$?
- Information matrix T_b!

Neumann boundary condition

- Recall that $T_b(\beta, n_k)$ give the global node indices of the local test basis functions $\psi_{n_k\beta}$ $(\beta = 1, \cdots, N_{lb})$.
- That is,

$$\int_{e_k} cp\psi_{n_k\beta}\, ds \; (\beta = 1, \cdots, N_{lb})$$

should be assembled to v_i where $i = T_b(\beta, n_k)$.

Neumann boundary condition

Algorithm VI-1:

- Initialize the vector: $v = sparse(N_b, 1)$;
- Compute the integrals and assemble them into v:

```
FOR $k = 1, \cdots, nbe$:
    IF $boundaryedges(1,k)$ shows Neumann boundary condition, THEN
        $n_k = boundaryedges(2,k)$;
        FOR $\beta = 1, \cdots, N_{lb}$:
            Compute $r = \int_{e_k} cp\psi_{n_k\beta}\ ds$;
            $v(T_b(\beta, n_k), 1) = v(T_b(\beta, n_k), 1) + r$;
        END
    ENDIF
END
```

Neumann boundary condition

- If we follow Algorithm VI-1 to develop a subroutine to assemble the vector arising from

$$\int_{e_k} \tilde{p} \frac{\partial^{a+b}\psi_{n_k\beta}}{\partial x^a \partial y^b}\, ds,$$

  then Algorithm VI-1 is equivalent to calling this subroutine with parameters: $a = b = 0$ and $\tilde{p} = cp$.

Neumann boundary condition

Algorithm VI:

- Initialize the vector: $v = sparse(N_b, 1)$;
- Compute the integrals and assemble them into v:

```
FOR $k = 1, \cdots, nbe$:
    IF $boundaryedges(1, k)$ shows Neumann boundary condition, THEN
        $n_k = boundaryedges(2, k)$;
        FOR $\beta = 1, \cdots, N_{lb}$:
            Compute $r = \int_{e_k} \tilde{p} \frac{\partial^{a+b} \psi_{n_k \beta}}{\partial x^a \partial y^b} \, ds$;
            $v(T_b(\beta, n_k), 1) = v(T_b(\beta, n_k), 1) + r$;
        END
    ENDIF
END
```

Neumann boundary condition

Recall

- Matrix *boundarynodes*:
- $boundarynodes(1, k)$ is the type of the k^{th} boundary finite element node: Dirichlet (-1), Neumann (-2), Robin (-3)......
- The intersection nodes of Dirichlet boundary condition and other boundary conditions usually need to be treated as Dirichlet boundary nodes.
- $boundarynodes(2, k)$ is the global node index of the k^{th} boundary boundary finite element node.
- Set $nbn = size(boundarynodes, 2)$ to be the number of boundary finite element nodes;

Neumann boundary condition

- Example 2: Use the finite element method to solve the following equation on the domain $\Omega = [-1, 1] \times [-1, 1]$:

$$\begin{aligned} -\nabla \cdot (\nabla u) &= -2e^{x+y}, \\ u &= e^{-1+y} \text{ on } x = -1, \\ u &= e^{1+y} \text{ on } x = 1, \\ \nabla u \cdot \vec{n} &= -e^{x-1} \text{ on } y = -1, \\ u &= e^{x+1} \text{ on } y = 1. \end{aligned}$$

- The analytic solution of this problem is $u = e^{x+y}$, which can be used to compute the error of the numerical solution.

Neumann boundary condition

- Let's code for the linear and quadratic finite element method of the 2D second order elliptic equation with Neumann boundary condition!
- Open your Matlab!

Neumann boundary condition

h	$\|u - u_h\|_\infty$	$\|u - u_h\|_0$	$\lvert u - u_h\rvert_1$
$1/8$	1.3358×10^{-2}	5.1224×10^{-3}	1.8523×10^{-1}
$1/16$	3.4487×10^{-3}	1.2793×10^{-3}	9.2559×10^{-2}
$1/32$	8.7622×10^{-4}	3.1973×10^{-4}	4.6273×10^{-2}
$1/64$	2.2084×10^{-4}	7.9928×10^{-5}	2.3136×10^{-2}
$1/128$	5.5433×10^{-5}	1.9982×10^{-5}	1.1568×10^{-2}

Table : The numerical errors for linear finite element.

- Any Observation?
- Second order convergence $O(h^2)$ in L^2/L^∞ norm and first order convergence $O(h)$ in H^1 semi-norm, which match the optimal approximation capability expected from piecewise linear functions.

Neumann boundary condition

h	$\|u-u_h\|_\infty$	$\|u-u_h\|_0$	$\|u-u_h\|_1$
$1/8$	1.0956×10^{-4}	3.9285×10^{-5}	2.9874×10^{-3}
$1/16$	1.4074×10^{-5}	4.9015×10^{-6}	7.4668×10^{-4}
$1/32$	1.7835×10^{-6}	6.1244×10^{-7}	1.8667×10^{-4}
$1/64$	2.2447×10^{-7}	7.6549×10^{-8}	4.6667×10^{-5}
$1/128$	2.8155×10^{-8}	9.5686×10^{-9}	1.1667×10^{-5}

Table : The numerical errors for quadratic finite element.

- Any Observation?
- Third order convergence $O(h^3)$ in L^2/L^∞ norm and second order convergence $O(h^2)$ in H^1 semi-norm, which match the optimal approximation capability expected from piecewise quadratic functions.

Robin boundary conditions

- Consider

$$
\begin{aligned}
&-\nabla\cdot(c\nabla u)=f \quad \text{in } \Omega,\\
&\nabla u\cdot\vec{n}+ru=q \quad \text{on } \Gamma_2\subseteq\partial\Omega,\\
&u=g \quad \text{on } \partial\Omega/\Gamma_2.
\end{aligned}
$$

- Recall

$$\int_\Omega c\nabla u\cdot\nabla v\ dxdy-\int_{\partial\Omega}(c\nabla u\cdot\vec{n})\,v\ ds=\int_\Omega fv\ dxdy.$$

- Since the solution on $\partial\Omega/\Gamma_2$ is given by $u=g$, then we can choose the test function $v(x)$ such that $v=0$ on $\partial\Omega/\Gamma_2$.

Robin boundary condition

- Since

$$\begin{aligned}\int_{\partial\Omega} (c\nabla u \cdot \vec{n})\, v \ ds &= \int_{\Gamma_2} (c\nabla u \cdot \vec{n})\, v \ ds + \int_{\partial\Omega/\Gamma_2} (c\nabla u \cdot \vec{n})\, v \ ds \\ &= \int_{\Gamma_2} c(q - ru)v \ ds \\ &= \int_{\Gamma_2} cqv \ ds - \int_{\Gamma_2} cruv \ ds,\end{aligned}$$

then

$$\int_{\Omega} c\nabla u \cdot \nabla v \ dxdy - \left(\int_{\Gamma_2} cqv \ ds - \int_{\Gamma_2} cruv \ ds \right) = \int_{\Omega} fv \ dxdy$$

- Hence the weak formulation is

$$\int_{\Omega} c\nabla u \cdot \nabla v \ dxdy + \int_{\Gamma_2} cruv \ ds = \int_{\Omega} fv \ dxdy + \int_{\Gamma_2} cqv \ ds.$$

Robin boundary condition

- Then the Galerkin formulation is to find $u_h \in U_h$ such that

$$\int_\Omega c\nabla u_h \cdot \nabla v_h \; dxdy + \int_{\Gamma_2} c r u_h v_h \; ds = \int_\Omega f v_h \; dxdy + \int_{\Gamma_2} c q v_h \; ds$$

  for any $v_h \in U_h$.

- Recall: Since $u_h \in U_h = span\{\phi_j\}_{j=1}^{N_b}$, then

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

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

- Recall: Choose $v_h = \phi_i$ $(i = 1, \cdots, N_b)$.

Robin boundary condition

- Then for $i = 1, \cdots, N_b$, the finite element formulation gives

$$\int_\Omega c\nabla(\sum_{j=1}^{N_b} u_j\phi_j) \cdot \nabla\phi_i \; dxdy + \int_{\Gamma_2} cr(\sum_{j=1}^{N_b} u_j\phi_j)\phi_i \; ds$$

$$= \int_\Omega f\phi_i \; dxdy + \int_{\Gamma_2} cq\phi_i \; ds,$$

$$\Rightarrow \quad \sum_{j=1}^{N_b} u_j \left[\int_\Omega c\nabla\phi_j \cdot \nabla\phi_i \; dxdy\right] + \sum_{j=1}^{N_b} u_j \left[\int_{\Gamma_2} cr\phi_j\phi_i \; ds\right]$$

$$= \int_\Omega f\phi_i \; dxdy + \int_{\Gamma_2} cq\phi_i \; ds.$$

Robin boundary condition

- Recall: Define the stiffness matrix

$$A = [a_{ij}]_{i,j=1}^{N_b} = \left[\int_\Omega c \nabla \phi_j \cdot \nabla \phi_i \ dxdy \right]_{i,j=1}^{N_b}.$$

- Recall: Define the load vector

$$\vec{b} = [b_i]_{i=1}^{N_b} = \left[\int_\Omega f \phi_i \ dxdy \right]_{i=1}^{N_b}.$$

- Recall: Define the unknown vector

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

- Define the additional vector from the Robin boundary condition

$$\vec{w} = [w_i]_{i=1}^{N_b} = \left[\int_{\Gamma_2} cq\phi_i \ ds \right]_{i=1}^{N_b}.$$

Robin boundary condition

- Define the additional matrix from the Robin boundary condition

$$R = [r_{ij}]_{i,j=1}^{N_b} = \left[\int_{\Gamma_2} c r \phi_j \phi_i \, ds\right]_{i,j=1}^{N_b}.$$

- Define the new vector $\widetilde{\vec{b}} = \vec{b} + \vec{w}$.
- Define the new matrix $\widetilde{A} = A + R$.
- Then we obtain the linear algebraic system

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

- Code?
- Add one more subroutine for $\vec{w}$ and R to the existing code!

Robin boundary condition

Recall

- Matrix *boundaryedges*:
- $boundaryedges(1, k)$ is the type of the k^{th} boundary edge e_k: Dirichlet (-1), Neumann (-2), Robin (-3)......
- $boundaryedges(2, k)$ is the index of the element which contains the k^{th} boundary edge e_k.
- Each boundary edge has two end nodes. We index them as the first and the second counterclock wise along the boundary.
- $boundaryedges(3, k)$ is the global node index of the first end node of the k^{th} boundary boundary edge e_k.
- $boundaryedges(4, k)$ is the global node index of the second end node of the k^{th} boundary boundary edge e_k.
- Set $nbe = size(boundaryedges, 2)$ to be the number of boundary edges;

Robin boundary condition

- The idea for the assembly of the matrix R and the vector $\vec{w}$ is similar to that of the stiffness matrix and the load vector. We have

$$\begin{aligned} w_i &= \int_{\Gamma_2} cq\phi_i \, ds = \sum_{\substack{e_k \subset \Gamma_2 \\ 1 \le k \le nbe}} \int_{e_k} cq\phi_i \, ds, \ i = 1, \cdots, N_b, \\ r_{ij} &= \int_{\Gamma_2} cr\phi_j\phi_i \, ds = \sum_{\substack{e_k \subset \Gamma_2 \\ 1 \le k \le nbe}} \int_{e_k} cr\phi_j\phi_i \, ds, \ i, j = 1, \cdots, N_b. \end{aligned}$$

- Loop over all the boundary edges;
- Compute all non-zero local integrals on each Robin boundary edge for the vector $\vec{w}$ and the matrix R;
- Assemble these non-zero local integrals into the corresponding entries of the vector $\vec{w}$ and the matrix R.

Robin boundary condition

Compute all non-zero local integrals on each Robin boundary edge for the vector $\vec{w}$ and the matrix R:

- The index of the element which contains the k^{th} boundary edge e_k is $n_k = boundaryedges(2,k)$. Then on e_k, we get non-zero local integrals only when the test and trial basis functions are corresponding to the finite element nodes of the n_k^{th} element E_{n_k}.
- Let $p_s = T_b(s,n)$ $(s = 1, \cdots, N_{lb})$.
- Then we only consider the test basis functions to be ϕ_{p_s} $(s = 1, \cdots, N_{lb})$.

Robin boundary condition

- There are only N_{lb} non-zero local integrals on e_k with the global basis functions ϕ_{p_s} $(s = 1, \cdots, N_{lb})$:

$$\int_{e_k} cq\phi_i \, ds, \;\; i = p_1, \cdots, p_{N_{lb}},$$

$$\int_{e_k} cr\phi_j\phi_i \, ds, \;\; i, j = p_1, \cdots, p_{N_{lb}}.$$

- In fact, we have

$$\psi_{n_k s} = \phi_{p_s}|_{E_{n_k}} \;\; (s = 1, \cdots, N_{lb}).$$

Robin boundary condition

- That is, instead of the original non-zero local integrals with the global basis functions ϕ_{p_s} $(s = 1, \cdots, N_{lb})$, we will compute the following non-zero local integrals with the local basis functions $\psi_{n_k s}$ $(s = 1, \cdots, N_{lb})$:

$$\int_{e_k} cp\psi_{n_k\beta}\, ds, \;\; \beta = 1, \cdots, N_{lb},$$

$$\int_{e_k} cr\psi_{n_k\beta}\psi_{n_k\alpha}\, ds, \;\; \alpha, \beta = 1, \cdots, N_{lb}.$$

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

Robin boundary condition

Recall

- $P(:, boundaryedges(3:4, k))$ provides the coordinates of the two end points of the k^{th} boundary edge. We discuss three cases based on these coordinates.
- Case 1: If a boundary edge is vertical, then it can be described as $x = c$ $(y_1 \leq y \leq y_2)$. The $y-$coordinates of the Gauss quadrature nodes on this boundary edge and the Gauss quadrature weights can be obtained from the 1D local Gauss quadrature on $[y_1, y_2]$. And the $x-$coordinates of the Gauss quadrature nodes are fixed to be c.

Robin boundary condition

- Case 2: If a boundary edge is horizontal, then it can be described as $y = c$ $(x_1 \leq x \leq x_2)$. The $x-$coordinates of the Gauss quadrature nodes on this boundary edge and the Gauss quadrature weights can be obtained from the 1D local Gauss quadrature on $[x_1, x_2]$. And the $y-$coordinates of the Gauss quadrature nodes are fixed to be c.
- Case 3: Otherwise, a boundary edge can be described as $y = ax + b$ $(x_1 \leq x \leq x_2)$. The $x-$coordinates of the Gauss quadrature nodes on this boundary edge and the Gauss quadrature weights can be obtained from the 1D local Gauss nodes in $[x_1, x_2]$. And the $y-$coordinates of the Gauss quadrature nodes are obtained from $y = ax + b$.
- The case 3 with $a = 0$ and $b = c$ is equivalent to case 2. Hence case 2 and case 3 can be combined into one case.

Robin boundary condition

Assemble the non-zero local integrals into $\vec{w}$ and R:

- When the test function is ϕ_i, the corresponding non-zero local integrals should be assembled to w_i.
- When the trial function is ϕ_i and the test function is ϕ_j, the corresponding non-zero local integrals should be assembled to r_{ij}.
- 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.

Robin boundary condition

- Question: Since we compute

$$\int_{e_k} cq\psi_{n_k\beta}\ ds\ (\beta = 1, \cdots, N_{lb})$$

instead of

$$\int_{e_k} cq\phi_i\ ds\ (i = p_1, \cdots, p_{N_{lb}}),$$

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

Robin boundary condition

- Question: Since we compute

$$\int_{e_k} cr\psi_{n_k\beta}\psi_{n_k\alpha}\, ds \ (\alpha, \beta = 1, \cdots, N_{lb})$$

instead of

$$\int_{e_k} cr\phi_j\phi_i\, ds \ (i, j = p_1, \cdots, p_{N_{lb}}),$$

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

- Information matrix T_b!

Robin boundary condition

- Recall that $T_b(\alpha, n_k)$ and $T_b(\beta, n_k)$ give the global node indices of the local trial and test basis functions $\psi_{n_k\alpha}$ and $\psi_{n_k\beta}$ $(\alpha, \beta = 1, \cdots, N_{lb})$.
- That is,
$$\int_{e_k} cq\psi_{n_k\beta} \, ds \ (\beta = 1, \cdots, N_{lb})$$
should be assembled to w_i where $i = T_b(\beta, n_k)$.
- And
$$\int_{e_k} cr\psi_{n_k\alpha}\psi_{n_k\beta} \, ds \ (\alpha, \beta = 1, \cdots, N_{lb})$$
should be assembled to r_{ij} where $i = T_b(\beta, n_k)$ and $j = T_b(\alpha, n_k)$.

Robin boundary condition

Algorithm VII-1:

- Initialize $R = sparse(N_b, N_b)$ and $w = sparse(N_b, 1)$;
- Compute the integrals and assemble them into R and w:

```
$FOR\ k = 1, \cdots, nbe$:
    $IF\ boundaryedges(1, k)$ shows Robin boundary condition, $THEN$
        $n_k = boundaryedges(2, k)$;
        $FOR\ \beta = 1, \cdots, N_{lb}$:
            Compute $r = \int_{e_k} cq\psi_{n_k\beta}\ ds$;
            $w(T_b(\beta, n_k), 1) = w(T_b(\beta, n_k), 1) + r$;
        $END$
        $FOR\ \alpha = 1, \cdots, N_{lb}$:
            $FOR\ \beta = 1, \cdots, N_{lb}$:
                Compute $r = \int_{e_k} cr\psi_{n_k\beta}\psi_{n_k\alpha}\ ds$;
                Add $r$ to $R(T_b(\beta, n_k), T_b(\alpha, n_k))$;
            $END$
        $END$
    $ENDIF$
$END$
```

Robin boundary condition

Algorithm VII-2:

- Initialize $R = sparse(N_b, N_b)$ and $w = sparse(N_b, 1)$;
- Compute the integrals and assemble them into R and w:

```
$FOR\ k = 1, \cdots, nbe$:
    $IF\ boundaryedges(1, k)$ shows Robin boundary condition, $THEN$
        $n_k = boundaryedges(2, k)$;
        $FOR\ \beta = 1, \cdots, N_{lb}$:
            Compute $r = \int_{e_k} cq\psi_{n_k\beta}\ ds$;
            $w(T_b(\beta, n_k), 1) = w(T_b(\beta, n_k), 1) + r$;
            $FOR\ \alpha = 1, \cdots, N_{lb}$:
                Compute $r = \int_{e_k} cr\psi_{n_k\beta}\psi_{n_k\alpha}\ ds$;
                Add $r$ to $R(T_b(\beta, n_k), T_b(\alpha, n_k))$;
            $END$
        $END$
    $ENDIF$
$END$
```

Robin boundary condition

- If we follow Algorithm VII-1 to develop a subroutine to assemble the vector arising from

$$\int_{e_k} \tilde{p} \frac{\partial^{a+b} \psi_{n_k \beta}}{\partial x^a \partial y^b} \, ds,$$

and the vector arising from

$$\int_{e_k} \tilde{r} \frac{\partial^{m+s} \psi_{n_k \alpha}}{\partial x^m \partial y^s} \frac{\partial^{d+l} \psi_{n_k \beta}}{\partial x^d \partial y^l} \, ds,$$

then Algorithm VII-1 is equivalent to calling this subroutine with parameters: $a = b = r = s = d = l = 0$, $\tilde{p} = cq$, and $\tilde{r} = cr$.

- Note that the vector part is exactly the same as what we had for the Neumann boundary condition!

Robin boundary condition

Algorithm VII:

- Initialize $R = sparse(N_b, N_b)$ and $w = sparse(N_b, 1)$;
- Compute the integrals and assemble them into R and w:

```
$FOR\ k = 1, \cdots, nbe$:
    $IF\ boundaryedges(1,k)$ shows Robin boundary condition, $THEN$
        $n_k = boundaryedges(2,k)$;
        $FOR\ \beta = 1, \cdots, N_{lb}$:
            Compute $r = \int_{e_k} \tilde{p} \frac{\partial^{a+b} \psi_{n_k \beta}}{\partial x^a \partial y^b}\ ds$;
            $w(T_b(\beta, n_k), 1) = w(T_b(\beta, n_k), 1) + r$;
        $END$
        $FOR\ \alpha = 1, \cdots, N_{lb}$:
            $FOR\ \beta = 1, \cdots, N_{lb}$:
                Compute $r = \int_{e_k} c r \psi_{n_k \beta} \psi_{n_k \alpha}\ ds$;
                Add $r$ to $R(T_b(\beta, n_k), T_b(\alpha, n_k))$;
            $END$
        $END$
    $ENDIF$
$END$
```

Robin boundary condition

Recall

- Matrix *boundarynodes*:
- $boundarynodes(1, k)$ is the type of the k^{th} boundary finite element node: Dirichlet (-1), Neumann (-2), Robin (-3)......
- The intersection nodes of Dirichlet boundary condition and other boundary conditions usually need to be treated as Dirichlet boundary nodes.
- $boundarynodes(2, k)$ is the global node index of the k^{th} boundary boundary finite element node.
- Set $nbn = size(boundarynodes, 2)$ to be the number of boundary finite element nodes;

Robin boundary condition

- Example 3: Use the finite element method to solve the following equation on the domain $\Omega = [-1,1] \times [-1,1]$:

$$\begin{aligned} -\nabla \cdot (\nabla u) &= -2e^{x+y}, \\ u &= e^{-1+y} \ \text{ on } x = -1, \\ u &= e^{1+y} \ \text{ on } x = 1, \\ \nabla u \cdot \vec{n} + u &= 0 \ \text{ on } y = -1, \\ u &= e^{x+1} \ \text{ on } y = 1. \end{aligned}$$

- The analytic solution of this problem is $u = e^{x+y}$, which can be used to compute the error of the numerical solution.

Robin boundary condition

- Let's code for the linear and quadratic finite element method of the 2D second order elliptic equation with Neumann boundary condition!
- Open your Matlab!

Robin boundary condition

h	$\lVert u - u_h \rVert_\infty$	$\lVert u - u_h \rVert_0$	$\lvert u - u_h \rvert_1$
$1/8$	1.3358×10^{-2}	5.1094×10^{-3}	1.8523×10^{-1}
$1/16$	3.4487×10^{-3}	1.2760×10^{-3}	9.2559×10^{-2}
$1/32$	8.7622×10^{-4}	3.1893×10^{-4}	4.6273×10^{-2}
$1/64$	2.2084×10^{-4}	7.9727×10^{-5}	2.3136×10^{-2}
$1/128$	5.5433×10^{-5}	1.9932×10^{-5}	1.1568×10^{-2}

Table : The numerical errors for linear finite element.

- Any Observation?
- Second order convergence $O(h^2)$ in L^2/L^∞ norm and first order convergence $O(h)$ in H^1 semi-norm, which match the optimal approximation capability expected from piecewise linear functions.

Robin boundary condition

h	$\|u-u_h\|_\infty$	$\|u-u_h\|_0$	$\|u-u_h\|_1$
$1/8$	1.0956×10^{-4}	3.9278×10^{-5}	2.9874×10^{-3}
$1/16$	1.4074×10^{-5}	4.9012×10^{-6}	7.4668×10^{-4}
$1/32$	1.7835×10^{-6}	6.1243×10^{-7}	1.8667×10^{-4}
$1/64$	2.2447×10^{-7}	7.6549×10^{-8}	4.6667×10^{-5}
$1/128$	2.8155×10^{-8}	9.5686×10^{-9}	1.1667×10^{-5}

Table : The numerical errors for quadratic finite element.

- Any Observation?
- Third order convergence i$O(h^3)$ in L^2/L^∞ norm and second order convergence $O(h^2)$ in H^1 semi-norm, which match the optimal approximation capability expected from piecewise quadratic functions.

Dirichlet/Neumann/Robin mixed boundary condition

- Consider

$$\begin{aligned}
&-\nabla\cdot(c\nabla u) = f \quad \text{in } \Omega,\\
&\nabla u\cdot\vec{n} = p \quad \text{on } \Gamma_1 \subset \partial\Omega,\\
&\nabla u\cdot\vec{n} + ru = q \quad \text{on } \Gamma_2 \subseteq \partial\Omega,\\
&u = g \quad \text{on } \partial\Omega/(\Gamma_1\cup\Gamma_2).
\end{aligned}$$

- Recall

$$\int_\Omega c\nabla u\cdot\nabla v\ dxdy - \int_{\partial\Omega}(c\nabla u\cdot\vec{n})\,v\ ds = \int_\Omega fv\ dxdy.$$

- Since the solution on $\partial\Omega/(\Gamma_1\cup\Gamma_2)$ is given by $u = g$, then we can choose the test function $v(x)$ such that $v = 0$ on $\partial\Omega/(\Gamma_1\cup\Gamma_2)$.

Dirichlet/Neumann/Robin mixed boundary condition

- Hence

$$\int_\Omega c\nabla u \cdot \nabla v \; dxdy + \int_{\Gamma_2} cruv \; ds$$
$$= \int_\Omega fv \; dxdy + \int_{\Gamma_1} cpv \; ds + \int_{\Gamma_2} cqv \; ds.$$

- Code?
- Combine all of the subroutines for Dirichlet/Neumann/Robin boundary conditions.

Non-isotropic equation

- Consider

$$
\begin{aligned}
&-\nabla\cdot(c\nabla u) = f \quad \text{in } \Omega,\\
&c\nabla u\cdot\vec{n} = p \quad \text{on } \Gamma_1 \subset \partial\Omega,\\
&c\nabla u\cdot\vec{n} + ru = q \quad \text{on } \Gamma_2 \subseteq \partial\Omega,\\
&u = g \quad \text{on } \partial\Omega/(\Gamma_1\cup\Gamma_2),
\end{aligned}
$$

where

$$c = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}.$$

- Recall

$$\int_\Omega c\nabla u\cdot\nabla v\ dxdy - \int_{\partial\Omega}(c\nabla u\cdot\vec{n})\,v\ ds = \int_\Omega fv\ dxdy.$$

Non-isotropic equation

- Since the solution on $\partial\Omega/(\Gamma_1 \cup \Gamma_2)$ is given by $u = g$, then we can choose the test function $v(x)$ such that $v = 0$ on $\partial\Omega/(\Gamma_1 \cup \Gamma_2)$.
- Hence

$$\begin{aligned}&\int_\Omega c\nabla u \cdot \nabla v \ dxdy + \int_{\Gamma_2} ruv \ ds\\ = \ &\int_\Omega fv \ dxdy + \int_{\Gamma_1} pv \ ds + \int_{\Gamma_2} qv \ ds.\end{aligned}$$

where

$$\begin{aligned}c\nabla u \cdot \nabla v &= \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix}\\ &= \begin{pmatrix} c_{11}u_x + c_{12}u_y \\ c_{21}u_x + c_{22}u_y \end{pmatrix} \cdot \begin{pmatrix} v_x \\ v_y \end{pmatrix}\\ &= c_{11}u_xv_x + c_{12}u_yv_x + c_{21}u_xv_y + c_{22}u_yv_y.\end{aligned}$$

Non-isotropic equation

- Code? Just call Algorithm I-3 four times! Everything else is the same as before!
- Call Algorithm I-3 with $r = 1$, $s = 0$, $p = 1$, $q = 0$, and $c = c_{11}$ to obtain A_1;
- Call Algorithm I-3 with $r = 0$, $s = 1$, $p = 1$, $q = 0$, and $c = c_{11}$ to obtain A_2;
- Call Algorithm I-3 with $r = 1$, $s = 0$, $p = 0$, $q = 1$, and $c = c_{21}$ to obtain A_3;
- Call Algorithm I-3 with $r = 0$, $s = 1$, $p = 0$, $q = 1$, and $c = c_{22}$ to obtain A_4.
- Then the stiffness matrix is $A = A_1 + A_2 + A_3 + A_4$.

A more general second order equation

- Consider

$$-\nabla \cdot (c\nabla u) + au = f \quad \text{in } \Omega,$$
$$c\nabla u \cdot \vec{n} = p \quad \text{on } \Gamma_1 \subset \partial\Omega,$$
$$c\nabla u \cdot \vec{n} + ru = q \quad \text{on } \Gamma_2 \subseteq \partial\Omega,$$
$$u = g \quad \text{on } \partial\Omega/(\Gamma_1 \cup \Gamma_2),$$

where

$$c = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}.$$

- Then

$$\int_\Omega c\nabla u \cdot \nabla v \; dxdy - \int_{\partial\Omega} (c\nabla u \cdot \vec{n}) \; v \; ds + \int_\Omega auv \; dxdy = \int_\Omega fv \; dxdy.$$

A more general second order equation

- Since the solution on $\partial\Omega/(\Gamma_1 \cup \Gamma_2)$ is given by $u = g$, then we can choose the test function $v(x)$ such that $v = 0$ on $\partial\Omega/(\Gamma_1 \cup \Gamma_2)$.
- Hence

$$\begin{aligned} & \int_\Omega c\nabla u \cdot \nabla v \ dxdy + \int_\Omega auv \ dxdy + \int_{\Gamma_2} ruv \ ds \\ = \ & \int_\Omega fv \ dxdy + \int_{\Gamma_1} pv \ ds + \int_{\Gamma_2} qv \ ds. \end{aligned}$$

where

$$c\nabla u \cdot \nabla v \ = \ c_{11}u_x v_x + c_{12}u_y v_x + c_{21}u_x v_y + c_{22}u_y v_y.$$

A more general second order equation

- Code? Just call Algorithm I-3 five times! Everything else is the same as before!
- Call Algorithm I-3 with $r = 0$, $s = 0$, $p = 0$, $q = 0$, and $c = a$ to obtain A_0;
- Call Algorithm I-3 with $r = 1$, $s = 0$, $p = 1$, $q = 0$, and $c = c_{11}$ to obtain A_1;
- Call Algorithm I-3 with $r = 0$, $s = 1$, $p = 1$, $q = 0$, and $c = c_{11}$ to obtain A_2;
- Call Algorithm I-3 with $r = 1$, $s = 0$, $p = 0$, $q = 1$, and $c = c_{21}$ to obtain A_3;
- Call Algorithm I-3 with $r = 0$, $s = 1$, $p = 0$, $q = 1$, and $c = c_{22}$ to obtain A_4.
- Then the stiffness matrix is $A = A_0 + A_1 + A_2 + A_3 + A_4$.

Linear regression for the convergence order

- Consider $\|u - u_h\| = Ch^r$.
- The goal is to design a linear regression to obtain the C and r based on the h and errors given in the table.
- First,

$$\begin{aligned} log\left(\|u - u_h\|\right) &= log(Ch^r) \\ &= log(C) + log(h^r) \\ &= log(C) + r\ log(h). \end{aligned}$$

- Let $y = log\left(\|u - u_h\|\right),\ x = log(h),\ a = r,\ b = log(C)$.
- Then $y = ax + b$.
- For different h, we can obtain the corresponding x and y.
- Then by the regular linear regression, we can obtain a and b, which give us the $C = e^b$ and $r = a$.