Chapter 2

FEM for Poisson problem in 2D

Consider the two dimensional Poisson problem

$$-\Delta u = f \qquad \text{in } \Omega \tag{2.1a}$$

$$u = 0 \qquad \text{on } \partial\Omega \tag{2.1b}$$

In order to solve the problem (2.1), we multiply both sides of (2.1a) by the test function $v(x, y)$ and integrate over the domain,

$$-\int_\Omega \Delta u v \, d\boldsymbol{x} = \int_\Omega f v \, d\boldsymbol{x}. \tag{2.2}$$

Then, the weak formulation of the problem (2.1) is as follows: find $u(x, y) \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v \, d\boldsymbol{x} = \int_\Omega f v \, d\boldsymbol{x}, \tag{2.3}$$

for all $v \in H_0^1(\Omega)$. The weak formulation (2.3) is obtained from (2.2) by using the boundary condition (2.1b) and integration by parts.

For the finite element approach, we need a conforming finite element space for the rectangular elements

$$V_h^k = \{v_h \in H_0^1(\Omega) \mid v_h|_R \in Q_k(R), \ \forall R \in \mathcal{T}_h\}.$$

where $Q_k(R)$ is the polynomial function space of degrees $\leq k$ in each variable. Then, the variational formulation of the problem (2.1) is as follows: find $u_h \in V_h^k$ such that

$$\int_\Omega \nabla u_h \cdot \nabla v_h \, d\boldsymbol{x} = \int_\Omega f v_h \, d\boldsymbol{x}, \tag{2.4}$$

for all $v_h \in V_h^k$.

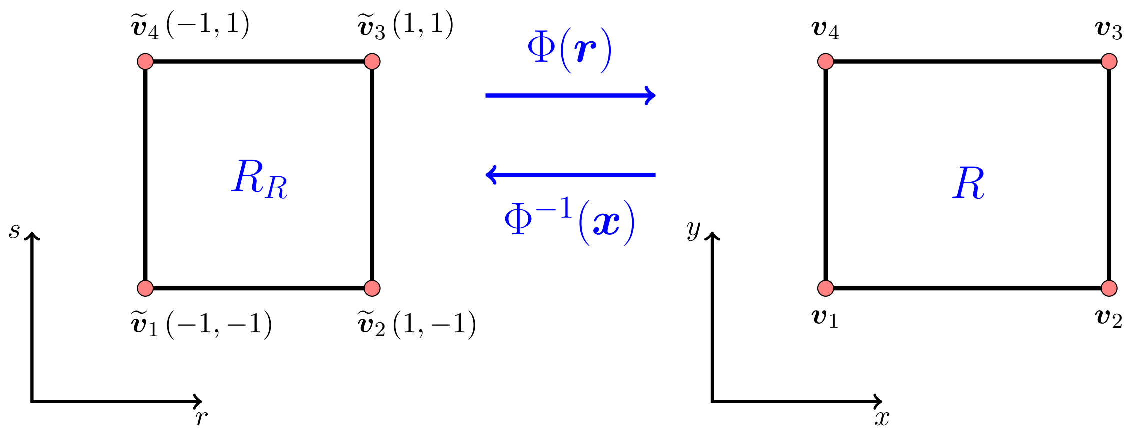


Figure 2.1: An affine mapping from the reference rectangle R_Q to a rectangle Q.

2.1 Affine mapping

Let us define the reference rectangle R_R as

$$R_R = \{\boldsymbol{r} = (r, s) \mid -1 \le r \le 1, \text{ and } -1 \le s \le 1\}.$$

Barycentric coordinates $(\widetilde{\lambda}_1,\ \widetilde{\lambda}_2,\ \widetilde{\lambda}_3,\ \widetilde{\lambda}_4)$ have the properties

$$\begin{cases} 0 \le \widetilde{\lambda}_i(\boldsymbol{r}) \le 1, & i = 1,\ 2,\ 3,\ 4 \\ \widetilde{\lambda}_1(\boldsymbol{r}) + \widetilde{\lambda}_2(\boldsymbol{r}) + \widetilde{\lambda}_3(\boldsymbol{r}) + \widetilde{\lambda}_4(\boldsymbol{r}) = 1. \end{cases} \tag{2.5}$$

Let us define a rectangle R as

$$R = \mathrm{span}\{\boldsymbol{v}_1,\ \boldsymbol{v}_2,\ \boldsymbol{v}_3, \boldsymbol{v}_4\}, \quad \boldsymbol{v}_i = (v_i^{(1)}, v_i^{(2)}),$$

where $\boldsymbol{v}_1,\ \boldsymbol{v}_2,\ \boldsymbol{v}_3, \boldsymbol{v}_4$ are vertices of R. Then, we have an affine mapping Φ such that

$$\Phi(\boldsymbol{r}) = \boldsymbol{v}_1\widetilde{\lambda}_1(\boldsymbol{r}) + \boldsymbol{v}_2\widetilde{\lambda}_2(\boldsymbol{r}) + \boldsymbol{v}_3\widetilde{\lambda}_3(\boldsymbol{r}) + \boldsymbol{v}_4\widetilde{\lambda}_4(\boldsymbol{r}) = \boldsymbol{x}, \tag{2.6}$$

where $\boldsymbol{x}$ is a point in R. The equation (2.6) can be rewritten as

$$\Phi(\boldsymbol{r}) = \boldsymbol{v}_1 + \frac{r+1}{2}(\boldsymbol{v}_2 - \boldsymbol{v}_1) + \frac{s+1}{2}(\boldsymbol{v}_4 - \boldsymbol{v}_1) = \boldsymbol{x}.$$

From the above equation,

$$\frac{\partial \boldsymbol{x}}{\partial r} = \frac{1}{2}\boldsymbol{v}_2 - \frac{1}{2}\boldsymbol{v}_1, \qquad \frac{\partial \boldsymbol{x}}{\partial s} = \frac{1}{2}\boldsymbol{v}_4 - \frac{1}{2}\boldsymbol{v}_1.$$

Hence,

$$x_r = (v_2^{(1)} - v_1^{(1)})/2, \qquad y_r = (v_2^{(2)} - v_1^{(2)})/2,$$
$$x_s = (v_4^{(1)} - v_1^{(1)})/2, \qquad y_s = (v_4^{(2)} - v_1^{(2)})/2.$$

Note that, for a rectangular element R, $y_r = x_s = 0$. Using the property $I = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{r}}\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}$, we have

$$I = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{r}}\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}} = \begin{bmatrix} x_r & x_s \\ y_r & y_s \end{bmatrix}\begin{bmatrix} r_x & r_y \\ s_x & s_y \end{bmatrix} = \begin{bmatrix} x_r & 0 \\ 0 & y_s \end{bmatrix}\begin{bmatrix} r_x & r_y \\ s_x & s_y \end{bmatrix}.$$

$$\Longrightarrow \quad \begin{bmatrix} r_x & r_y \\ s_x & s_y \end{bmatrix} = \begin{bmatrix} x_r & 0 \\ 0 & y_s \end{bmatrix}^{-1} = \frac{1}{x_r y_x}\begin{bmatrix} y_s & 0 \\ 0 & x_r \end{bmatrix}$$

Therefore,

$$r_x = \frac{y_s}{J}, \quad r_y = 0, \quad s_x = 0, \quad s_y = \frac{x_r}{J},$$

where $J = x_r y_s$.

Let R be the rectangular element and $\lambda_1(\boldsymbol{x})$, $\lambda_2(\boldsymbol{x})$, $\lambda_3(\boldsymbol{x})$ and $\lambda_4(\boldsymbol{x})$ be the barycentric coordinates in R. Then, we have

$$\begin{aligned} \lambda_1(\boldsymbol{x}) &= \left(\frac{v_2^{(1)} - x}{v_2^{(1)} - v_1^{(1)}}\right)\left(\frac{v_4^{(2)} - y}{v_4^{(2)} - v_1^{(2)}}\right), & \lambda_2(\boldsymbol{x}) &= \left(\frac{x - v_1^{(1)}}{v_2^{(1)} - v_1^{(1)}}\right)\left(\frac{v_4^{(2)} - y}{v_4^{(2)} - v_1^{(2)}}\right) \\ \lambda_3(\boldsymbol{x}) &= \left(\frac{x - v_1^{(1)}}{v_2^{(1)} - v_1^{(1)}}\right)\left(\frac{y - v_1^{(2)}}{v_4^{(2)} - v_1^{(2)}}\right), & \lambda_4(\boldsymbol{x}) &= \left(\frac{v_2^{(1)} - x}{v_2^{(1)} - v_1^{(1)}}\right)\left(\frac{y - v_1^{(2)}}{v_4^{(2)} - v_1^{(2)}}\right) \end{aligned} \tag{2.7}$$

Since $\lambda_i(\boldsymbol{x})$, $\widetilde{\lambda}(\boldsymbol{r})$ $(i = 1, 2, 3, 4)$ are bilinear functions in R, R_R respectively, and Φ is a linear mapping, we have

$$\lambda_i(\boldsymbol{x}) = \widetilde{\lambda}_i(\Phi^{-1}(\boldsymbol{x})), \tag{2.8a}$$
$$\widetilde{\lambda}_i(\boldsymbol{r}) = \lambda_i(\Phi(\boldsymbol{r})). \tag{2.8b}$$

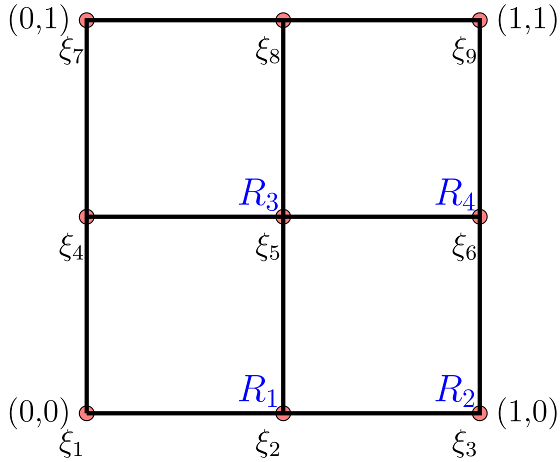
Therefore, we can obtain the following relations by simple calculation,

$$\begin{aligned} \frac{d}{dx}\lambda_i(\boldsymbol{x}) &= \frac{dr}{dx}\frac{d}{dr}\widetilde{\lambda}_i(\Phi^{-1}(\boldsymbol{x})) + \frac{ds}{dx}\frac{d}{ds}\widetilde{\lambda}_i(\Phi^{-1}(\boldsymbol{x})) \\ &= r_x\frac{d}{dr}\widetilde{\lambda}_i(\boldsymbol{r}) + s_x\frac{d}{ds}\widetilde{\lambda}_i(\boldsymbol{r}) = r_x\frac{d}{dr}\widetilde{\lambda}_i(\boldsymbol{r}) \end{aligned} \tag{2.9a}$$

$$\begin{aligned} \frac{d}{dy}\lambda_i(\boldsymbol{x}) &= \frac{dr}{dy}\frac{d}{dr}\widetilde{\lambda}_i(\Phi^{-1}(\boldsymbol{x})) + \frac{ds}{dy}\frac{d}{ds}\widetilde{\lambda}_i(\Phi^{-1}(\boldsymbol{x})) \\ &= r_y\frac{d}{dr}\widetilde{\lambda}_i(\boldsymbol{r}) + s_y\frac{d}{ds}\widetilde{\lambda}_i(\boldsymbol{r}) = s_y\frac{d}{ds}\widetilde{\lambda}_i(\boldsymbol{r}). \end{aligned} \tag{2.9b}$$

2.2 Triangulation

Consider a domain $\Omega = (0, 1)^2$. The number of nodes in the triangulation $\mathcal{T}_h$ is determined by the number of elements and the polynomial order. Let N be the number of nodes, $h = 1/M$ be the length of an edge in $\mathcal{T}_h$. Then the number of elements is M^2 and the number of nodes is $N = (kM + 1)^2$. The data for a given triangulation $\mathcal{T}_h$ are described in $N \times 2$ matrix **c4n**, $M^2 \times 4$ matrix **n4e**, N_D dimensional vector **n4db** and $M^2 \times (k+1)^2$ matrix **ind4e**. Here, N_D is the number of nodes on Dirichlet boundary. For example, if $\Omega = (0,\ 1)^2$, $M = 2$ and $k = 1$, data are stored as follows.

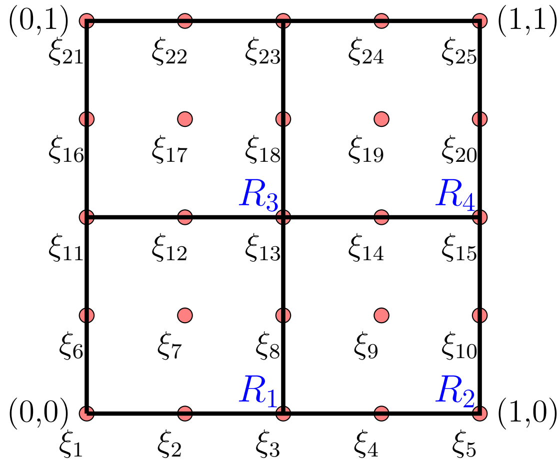


$$\text{c4n} = \begin{bmatrix} 0 & 0 \\ 1/2 & 0 \\ 1 & 0 \\ 0 & 1/2 \\ 1/2 & 1/2 \\ 1 & 1/2 \\ 0 & 1 \\ 1/2 & 1 \\ 1 & 1 \end{bmatrix}, \quad \text{n4db} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 6 \\ 7 \\ 8 \\ 9 \end{bmatrix},$$

$$\text{n4e} = \begin{bmatrix} 1 & 2 & 5 & 4 \\ 2 & 3 & 6 & 5 \\ 4 & 5 & 8 & 7 \\ 5 & 6 & 9 & 8 \end{bmatrix}, \quad \text{ind4e} = \begin{bmatrix} 1 & 2 & 4 & 5 \\ 2 & 3 & 5 & 6 \\ 4 & 5 & 7 & 8 \\ 5 & 6 & 8 & 9 \end{bmatrix}.$$

Here, the size of n4e is the same as the size of ind4e. However, their elements are different each other. The matrix n4e has 4 columns and the components in each row are corresponding to the vertex nodes in the corresponding element. These nodes have usually in a counterclockwise orientation, and the first vertex is the bottom-left node in an element. On the other hand, ind4e has $(k+1)^2$ columns and the components in each row are corresponding to all nodes in the corresponding element. These nodes are ordered from left to right and from bottom to top.

If $\Omega = (0,\ 1)^2$, $M = 2$ and $k = 2$, data are stored as follows.



$$
\text{c4n} = \begin{bmatrix} 0 & 0 \\ 1/4 & 0 \\ 1/2 & 0 \\ 3/4 & 0 \\ 1 & 0 \\ 0 & 1/4 \\ 1/4 & 1/4 \\ 1/2 & 1/4 \\ 3/4 & 1/4 \\ 1 & 1/4 \\ 0 & 1/2 \\ 1/4 & 1/2 \\ 1/2 & 1/2 \\ 3/4 & 1/2 \\ 1 & 1/2 \\ 0 & 3/4 \\ 1/4 & 3/4 \\ 1/2 & 3/4 \\ 3/4 & 3/4 \\ 1 & 3/4 \\ 0 & 1 \\ 1/4 & 1 \\ 1/2 & 1 \\ 3/4 & 1 \\ 1 & 1 \end{bmatrix}, \quad \text{n4db} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 6 \\ 10 \\ 11 \\ 15 \\ 16 \\ 20 \\ 21 \\ 22 \\ 23 \\ 24 \\ 25 \end{bmatrix},
$$

$$\text{n4e} = \begin{bmatrix} 1 & 3 & 13 & 11 \\ 3 & 5 & 15 & 13 \\ 11 & 13 & 23 & 21 \\ 13 & 15 & 25 & 23 \end{bmatrix}, \quad \text{ind4e} = \begin{bmatrix} 1 & 2 & 3 & 6 & 7 & 8 & 11 & 12 & 13 \\ 3 & 4 & 5 & 8 & 9 & 10 & 13 & 14 & 15 \\ 11 & 12 & 13 & 16 & 17 & 18 & 21 & 22 & 23 \\ 13 & 14 & 15 & 18 & 19 & 20 & 23 & 24 & 25 \end{bmatrix}.$$

In this case, each element has an extra point except vertex nodes. Thus, the size of ind4e is larger than that of n4e. Here, the components in n4e and ind4e are ordered similarly to n4e and ind4e for $k = 1$, respectively.

mesh_fem_2d_rectangle The following Matlab code generates an uniform rectangular mesh on the domain $[xl, xr] \times [yl, yr]$ in 2D with M_x elements along x-direction and M_y elements along y-direction. Also this code returns an index matrix for continuous k-th order polynomial approximations.

```
function [c4n,n4e,ind4e,inddb] = mesh_fem_2d_rectangle(xl,xr,yl,yr,Mx,My,k)

ind4e = zeros(Mx*My, (k+1)^2);
tmp = repmat((1:k:k*Mx)', 1, My) ...
    + repmat((0:k*(k*Mx+1):((k*Mx+1)*((My-1)*k+1)-1)), Mx, 1);
tmp = tmp(:);
for j=1:k+1
    ind4e(:,(j-1)*(k+1)+(1:(k+1))) = repmat(tmp+(j-1)*(k*Mx+1), 1, k+1) ...
        + repmat((0:k), Mx*My, 1);
end

n4e = ind4e(:,[1 k+1 (k+1)^2 (k*(k+1)+1)]);

inddb = [1:(k*Mx+1), 2*(k*Mx+1):(k*Mx+1):(k*Mx+1)*(k*My+1), ...
                ((k*Mx+1)*(k*My+1)-1):-1:(k*My*(k*Mx+1)+1), ...
                ((k*My-1)*(k*Mx+1)+1):-(k*Mx+1):(k*Mx+2)]';

x = linspace(xl, xr, k*Mx+1);
y = linspace(yl, yr, k*My+1);
y = repmat(y, k*Mx+1, 1);
x = repmat(x, k*My+1, 1)';
c4n = [x(:), y(:)];
end
```

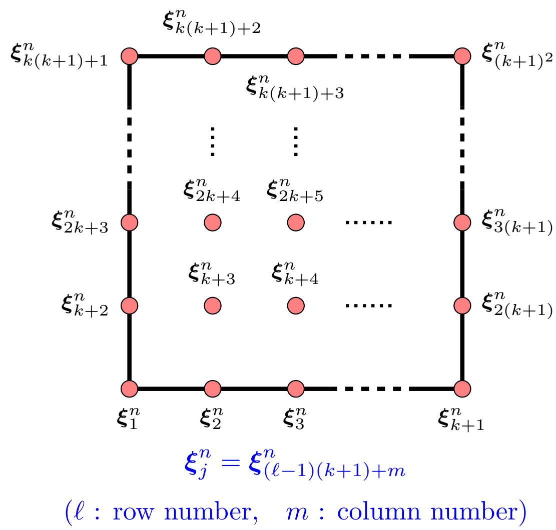


Figure 2.2: Node numbering in the n-th rectangular element

2.3 Basis functions of V_h^k and Numerical solution

Basis functions of V_h^k are piecewise polynomial and they have Kronecker delta property,

$$\psi_i(\boldsymbol{\xi}_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \tag{2.10}$$

$$\sum_{i=1}^{N} \psi_i(\boldsymbol{x}) = 1 \qquad \forall x \in \Omega \tag{2.11}$$

where ψ_i is the i-th basis function of V_h^k and $\boldsymbol{\xi}_j$ is the j-th node point of a triangulation $\mathcal{T}_h$. These basis functions can be locally expressed similar to the 1D case. The local basis function $\psi_i^n(x)$ is the i-th basis function in the n-th element. Then, the following relations hold

$$\begin{aligned} \psi_i^n(\boldsymbol{x}) &= \phi_m(x)\phi_\ell(y), \qquad (i = (\ell-1)(k+1) + m,\ 1 \leq m,\ \ell \leq k+1) \\ \psi_i^n(\boldsymbol{x}) &= 0 \quad \text{if } x \in \Omega \setminus R_n \\ \sum_{i=1}^{(k+1)^2} \psi_i^n(\boldsymbol{x}) &= 1 \quad \forall x \in R_n. \end{aligned}$$

where $\phi_m(x)$ and $\phi_\ell(y)$ are the basis functions in 1D. Figure 2.2 shows node numbering in R_n.

Using these basis functions, the interpolate function of a function $f(\boldsymbol{x})$ can be written as follows

$$\mathcal{I}f(\boldsymbol{x}) = \sum_{i=1}^{N} f_i \psi_i(\boldsymbol{x}) \tag{2.12}$$

where $f_i = f(\boldsymbol{\xi}_i)$. Clearly, $f(\boldsymbol{\xi}_i) = \mathcal{I}f(\boldsymbol{\xi}_i)$ for all $1 \leq i \leq N$ by (2.10). If $f(\boldsymbol{x}) \in V_h^k$, the interpolate function is the same as $f(\boldsymbol{x})$, i.e., $f(\boldsymbol{x}) = \mathcal{I}f(\boldsymbol{x})$. Thus, the numerical solution can be written as follows

$$u_h = \sum_{i=1}^{N} u_i \psi_i \tag{2.13}$$

where $u_i = u_h(\xi_i)$. Especially, the numerical solution can be written locally because the solution is piecewise polynomial function:

$$u_h\Big|_{R_n} = \sum_{i=1}^{(k+1)^2} u_i^n \psi_i^n \tag{2.14}$$

where $u_i^j = u_h(\xi_i^j)$. Then, the gradient of the solution is easily obtained from (2.13) and (2.14).

$$\nabla u_h = \sum_{i=1}^{N} u_i \nabla \psi_i, \tag{2.15a}$$

$$\nabla u_h\Big|_{R_n} = \sum_{i=1}^{(k+1)^2} u_i^n \nabla \psi_i^n. \tag{2.15b}$$

2.4 Mass matrix and Stiffness matrix

For a test function $\psi_i(\boldsymbol{x}) \in V_h^k$, the variational formulation (2.4) can be rewritten as

$$\sum_{j=1}^{N} u_j \int_\Omega \nabla \psi_i \cdot \nabla \psi_j \; d\boldsymbol{x} = \int_\Omega f \psi_i \; d\boldsymbol{x} \tag{2.16}$$

by (2.13) and (2.14). Then, for all basis functions in V_h^k, we have the following finite element system

$$A\boldsymbol{u} = \boldsymbol{b} \tag{2.17}$$

where

$$(A)_{ij} = \int_\Omega \nabla \psi_i \cdot \nabla \psi_j \; d\boldsymbol{x} \tag{2.18a}$$

$$(\boldsymbol{b})_i = \int_\Omega f \psi_i \; d\boldsymbol{x} \tag{2.18b}$$

$$(\boldsymbol{u})_j = u_j. \tag{2.18c}$$

Here, the matrix A is called the global stiffness matrix and the right-hand side b is called the load vector. In order to compute the load vector, f is replaced with the interpolate function $\mathcal{I}f(\boldsymbol{x})$ in general. Thus, FE system (2.17) can be rewritten as

$$A\boldsymbol{u} = M\boldsymbol{f} \tag{2.19}$$

where

$$(M)_{ij} = \int_\Omega \psi_i \psi_j \; d\boldsymbol{x} \tag{2.20a}$$

$$(\boldsymbol{f})_j = f_j. \tag{2.20b}$$

Here, the matrix M is called the global mass matrix. For the accurate computation, f can be replaced by the L^2-orthogonal projection $\pi f(\boldsymbol{x})$.

As in the previous section, basis functions are zero except a few elements. Thus, $\nabla\phi_i$ is also zero vector in the elements where ψ_i is zero. Thus the global stiffness matrix and the global mass matrix can be assembled by using local basis functions.

$$A = \sum_{n=1}^{M} A_{R_n}, \qquad M = \sum_{n=1}^{M} M_{R_n} \tag{2.21}$$

where $(k+1)^2$-by-$(k+1)^2$ matrices A_{R_n} and M_{R_n} are defined as

$$(A_{R_n})_{ij} = \int_{R_n} \nabla\psi_i^n(x) \cdot \nabla\psi_j^n \; d\boldsymbol{x} \tag{2.22}$$

$$(M_{R_n})_{ij} = \int_{R_n} \psi_i^n \psi_j^n \; d\boldsymbol{x} \tag{2.23}$$

where $1 \le i,\ j \le (k+1)^2$. Here A_{R_n} and M_{R_n} are called the local stiffness matrix and the local mass matrix, respectively.

In order to compute gradient, we now introduce differentiation matrices Dx and Dy such that

$$(Dx)_{ij} = \frac{\partial\psi_j}{\partial x}(\boldsymbol{\xi}_i), \qquad (Dy)_{ij} = \frac{\partial\psi_j}{\partial y}(\boldsymbol{\xi}_i). \tag{2.24}$$

Clearly, $\frac{\partial\psi_j}{\partial x}(\boldsymbol{\xi}_i),\ \frac{\partial\psi_j}{\partial y}(\boldsymbol{\xi}_i) \in V_h^k$ because $\psi_i(\boldsymbol{x}) \in V_h^k$. Thus,

$$\frac{\partial}{\partial x}\psi_i(\boldsymbol{x}) = \sum_{j=1}^{N} \frac{\partial\psi_i}{\partial x}(\boldsymbol{\xi}_j)\psi_j(\boldsymbol{x}) = (Dx^t)_i \boldsymbol{\psi}$$

$$\frac{\partial}{\partial y}\psi_i(\boldsymbol{x}) = \sum_{j=1}^{N} \frac{\partial\psi_i}{\partial y}(\boldsymbol{\xi}_j)\psi_j(\boldsymbol{x}) = (Dy^t)_i \boldsymbol{\psi}$$

where $(Dx^t)_i,\ (Dy^t)_i$ are the i-th row of the matrices $Dx^t,\ Dy^t$, i.e. $(Dx^t)_i = [\frac{\partial\psi_i}{\partial x}(\xi_1)\ \cdots\ \frac{\partial\psi_i}{\partial x}(\xi_N)]$, $(Dy^t)_i = [\frac{\partial\psi_i}{\partial y}(\xi_1)\ \cdots\ \frac{\partial\psi_i}{\partial y}(\xi_N)]$, and $\boldsymbol{\psi} = [\psi_1(x)\ \cdots\ \psi_N(x)]^t$. Similar to the stiffness and mass matrices, Dx and Dy can be assembled by the local differentiation matrix

$$Dx = \sum_{n=1}^{M^2} Dx_{R_n}, \qquad (Dx_{R_n})_{ij} = \frac{\partial\psi_j^n}{\partial x}(\boldsymbol{\xi}_i^n)$$
$$Dy = \sum_{n=1}^{M^2} Dx_{R_n}, \qquad (Dy_{R_n})_{ij} = \frac{\partial\psi_j^n}{\partial y}(\boldsymbol{\xi}_i^n).$$

Thus, the gradient of the local basis function $\psi_i^n(\boldsymbol{x})$ is written as

$$\nabla\psi_i^n(\boldsymbol{x}) = \Big(\sum_{j=1}^{(k+1)^2} \frac{\partial\psi_i^n}{dx}(\boldsymbol{\xi}_j^n)\psi_j^n(\boldsymbol{x}),\ \sum_{j=1}^{(k+1)^2} \frac{\partial\psi_i^n}{dy}(\boldsymbol{\xi}_j^n)\psi_j^n(\boldsymbol{x})\Big) = \Big((Dx_{R_n}^t)_i\boldsymbol{\psi}^n,\ (Dy_{R_n}^t)_i\boldsymbol{\psi}^n\Big) \quad (2.25)$$

where $(Dx_{R_n}^t)_i$ and $(Dy_{R_n}^t)_i$ are the i-th row of the matrices $(Dx_{R_n}^t)_i$ and $(Dy_{R_n}^t)_i$, respectively, and $\boldsymbol{\psi}^n = [\psi_1^n(\boldsymbol{x})\ \cdots\ \psi_{(k+1)^2}^n(\boldsymbol{x})]^t$. Then, we have

$$\begin{aligned}
\nabla u_h(\boldsymbol{\xi}_m) &= \sum_{i=1}^{N} u_i\nabla\psi_i(\boldsymbol{\xi}_m) = \sum_{i=1}^{N} u_i \sum_{j=1}^{N} \nabla\psi_i(\boldsymbol{\xi}_j)\psi_j(\boldsymbol{\xi}_m) = \sum_{i=1}^{N} u_i\nabla\psi_i(\boldsymbol{\xi}_m) \\
&= \Big((Dx)_m\boldsymbol{u},\ (Dy)_m\boldsymbol{u}\Big) \\
\nabla u_h(\boldsymbol{\xi}_m^n) &= \sum_{i=1}^{(k+1)^2} u_i^n\nabla\psi_i^n(\boldsymbol{\xi}_m) = \sum_{i=1}^{(k+1)^2} u_i^n \sum_{j=1}^{(k+1)^2} \nabla\psi_i^n(\boldsymbol{\xi}_j)\psi_j(\boldsymbol{\xi}_m^n) = \sum_{i=1}^{(k+1)^2} u_i^n\nabla\psi_i^n(\boldsymbol{\xi}_m^\ell) \\
&= \Big((Dx_{R_n})_m\boldsymbol{u},\ (Dy_{R_n})_m\boldsymbol{u}\Big).
\end{aligned}$$

We can compute the local stiffness and mass matrices using the results in Lemma 1.1 and 1.2 (see, Chapter 1). For convenience, we will use the notations M and S instead of M_{R_n} and A_{R_n} for a fixed element R_n, respectively. Also, ψ and $\widetilde{\psi}$ are basis functions on the element R_n and the reference interval R_R, respectively. By the affine mapping, we have

$$(M)_{ij} = \int_{R_n} \psi_i(\boldsymbol{x})\psi_j(\boldsymbol{x})\ d\boldsymbol{x} = J\int_{R_R} \widetilde{\psi}_i(\boldsymbol{r})\widetilde{\psi}_j(\boldsymbol{r})\ dr = J(M_R)_{ij} \quad (2.26)$$

where M_R is the mass matrix on R_R. By (1.29) (in Chapter 1), (2.9) and (2.15a),

$$
\begin{aligned}
(S)_{ij} &= \int_{R_n} \nabla\psi_i(\boldsymbol{x}) \cdot \nabla\psi_j(\boldsymbol{x})\ d\boldsymbol{x} \\
&= \int_{R_n} \frac{\partial}{\partial x}\psi_i(\boldsymbol{x})\frac{\partial}{\partial x}\nabla\psi_j(\boldsymbol{x})\ d\boldsymbol{x} + \int_{R_n} \frac{\partial}{\partial y}\psi_i(\boldsymbol{x})\frac{\partial}{\partial y}\nabla\psi_j(\boldsymbol{x})\ d\boldsymbol{x} \\
&= \int_{R_n} \Big(\sum_{\ell=1}^{(k+1)^2} \frac{\partial\psi_i}{\partial x}(\boldsymbol{\xi}_\ell)\psi_\ell(\boldsymbol{x})\Big)\Big(\sum_{m=1}^{(k+1)^2} \frac{\partial\psi_j}{\partial x}(\boldsymbol{\xi}_m)\psi_m(\boldsymbol{x})\Big)\ d\boldsymbol{x} \\
&\quad + \int_{R_n} \Big(\sum_{\ell=1}^{(k+1)^2} \frac{\partial\psi_i}{\partial y}(\boldsymbol{\xi}_\ell)\psi_\ell(\boldsymbol{x})\Big)\Big(\sum_{m=1}^{(k+1)^2} \frac{\partial\psi_j}{\partial y}(\boldsymbol{\xi}_m)\psi_m(\boldsymbol{x})\Big)\ d\boldsymbol{x} \\
&= J\int_{R_R} \Big(\sum_{\ell=1}^{(k+1)^2} r_x\frac{\partial\widetilde{\psi}_i}{\partial r}(\widetilde{\boldsymbol{\xi}}_\ell)\widetilde{\psi}_\ell(\boldsymbol{r})\Big)\Big(\sum_{m=1}^{(k+1)^2} r_x\frac{\partial\widetilde{\psi}_j}{\partial r}(\widetilde{\boldsymbol{\xi}}_m)\widetilde{\psi}_m(r)\Big)\ d\boldsymbol{r} \\
&\quad + J\int_{R_R} \Big(\sum_{\ell=1}^{(k+1)^2} s_y\frac{\partial\widetilde{\psi}_i}{\partial s}(\widetilde{\boldsymbol{\xi}}_\ell)\widetilde{\psi}_\ell(\boldsymbol{r})\Big)\Big(\sum_{m=1}^{(k+1)^2} s_y\frac{\partial\widetilde{\psi}_j}{\partial s}(\widetilde{\boldsymbol{\xi}}_m)\widetilde{\psi}_m(r)\Big)\ d\boldsymbol{r} \\
&= J\Big(r_x^2(S_R^{rr})_{ij} + s_y^2(S_R^{ss})_{ij}\Big)
\end{aligned}
\tag{2.27}
$$

and

$$
\begin{aligned}
(S_R^{rr})_{ij} &= \int_{R_R} \Big((Dr_R^t)_i\widetilde{\boldsymbol{\psi}}\Big)\Big((Dr_R^t)_j\widetilde{\boldsymbol{\psi}}\Big)\ d\boldsymbol{r} \\
&= (Dr_R^t)_i M_R (Dr_R)_j \\
&= (Dr_R^t M_R Dr_R)_{ij}
\end{aligned}
\tag{2.28a}
$$

$$
\begin{aligned}
(S_R^{ss})_{ij} &= \int_{R_R} \Big((Ds_R^t)_i\widetilde{\boldsymbol{\psi}}\Big)\Big((Ds_R^t)_j\widetilde{\boldsymbol{\psi}}\Big)\ d\boldsymbol{r} \\
&= (Ds_R^t)_i M_R (Ds_R)_j \\
&= (Ds_R^t M_R Ds_R)_{ij}
\end{aligned}
\tag{2.28b}
$$

where S_R^{rr} and S_R^{ss} are the local stiffness matrices on R_R, Dr_R and Ds_R is the differentiation matrices on R_R, and $\widetilde{\boldsymbol{\psi}} = [\widetilde{\psi}_1(\boldsymbol{r})\ \cdots \widetilde{\psi}_{(k+1)^2}(\boldsymbol{r})]^t$. Thus, the local stiffness and mass matrices are obtained from the reference mass and stiffness matrices by using affine mapping.

$$
M = JM_R, \qquad S = J\Big(r_x^2 S_R^{rr} + s_y^2 S_R^{ss}\Big) \tag{2.29}
$$

Here, we compute the reference matrices for the linear ($k = 1$) and quadratic ($k = 2$) approximations. For the k-th order approximation, the matrices are obtained similarly.

P_1 matrices The basis functions of $P_1(R_R)$ are obtained from the 1D basis functions or barycentric coordinates such that

$$
\begin{aligned}
\widetilde{\psi}_1(\boldsymbol{r}) &= \widetilde{\phi}_1(r)\widetilde{\phi}_1(s) = \widetilde{\lambda}_1^{1D}(r)\widetilde{\lambda}_1^{1D}(s), &\qquad \nabla\widetilde{\psi}_1(\boldsymbol{r}) &= \Big(-\frac{1}{2}\widetilde{\lambda}_1^{1D}(s),\ -\frac{1}{2}\widetilde{\lambda}_1^{1D}(r)\Big),\\
\widetilde{\psi}_2(\boldsymbol{r}) &= \widetilde{\phi}_2(r)\widetilde{\phi}_1(s) = \widetilde{\lambda}_2^{1D}(r)\widetilde{\lambda}_1^{1D}(s), &\qquad \nabla\widetilde{\psi}_2(\boldsymbol{r}) &= \Big(\frac{1}{2}\widetilde{\lambda}_1^{1D}(s),\ -\frac{1}{2}\widetilde{\lambda}_2^{1D}(r)\Big),\\
\widetilde{\psi}_3(\boldsymbol{r}) &= \widetilde{\phi}_1(r)\widetilde{\phi}_2(s) = \widetilde{\lambda}_1^{1D}(r)\widetilde{\lambda}_2^{1D}(s), &\qquad \nabla\widetilde{\psi}_3(\boldsymbol{r}) &= \Big(-\frac{1}{2}\widetilde{\lambda}_2^{1D}(s),\ \frac{1}{2}\widetilde{\lambda}_1^{1D}(r)\Big),\\
\widetilde{\psi}_4(\boldsymbol{r}) &= \widetilde{\phi}_2(r)\widetilde{\phi}_2(s) = \widetilde{\lambda}_2^{1D}(r)\widetilde{\lambda}_2^{1D}(s), &\qquad \nabla\widetilde{\psi}_4(\boldsymbol{r}) &= \Big(\frac{1}{2}\widetilde{\lambda}_2^{1D}(s),\ \frac{1}{2}\widetilde{\lambda}_2^{1D}(r)\Big),
\end{aligned}
$$

Then, (2.23) and the barycentric coordinates in 1D (see, (1.27) in Chapter 1) yield

$$
M_R = \frac{1}{9}\begin{pmatrix} 4 & 2 & 2 & 1\\ 2 & 4 & 1 & 2\\ 2 & 1 & 4 & 2\\ 1 & 2 & 2 & 4 \end{pmatrix}. \tag{2.30}
$$

The differentiation matrices Dr_R and Ds_R are obtained from (2.24)

$$
Dr_R = \frac{1}{2}\begin{pmatrix} -1 & 1 & 0 & 0\\ -1 & 1 & 0 & 0\\ 0 & 0 & -1 & 1\\ 0 & 0 & -1 & 1 \end{pmatrix} \tag{2.31a}
$$

$$
Ds_R = \frac{1}{2}\begin{pmatrix} -1 & 0 & 1 & 0\\ 0 & -1 & 0 & 1\\ -1 & 0 & 1 & 0\\ 0 & -1 & 0 & 1 \end{pmatrix}. \tag{2.31b}
$$

Therefore, the local stiffness matrices are obtained from (2.28)

$$
S_R^{rr} = Dr_R^t M_R Dr_R = \frac{1}{6}\begin{pmatrix} 2 & -2 & 1 & -1\\ -2 & 2 & -1 & 1\\ 1 & -1 & 2 & -2\\ -1 & 1 & -2 & 2 \end{pmatrix} \tag{2.32a}
$$

$$
S_R^{ss} = Ds_R^t M_R Ds_R = \frac{1}{6}\begin{pmatrix} 2 & 1 & -2 & -1\\ 1 & 2 & -1 & -2\\ -2 & -1 & 2 & 1\\ -1 & -2 & 1 & 2 \end{pmatrix}. \tag{2.32b}
$$

P_2 matrices Similar to the P_1 matrices, the basis functions in $P_2(R_R)$ are obtained from the 1D quadratic basis functions and barycentric coordinates such that

$$
\begin{aligned}
\widetilde{\psi}_1(\boldsymbol{r}) &= \widetilde{\phi}_1(r)\widetilde{\phi}_1(s), & \nabla\widetilde{\psi}_1(\boldsymbol{r}) &= \Big(\big(-2\widetilde{\lambda}_1^{1D}(r)+\frac{1}{2}\big)\widetilde{\phi}_1(s),\ \big(-2\widetilde{\lambda}_1^{1D}(s)+\frac{1}{2}\big)\widetilde{\phi}_1(r)\Big),\\
\widetilde{\psi}_2(\boldsymbol{r}) &= \widetilde{\phi}_2(r)\widetilde{\phi}_1(s), & \nabla\widetilde{\psi}_2(\boldsymbol{r}) &= \Big(\big(2\widetilde{\lambda}_1^{1D}(r)-2\widetilde{\lambda}_2^{1D}(r)\big)\widetilde{\phi}_1(s),\ \big(-2\widetilde{\lambda}_1^{1D}(s)+\frac{1}{2}\big)\widetilde{\phi}_2(r)\Big),\\
\widetilde{\psi}_3(\boldsymbol{r}) &= \widetilde{\phi}_3(r)\widetilde{\phi}_1(s), & \nabla\widetilde{\psi}_3(\boldsymbol{r}) &= \Big(\big(2\widetilde{\lambda}_2^{1D}(r)-\frac{1}{2}\big)\widetilde{\phi}_1(s),\ \big(-2\widetilde{\lambda}_1^{1D}(s)+\frac{1}{2}\big)\widetilde{\phi}_3(r)\Big),\\
\widetilde{\psi}_4(\boldsymbol{r}) &= \widetilde{\phi}_1(r)\widetilde{\phi}_2(s), & \nabla\widetilde{\psi}_4(\boldsymbol{r}) &= \Big(\big(-2\widetilde{\lambda}_1^{1D}(r)+\frac{1}{2}\big)\widetilde{\phi}_2(s),\ \big(2\widetilde{\lambda}_1^{1D}(s)-2\widetilde{\lambda}_2^{1D}(s)\big)\widetilde{\phi}_1(r)\Big),\\
\widetilde{\psi}_5(\boldsymbol{r}) &= \widetilde{\phi}_2(r)\widetilde{\phi}_2(s), & \nabla\widetilde{\psi}_5(\boldsymbol{r}) &= \Big(\big(2\widetilde{\lambda}_1^{1D}(r)-2\widetilde{\lambda}_2^{1D}(r)\big)\widetilde{\phi}_2(s),\ \big(2\widetilde{\lambda}_1^{1D}(s)-2\widetilde{\lambda}_2^{1D}(s)\big)\widetilde{\phi}_2(r)\Big),\\
\widetilde{\psi}_6(\boldsymbol{r}) &= \widetilde{\phi}_3(r)\widetilde{\phi}_2(s), & \nabla\widetilde{\psi}_6(\boldsymbol{r}) &= \Big(\big(2\widetilde{\lambda}_2^{1D}(r)-\frac{1}{2}\big)\widetilde{\phi}_2(s),\ \big(2\widetilde{\lambda}_1^{1D}(s)-2\widetilde{\lambda}_2^{1D}(s)\big)\widetilde{\phi}_3(r)\Big),\\
\widetilde{\psi}_7(\boldsymbol{r}) &= \widetilde{\phi}_1(r)\widetilde{\phi}_3(s), & \nabla\widetilde{\psi}_7(\boldsymbol{r}) &= \Big(\big(-2\widetilde{\lambda}_1^{1D}(r)+\frac{1}{2}\big)\widetilde{\phi}_3(s),\ \big(2\widetilde{\lambda}_2^{1D}(s)-\frac{1}{2}\big)\widetilde{\phi}_1(r)\Big),\\
\widetilde{\psi}_8(\boldsymbol{r}) &= \widetilde{\phi}_2(r)\widetilde{\phi}_3(s), & \nabla\widetilde{\psi}_8(\boldsymbol{r}) &= \Big(\big(2\widetilde{\lambda}_1^{1D}(r)-2\widetilde{\lambda}_2^{1D}(r)\big)\widetilde{\phi}_3(s),\ \big(2\widetilde{\lambda}_2^{1D}(s)-\frac{1}{2}\big)\widetilde{\phi}_2(r)\Big),\\
\widetilde{\psi}_9(\boldsymbol{r}) &= \widetilde{\phi}_3(r)\widetilde{\phi}_3(s), & \nabla\widetilde{\psi}_9(\boldsymbol{r}) &= \Big(\big(2\widetilde{\lambda}_2^{1D}(r)-\frac{1}{2}\big)\widetilde{\phi}_3(s),\ \big(2\widetilde{\lambda}_2^{1D}(s)-\frac{1}{2}\big)\widetilde{\phi}_3(r)\Big),
\end{aligned}
$$

Then we have the mass matrix

$$
M_R = \frac{1}{225}\begin{pmatrix}
16 & 8 & -4 & 8 & 4 & -2 & -4 & -2 & 1\\
8 & 64 & 8 & 4 & 32 & 4 & -2 & -16 & -2\\
-4 & 8 & 16 & -2 & 4 & 8 & 1 & -2 & -4\\
8 & 4 & -2 & 64 & 32 & -16 & 8 & 4 & -2\\
4 & 32 & 4 & 32 & 256 & 32 & 4 & 32 & 4\\
-2 & 4 & 8 & -16 & 32 & 64 & -2 & 4 & 8\\
-4 & -2 & 1 & 8 & 4 & -2 & 16 & 8 & -4\\
-2 & -16 & -2 & 4 & 32 & 4 & 8 & 64 & 8\\
1 & -2 & -4 & -2 & 4 & 8 & -4 & 8 & 16
\end{pmatrix}, \tag{2.33}
$$

the differentiation matrices

$$
Dr_R = \frac{1}{2}\begin{pmatrix}
-3 & 4 & -1 & 0 & 0 & 0 & 0 & 0 & 0\\
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
1 & -4 & 3 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & -3 & 4 & -1 & 0 & 0 & 0\\
0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & -4 & 3 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & -3 & 4 & -1\\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -4 & 3
\end{pmatrix}, \tag{2.34a}
$$

$$Ds_R = \frac{1}{2}\begin{pmatrix} -3 & 0 & 0 & 4 & 0 & 0 & -1 & 0 & 0 \\ 0 & -3 & 0 & 0 & 4 & 0 & 0 & -1 & 0 \\ 0 & 0 & -3 & 0 & 0 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -4 & 0 & 0 & 3 & 0 & 0 \\ 0 & 1 & 0 & 0 & -4 & 0 & 0 & 3 & 0 \\ 0 & 0 & 1 & 0 & 0 & -4 & 0 & 0 & 3 \end{pmatrix}, \tag{2.34b}$$

and the stiffness matrices

$$S_R^{rr} = \frac{1}{90}\begin{pmatrix} 28 & -32 & 4 & 14 & -16 & 2 & -7 & 8 & -1 \\ -32 & 64 & -32 & -16 & 32 & -16 & 8 & -16 & 8 \\ 4 & -32 & 28 & 2 & -16 & 14 & -1 & 8 & -7 \\ 14 & -16 & 2 & 112 & -128 & 16 & 14 & -16 & 2 \\ -16 & 32 & -16 & -128 & 256 & -128 & -16 & 32 & -16 \\ 2 & -16 & 14 & 16 & -128 & 112 & 2 & -16 & 14 \\ -7 & 8 & -1 & 14 & -16 & 2 & 28 & -32 & 4 \\ 8 & -16 & 8 & -16 & 32 & -16 & -32 & 64 & -32 \\ -1 & 8 & -7 & 2 & -16 & 14 & 4 & -32 & 28 \end{pmatrix}, \tag{2.35a}$$

$$S_R^{ss} = \frac{1}{90}\begin{pmatrix} 28 & 14 & -7 & -32 & -16 & 8 & 4 & 2 & -1 \\ 14 & 112 & 14 & -16 & -128 & -16 & 2 & 16 & 2 \\ -7 & 14 & 28 & 8 & -16 & -32 & -1 & 2 & 4 \\ -32 & -16 & 8 & 64 & 32 & -16 & -32 & -16 & 8 \\ -16 & -128 & -16 & 32 & 256 & 32 & -16 & -128 & -16 \\ 8 & -16 & -32 & -16 & 32 & 64 & 8 & -16 & -32 \\ 4 & 2 & -1 & -32 & -16 & 8 & 28 & 14 & -7 \\ 2 & 16 & 2 & -16 & -128 & -16 & 14 & 112 & 14 \\ -1 & 2 & 4 & 8 & -16 & -32 & -7 & 14 & 28 \end{pmatrix}, \tag{2.35b}$$

get_matrices_2d_rectangle The following Matlab code generates the mass matrix M_R, the stiffness matrices Srr_R, Sss_R and the differentiation matrices Dr_R, Ds_R for continuous k-th order polynomial approximations on the reference rectangle R_R.

```
function [M_R, Srr_R, Sss_R, Dr_R, Ds_R] = get_matrices_2d_rectangle(k)
if k==1
    M_R = [4 2 2 1; 2 4 1 2; 2 1 4 2; 1 2 2 4]/9;
    Srr_R = [2 -2 1 -1; -2 2 -1 1; 1 -1 2 -2; -1 1 -2 2]/6;
```

```
    Sss_R = [2 1 -2 -1; 1 2 -1 -2; -2 -1 2 1; -1 -2 1 2]/6;
    Dr_R = [-1 1 0 0; -1 1 0 0; 0 0 -1 1; 0 0 -1 1]/2;
    Ds_R = [-1 0 1 0; 0 -1 0 1; -1 0 1 0; 0 -1 0 1]/2;
elseif k==2
    M_R = [16 8 -4 8 4 -2 -4 -2 1; 8 64 8 4 32 4 -2 -16 -2;
        -4 8 16 -2 4 8 1 -2 -4; 8 4 -2 64 32 -16 8 4 -2;
        4 32 4 32 256 32 4 32 4; -2 4 8 -16 32 64 -2 4 8;
        -4 -2 1 8 4 -2 16 8 -4; -2 -16 -2 4 32 4 8 64 8;
        1 -2 -4 -2 4 8 -4 8 16]/225;
    Srr_R = [28 -32 4 14 -16 2 -7 8 -1; -32 64 -32 -16 32 -16 8 -16 8;
        4 -32 28 2 -16 14 -1 8 -7; 14 -16 2 112 -128 16 14 -16 2;
        -16 32 -16 -128 256 -128 -16 32 -16; 2 -16 14 16 -128 112 2 -16 14;
        -7 8 -1 14 -16 2 28 -32 4; 8 -16 8 -16 32 -16 -32 64 -32;
        -1 8 -7 2 -16 14 4 -32 28]/90;
    Sss_R = [28 14 -7 -32 -16 8 4 2 -1; 14 112 14 -16 -128 -16 2 16 2;
        -7 14 28 8 -16 -32 -1 2 4; -32 -16 8 64 32 -16 -32 -16 8;
        -16 -128 -16 32 256 32 -16 -128 -16; 8 -16 -32 -16 32 64 8 -16 -32;
        4 2 -1 -32 -16 8 28 14 -7; 2 16 2 -16 -128 -16 14 112 14;
        -1 2 4 8 -16 -32 -7 14 28]/90;
    Dr_R = [-3 4 -1 0 0 0 0 0 0; -1 0 1 0 0 0 0 0 0; 1 -4 3 0 0 0 0 0 0;
        0 0 0 -3 4 -1 0 0 0; 0 0 0 -1 0 1 0 0 0; 0 0 0 1 -4 3 0 0 0;
        0 0 0 0 0 0 -3 4 -1; 0 0 0 0 0 0 -1 0 1; 0 0 0 0 0 0 1 -4 3]/2;
    Ds_R = [-3 0 0 4 0 0 -1 0 0; 0 -3 0 0 4 0 0 -1 0; 0 0 -3 0 0 4 0 0 -1;
        -1 0 0 0 0 0 1 0 0; 0 -1 0 0 0 0 0 1 0; 0 0 -1 0 0 0 0 0 1;
        1 0 0 -4 0 0 3 0 0; 0 1 0 0 -4 0 0 3 0; 0 0 1 0 0 -4 0 0 3]/2;
else
    M_R = 0; Srr_R = 0; Dr_R = 0;
end
end
```

2.4.1 Matlab codes in 2D with rectangular elements

Now we are ready to assemble the global stiffness matrix A and the global load vector b in (2.17). In the matlab code, A and b are assembled by using the local stiffness matrix (2.27) and the local mass matrix (2.26).

fem_for_poisson_2d_rectangle The following Matlab code solves the Poisson problem. In order to use this code, mesh information (c4n, n4e, n4db, ind4e), matrices (M_R, S_R^{rr}, S_R^{ss}), the source f, and the boundary condition u_D. Then the results of this code are the numerical solution u, the global stiffness matrix A, the global load vector b and the freenodes.

```
function [u, A, b, freenodes] = fem_for_poisson_2d_rectangle(c4n, n4e, ...
    n4db, ind4e, M_R, Srr_R, Sss_R, f, u_D)
number_of_nodes = size(c4n,1);
A = sparse(number_of_nodes, number_of_nodes);
b = zeros(number_of_nodes, 1);
u = b;
for j = 1:length(n4e)
    xr = (c4n(n4e(j,2),1)-c4n(n4e(j,1),1))/2;
    ys = (c4n(n4e(j,4),2)-c4n(n4e(j,1),2))/2;
    J = xr*ys;
    rx=ys/J; sy=xr/J;

    A(ind4e(j,:), ind4e(j,:)) = A(ind4e(j,:),ind4e(j,:)) ...
        + J*(rx^2*Srr_R + sy^2*Sss_R);
    b(ind4e(j,:)) = b(ind4e(j,:)) + J*M_R*f(c4n(ind4e(j,:),:));
end
freenodes = setdiff(1:length(c4n), n4db);
u(n4db) = u_D(c4n(n4db,:));
u(freenodes) = A(freenodes, freenodes)\b(freenodes);
end
```

compute_error_fem_2d_rectangle The following Matlab code computes the semi H1 error between the exact solution and the numerical solution.

```
function error = compute_error_fem_2d_rectangle(c4n, n4e, ind4e, M_R, ...
    Dr_R, Ds_R, u, ux, uy)
error = 0;
for j=1:size(ind4e,1)
    xr = (c4n(n4e(j,2),1)-c4n(n4e(j,1),1))/2;
    ys = (c4n(n4e(j,4),2)-c4n(n4e(j,1),2))/2;
    J = xr*ys;
    rx = ys/J; sy = xr/J;

    Dx_u = rx*Dr_R*u(ind4e(j,:));
    Dy_u = sy*Ds_R*u(ind4e(j,:));
    Dex = ux(c4n(ind4e(j,:),:)) - Dx_u;
    Dey = uy(c4n(ind4e(j,:),:)) - Dy_u;
    error = error + J*(Dex'*M_R*Dex + Dey'*M_R*Dey);
end
```

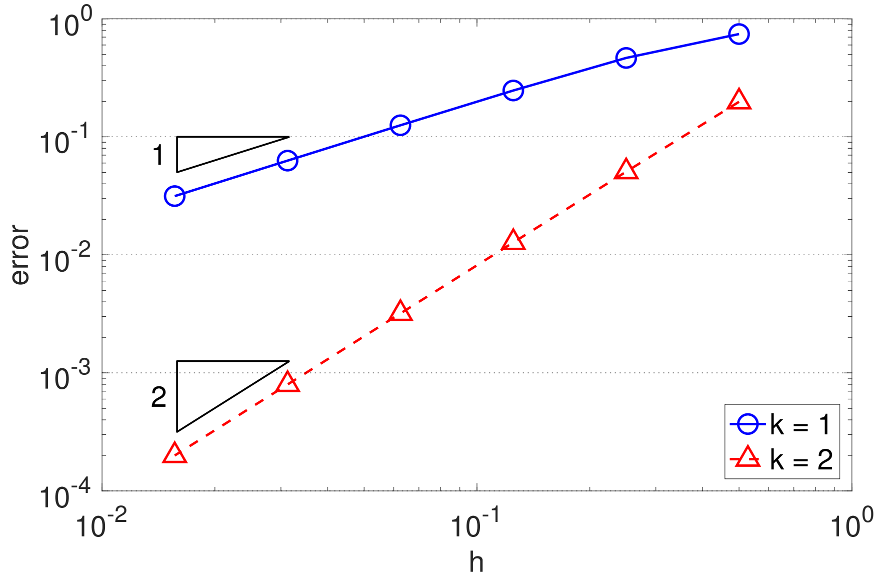


Figure 2.3: Convergence history for Example 2.1

```
error = sqrt(error);
end
```

The following numerical example is introduced to verify above matlab codes.

Example 2.1. *Consider the domain* $\Omega = [0, 1]^2$*. The source term* f *is chosen such that*

$$u = \sin(\pi x)\sin(\pi y) \tag{2.36}$$

is the analytical solution to (2.1)*.*

The results of this example are displayed in Figure 2.3. Here the optimal rates of convergence are obtained. The matlab code for this example is as follows.

main_fem_for_poisson_2d_rectangle The following Matlab code solves the Poisson problem by using several matlab codes such as mesh_fem_2d_rectangle.m, get_matrices_2d_rectangle.m, fem_for_poisson_2d_rectangle.m and compute_error_fem_2d_rectangle.m.

```
iter = 6;
k = 1;
xl = 0; xr = 1; yl = 0; yr = 1; M = 2.^(1:iter);
```

```
f=@(x) 2*pi^2*sin(pi*x(:,1)).*sin(pi*x(:,2));
u_D=@(x) x(:,1)*0;
ux=@(x) pi*cos(pi*x(:,1)).*sin(pi*x(:,2));
uy=@(x) pi*sin(pi*x(:,1)).*cos(pi*x(:,2));

error=zeros(1,iter);
time=zeros(1,iter);
h=1./M;
for j=1:iter
    [c4n,n4e,ind4e,n4db]=mesh_fem_2d_rectangle(xl,xr,yl,yr,M(j),M(j),k);
    [M_R,Srr_R,Sss_R,Dr_R,Ds_R]=get_matrices_2d_rectangle(k);
    u=fem_for_poisson_2d_rectangle(c4n,n4e,n4db,ind4e,M_R,Srr_R,Sss_R,f,u_D);
    error(j)=compute_error_fem_2d_rectangle(c4n,n4e,ind4e,M_R,Dr_R,Ds_R,u,ux,uy);
end
```

Exercises

1. Add the matrices for the cubic approximations ($k = 3$) in **get_matrices_2d_rectangle.m** and check the convergence rate.

2. Modify **fem_for_poisson_2d_rectangle_ex2.m** to solve the Poisson problem with non-homogeneous Dirichlet boundary condition,

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= \ f(\boldsymbol{x}) \qquad && \text{in } \Omega \\ u(\boldsymbol{x}) &= u_D(\boldsymbol{x}) \qquad && \text{on } \partial\Omega. \end{aligned}$$

3. Modify **fem_for_poisson_2d_rectangle_ex3.m** to solve the Poisson problem with mixed boundary condition,

$$\begin{aligned} -\Delta u(\boldsymbol{x}) &= \ f(\boldsymbol{x}) \qquad && \text{in } \Omega \\ u(\boldsymbol{x}) &= u_D(\boldsymbol{x}) \qquad && \text{on } \Gamma_D \\ \nabla u(x) \cdot \boldsymbol{n} &= u_N(\boldsymbol{x}) \qquad && \text{on } \Gamma_N, \end{aligned}$$

where Γ_D denotes the Dirichlet boundary, Γ_N denotes the Neumann boundary, and $\boldsymbol{n}$ is the outward unit normal vector.