Finite Element Methods

FEM for Poisson problem in 2D - Rectangular element

2016-2 CSE6820

Outline

2D Poisson problem

Consider the two dimensional Poisson problem

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

Weak formulation

Find $u(x, y) \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla v \, d\boldsymbol{x} = \int_\Omega f v \, d\boldsymbol{x}, \qquad \forall v \in H_0^1(\Omega).$$

Variational formulation

Find $u_h \in V_h^k$ such that

$$\int_\Omega \nabla u_h \cdot \nabla v_h \, d\boldsymbol{x} = \int_\Omega f v_h \, d\boldsymbol{x}, \qquad \forall v_h \in V_h^k$$

where

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

and $Q_k(R)$ is the polynomial function space of degrees $\leq k$ in each variable.

Affine mapping

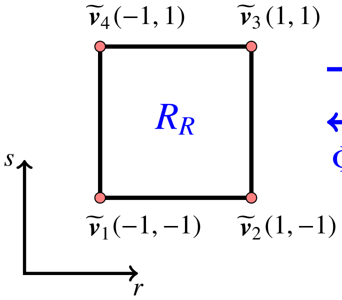


Figure: An affine mapping from the reference rectangle R_Q to a rectangle Q.

Barycentric coordinates

Barycentric coordinates $\{\lambda_i\}_{i=1}^4$ have the properties

$$\begin{cases} 0 \le \lambda_i(\boldsymbol{x}) \le 1, & i = 1,\ 2,\ 3,\ 4 \\ \sum_{i=1}^4 \lambda_i(\boldsymbol{x}) = 1. \end{cases}$$

Then we have an affine mapping Φ such that

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

where $\boldsymbol{x}$ is a point in R.

Properties

1. $\widetilde{\lambda}_k(\boldsymbol{r}) = \widetilde{\lambda}_i^{1D}(r)\widetilde{\lambda}_j^{1D}(s), \qquad (k = 2(j-1)+i, \quad i,j = 1,2)$

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

3. $r_x = \dfrac{y_s}{J}, \qquad r_y = -\dfrac{x_s}{J}, \qquad s_x = -\dfrac{y_r}{J}, \qquad s_y = \dfrac{x_r}{J}$

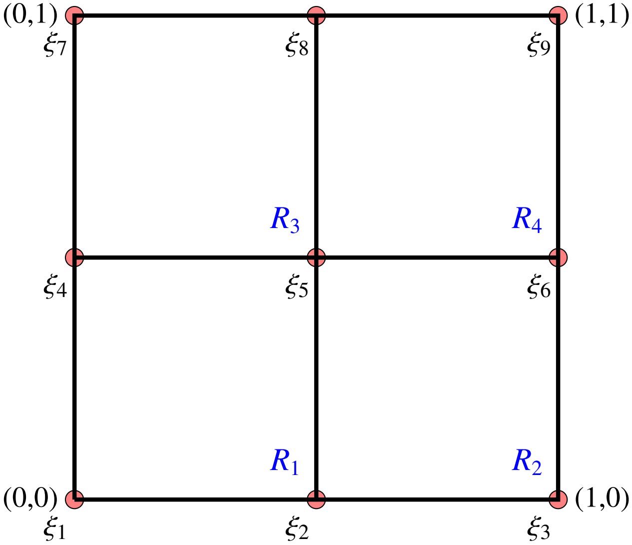
4. $$\lambda_1(\boldsymbol{x}) = \left(\frac{v_2^{(1)} - x}{v_2^{(1)} - v_1^{(1)}}\right)\left(\frac{v_4^{(2)} - y}{v_4^{(2)} - v_1^{(2)}}\right), \qquad \lambda_2(\boldsymbol{x}) = \left(\frac{x - v_1^{(1)}}{v_2^{(1)} - v_1^{(1)}}\right)\left(\frac{v_4^{(2)} - y}{v_4^{(2)} - v_1^{(2)}}\right)$$
$$\lambda_3(\boldsymbol{x}) = \left(\frac{x - v_1^{(1)}}{v_2^{(1)} - v_1^{(1)}}\right)\left(\frac{y - v_1^{(2)}}{v_4^{(2)} - v_1^{(2)}}\right), \qquad \lambda_4(\boldsymbol{x}) = \left(\frac{v_2^{(1)} - x}{v_2^{(1)} - v_1^{(1)}}\right)\left(\frac{y - v_1^{(2)}}{v_4^{(2)} - v_1^{(2)}}\right)$$

5. $\lambda_i(\boldsymbol{x}) = \widetilde{\lambda}_i(\Phi^{-1}(\boldsymbol{x})), \qquad \widetilde{\lambda}_i(\boldsymbol{r}) = \lambda_i(\Phi(\boldsymbol{r}))$

6. $\dfrac{d}{dx}\lambda_i(\boldsymbol{x}) = r_x \dfrac{d}{dr}\widetilde{\lambda}_i(\boldsymbol{r}), \qquad \dfrac{d}{dy}\lambda_i(\boldsymbol{x}) = s_y \dfrac{d}{ds}\widetilde{\lambda}_i(\boldsymbol{r})$

Triangulation data (P_1)

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



Triangulation data (P_1)

$$\text{c4n} = \begin{pmatrix} 0 & 1/2 & 1 & 0 & 1/2 & 1 & 0 & 1/2 & 1 \\ 0 & 0 & 0 & 1/2 & 1/2 & 1/2 & 1 & 1 & 1 \end{pmatrix}$$

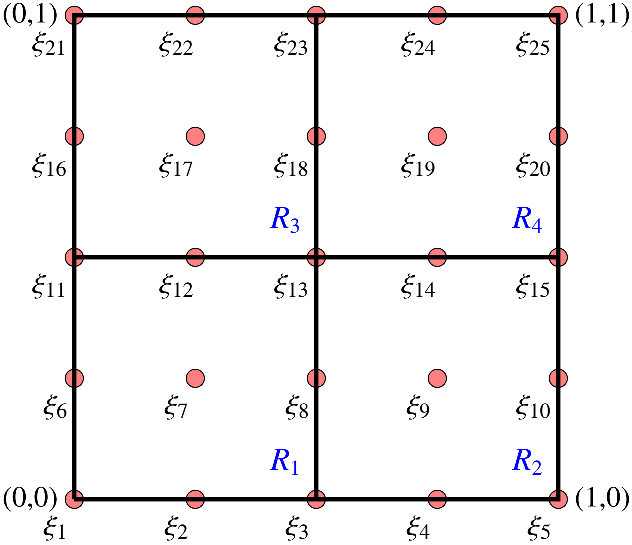
$$\text{n4e} = \begin{pmatrix} 1 & 2 & 4 & 5 \\ 2 & 3 & 5 & 6 \\ 5 & 6 & 8 & 9 \\ 4 & 5 & 7 & 8 \end{pmatrix}$$

$$\text{n4db} = (1,\ 2,\ 3,\ 4,\ 6,\ 7,\ 8,\ 9)$$

$$\text{ind4e} = \begin{pmatrix} 1 & 2 & 4 & 5 \\ 2 & 3 & 5 & 6 \\ 4 & 5 & 7 & 8 \\ 5 & 6 & 8 & 9 \end{pmatrix}$$

Triangulation data (P_2)

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



Triangulation data (P_2)

$$\mathrm{c4n} = \begin{pmatrix} 0 & 1/4 & 1/2 & 3/4 & 1 & 0 & 1/4 & 1/2 & 3/4 & 1 & 0 & 1/4 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 1/4 & 1/2 & 1/2 & 1/2 \end{pmatrix.$$
$$\begin{matrix} 3/4 & 1 & 0 & 1/4 & 1/2 & 3/4 & 1 & 0 & 1/4 & 1/2 & 3/4 & 1 \\ 1/2 & 1/2 & 3/4 & 3/4 & 3/4 & 3/4 & 3/4 & 1 & 1 & 1 & 1 & 1 \end{matrix} \Bigg)$$

$$\mathrm{n4e} = \begin{pmatrix} 1 & 3 & 11 & 13 \\ 3 & 5 & 13 & 15 \\ 13 & 15 & 23 & 25 \\ 11 & 13 & 21 & 23 \end{pmatrix}$$

$$\mathrm{n4db} = (1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 10,\ 11,\ 15,\ 16,\ 20,\ 21,\ 22,\ 23,\ 24,\ 25)$$

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

mesh_FEM2D_Rec

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_FEM2D_Rec(xl,xr,yl,yr,Mx,My,k)
ind4e = zeros((k+1)^2,Mx*My);
tmp = (1:k:k*Mx)' * ones(1,My) ...
    + ones(Mx,1) * (0:k*(k*Mx+1):((k*Mx+1)*((My-1)*k+1)-1));
tmp = tmp(:)';
for j=1:k+1
    ind4e((j-1)*(k+1)+(1:(k+1)), :) = repmat(tmp+(j-1)*(k*Mx+1),k+1,1) ...
        +repmat(0:k,Mx*My,1)';
end
n4e = ind4e([1 k+1 (k+1)^2 (k*(k+1)+1)],:);
inddb = unique([1:(k*Mx+1), (k*Mx+1):(k*Mx+1):(k*Mx+1)*(k*My+1), ...
(k*Mx+1)*(k*My+1):-1:(k*My*(k*Mx+1)+1), (k*My*(k*Mx+1)+1):-(k*Mx+1):1]);
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
```

Basis functions

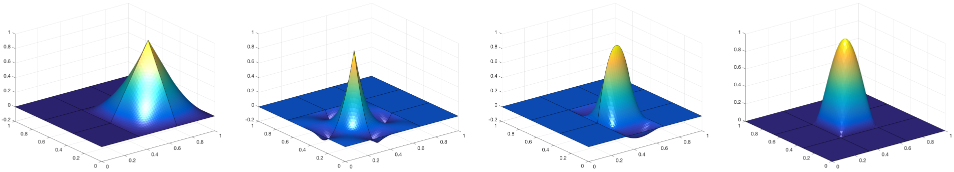


Figure: Global basis functions of V_h^1 (left) and V_h^2 (others) on an interval.

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

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

Local basis functions

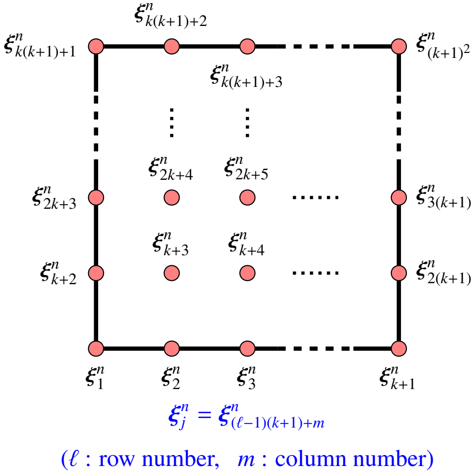
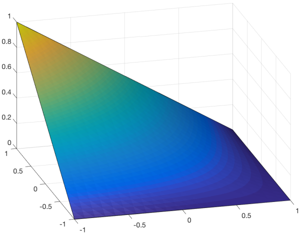
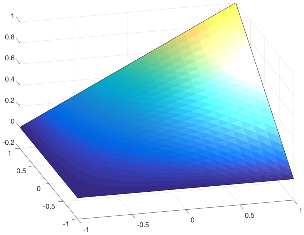
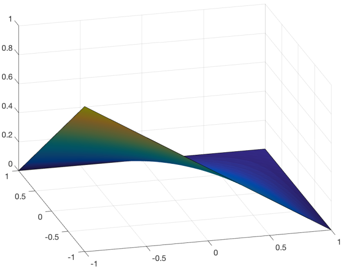


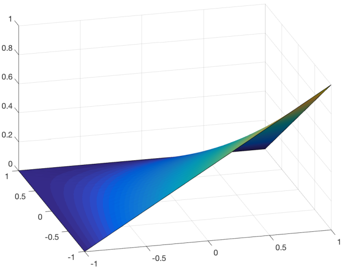
Figure: Node numbering in the n-th rectangular element

Local basis functions

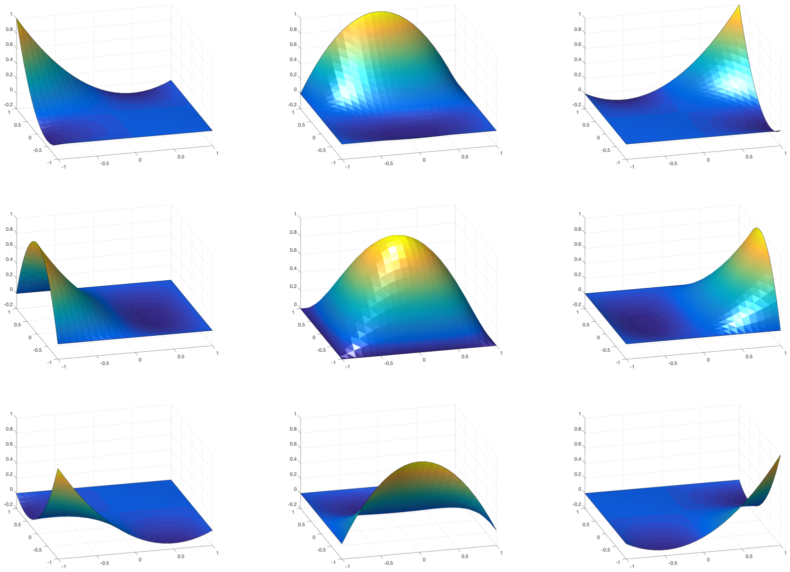








Local basis functions



Local basis functions

$$\psi_i^n(\boldsymbol{x}) = \phi_\ell(x)\phi_m(y)$$

$$\psi_i^n(\boldsymbol{x}) = 0 \quad \text{if } x \in \Omega \setminus R_n$$

$$\sum_{i=1}^{(k+1)^2} \psi_i^n(\boldsymbol{x}) = 1 \quad \forall x \in R_n.$$

Interpolation

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

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

where $f_i = f(\boldsymbol{\xi}_i)$.

Numerical solutions

- Numerical solution

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

- Local solution

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

- Derivatives

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

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

Variational formulation

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

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

Finite element system

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

where

$$(A)_{ij} = \int_\Omega \nabla\psi_i \cdot \nabla\psi_j \, d\boldsymbol{x}$$

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

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

Finite element system

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

where

$$\begin{aligned}
(A)_{ij} &= \int_\Omega \nabla\psi_i \cdot \nabla\psi_j \, d\boldsymbol{x} \\
(\boldsymbol{u})_j &= u_j \\
(M)_{ij} &= \int_\Omega \psi_i\psi_j \, d\boldsymbol{x} \\
(\boldsymbol{f})_j &= f_j.
\end{aligned}$$

Matrices

$$A = \sum_{\ell=1}^{M} A_{R_n}$$

$$M = \sum_{\ell=1}^{M} M_{R_n}$$

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

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

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

where $1 \le i, \; j \le (k+1)^2$.

Differentiation matrix

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

$\Rightarrow$

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

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

where

$$(Dx^t)_i = [\frac{\partial \psi_i}{\partial x}(\xi_1) \ \cdots \ \frac{\partial \psi_i}{\partial x}(\xi_N)]$$

$$(Dy^t)_i = [\frac{\partial \psi_i}{\partial y}(\xi_1) \ \cdots \ \frac{\partial \psi_i}{\partial y}(\xi_N)]$$

$$\boldsymbol{\psi} = [\psi_1(x) \ \cdots \ \psi_N(x)]^t$$

Properties of D

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

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

- $$\nabla \psi_i^n(\boldsymbol{x}) = \Big((Dx_{R_n}^t)_i \boldsymbol{\psi}^n,\ (Dy_{R_n}^t)_i \boldsymbol{\psi}^n\Big)$$

- $$\nabla u_h(\boldsymbol{\xi}_m) = \Big((Dx)_m \boldsymbol{u},\ (Dy)_m \boldsymbol{u}\Big)$$

- $$\nabla u_h(\boldsymbol{\xi}_m^n) = \Big((Dx_{R_n})_m \boldsymbol{u},\ (Dy_{R_n})_m \boldsymbol{u}\Big)$$

Mass matrix

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

$$\Rightarrow \qquad M = JM_R$$

Stiffness matrix

$$(S)_{ij} = J\Big(r_x^2 (S_R^{rr})_{ij} + s_y^2 (S_R^{ss})_{ij}\Big)$$

where

$$(S_R^{rr})_{ij} = (Dr_R^t M_R Dr_R)_{ij}$$
$$(S_R^{ss})_{ij} = (Ds_R^t M_R Ds_R)_{ij}$$

$$\Rightarrow \qquad S = J\Big(r_x^2 S_R^{rr} + s_y^2 S_R^{ss}\Big)$$

P_1 matrices

$$\widetilde{\psi}_1(\boldsymbol{r}) = \widetilde{\phi}_1(r)\widetilde{\phi}_1(s) = \widetilde{\lambda}_1^{1D}(r)\widetilde{\lambda}_1^{1D}(s), \qquad \nabla\widetilde{\psi}_1(\boldsymbol{r}) = \Big(-\frac{1}{2}\widetilde{\lambda}_1^{1D}(s),\ -\frac{1}{2}\widetilde{\lambda}_1^{1D}(r)\Big),$$

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

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

$$\widetilde{\psi}_4(\boldsymbol{r}) = \widetilde{\phi}_2(r)\widetilde{\phi}_2(s) = \widetilde{\lambda}_2^{1D}(r)\widetilde{\lambda}_2^{1D}(s), \qquad \nabla\widetilde{\psi}_4(\boldsymbol{r}) = \Big(\frac{1}{2}\widetilde{\lambda}_2^{1D}(s),\ \frac{1}{2}\widetilde{\lambda}_2^{1D}(r)\Big),$$

P_1 matrices

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

P_1 matrices

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

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

P_1 matrices

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

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

P_2 matrices

$$\widetilde{\psi}_1(\boldsymbol{r}) = \widetilde{\phi}_1(r)\widetilde{\phi}_1(s), \qquad \nabla\widetilde{\psi}_1(\boldsymbol{r}) = \Big((-2\widetilde{\lambda}_1^{1D}(r) + \frac{1}{2})\widetilde{\phi}_1(s),\ (-2\widetilde{\lambda}_1^{1D}(s) + \frac{1}{2})\widetilde{\phi}_1(r)\Big),$$

$$\widetilde{\psi}_2(\boldsymbol{r}) = \widetilde{\phi}_2(r)\widetilde{\phi}_1(s), \qquad \nabla\widetilde{\psi}_2(\boldsymbol{r}) = \Big((2\widetilde{\lambda}_1^{1D}(r) - 2\widetilde{\lambda}_2^{1D}(r))\widetilde{\phi}_1(s),\ (-2\widetilde{\lambda}_1^{1D}(s) + \frac{1}{2})\widetilde{\phi}_2(r)\Big),$$

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

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

$$\widetilde{\psi}_5(\boldsymbol{r}) = \widetilde{\phi}_2(r)\widetilde{\phi}_2(s), \qquad \nabla\widetilde{\psi}_5(\boldsymbol{r}) = \Big((2\widetilde{\lambda}_1^{1D}(r) - 2\widetilde{\lambda}_2^{1D}(r))\widetilde{\phi}_2(s),\ (2\widetilde{\lambda}_1^{1D}(s) - 2\widetilde{\lambda}_2^{1D}(s))\widetilde{\phi}_2(r)\Big),$$

$$\widetilde{\psi}_6(\boldsymbol{r}) = \widetilde{\phi}_3(r)\widetilde{\phi}_2(s), \qquad \nabla\widetilde{\psi}_6(\boldsymbol{r}) = \Big((2\widetilde{\lambda}_2^{1D}(r) - \frac{1}{2})\widetilde{\phi}_2(s),\ (2\widetilde{\lambda}_1^{1D}(s) - 2\widetilde{\lambda}_2^{1D}(s))\widetilde{\phi}_3(r)\Big),$$

$$\widetilde{\psi}_7(\boldsymbol{r}) = \widetilde{\phi}_1(r)\widetilde{\phi}_3(s), \qquad \nabla\widetilde{\psi}_7(\boldsymbol{r}) = \Big((-2\widetilde{\lambda}_1^{1D}(r) + \frac{1}{2})\widetilde{\phi}_3(s),\ (2\widetilde{\lambda}_2^{1D}(s) - \frac{1}{2})\widetilde{\phi}_1(r)\Big),$$

$$\widetilde{\psi}_8(\boldsymbol{r}) = \widetilde{\phi}_2(r)\widetilde{\phi}_3(s), \qquad \nabla\widetilde{\psi}_8(\boldsymbol{r}) = \Big((2\widetilde{\lambda}_1^{1D}(r) - 2\widetilde{\lambda}_2^{1D}(r))\widetilde{\phi}_3(s),\ (2\widetilde{\lambda}_2^{1D}(s) - \frac{1}{2})\widetilde{\phi}_2(r)\Big),$$

$$\widetilde{\psi}_9(\boldsymbol{r}) = \widetilde{\phi}_3(r)\widetilde{\phi}_3(s), \qquad \nabla\widetilde{\psi}_9(\boldsymbol{r}) = \Big((2\widetilde{\lambda}_2^{1D}(r) - \frac{1}{2})\widetilde{\phi}_3(s),\ (2\widetilde{\lambda}_2^{1D}(s) - \frac{1}{2})\widetilde{\phi}_3(r)\Big),$$

P_2 matrices

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

P_2 matrices

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

P_2 matrices

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

P_2 matrices

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

P_2 matrices

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

MatrixforPoisson_2D_Rec

This 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.

Introduction

Affine mapping

Triangulation

Basis functions of V_h^k

Mass matrix and Stiffness matrix

Matlab codes

FEMforPoisson_2D_Rec

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, fns] = ...
    FEMforPoisson_2D_Rec(c4n,n4e,n4db,ind4e,M_R,Srr_R,Sss_R,f,u_D)
A = sparse(length(c4n),length(c4n));
b = zeros(length(c4n),1);
u = b;
for j=1:length(n4e)
    xr = (c4n(1,n4e(2,j))-c4n(1,n4e(1,j)))/2;
    ys = (c4n(2,n4e(4,j))-c4n(2,n4e(1,j)))/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
fns = setdiff(1:length(c4n), n4db);
u(fns) = A(fns,fns)\b(fns);
end
```

ComputeErrorFEM_2D_Rec

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

```
function error = ...
    ComputeErrorFEM_2D_Rec(c4n,n4e,ind4e,M_R,Dr_R,Ds_R,u,ux,uy)
error = 0;
for j=1:size(ind4e,2)
    xr = (c4n(1,n4e(2,j))-c4n(1,n4e(1,j)))/2;
    ys = (c4n(2,n4e(4,j))-c4n(2,n4e(1,j)))/2;
    J = xr*ys;
    rx=ys/J; sy=xr/J;
    Dex=ux(c4n(:,ind4e(:,j))') - rx*Dr_R*u(ind4e(:,j));
    Dey=uy(c4n(:,ind4e(:,j))') - sy*Ds_R*u(ind4e(:,j));
    error=error+J*(Dex'*M_R*Dex+Dey'*M_R*Dey);
end
error=sqrt(error);
end
```

main_FEMforPoisson_2D_Rec

The following Matlab code solves the Poisson problem by using several matlab codes such as mesh_FEM2D_Rec_rectangle, MatrixforPoisson_2D_Rec, FEMforPoisson_2D_Rec and ComputeErrorFEM_2D_Rec.

```
iter = 10;
xl = 0; xr = 1; yl = 0; yr = 1; k = 2; 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);
h=1./M;
for j=1:iter
    [c4n, n4e, ind4e, n4db] = ...
        mesh_FEM2D_Rec_rectangle(xl, xr, yl, yr, M(j), M(j), k);
    [M_R, Srr_R, Sss_R, Dr_R, Ds_R] = MatrixforPoisson_2D_Rec(k);
    u = FEMforPoisson_2D_Rec(c4n,n4e,n4db,ind4e,M_R,Srr_R,Sss_R,f,u_D);
    error(j) = ComputeErrorFEM_2D_Rec(c4n,n4e,ind4e,M_R,Dr_R,Ds_R,u,ux,uy);
end
```

Example

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

$$u = \sin(\pi x)\sin(\pi y)$$

is the analytical solution to the Poisson problem.

Convergence history

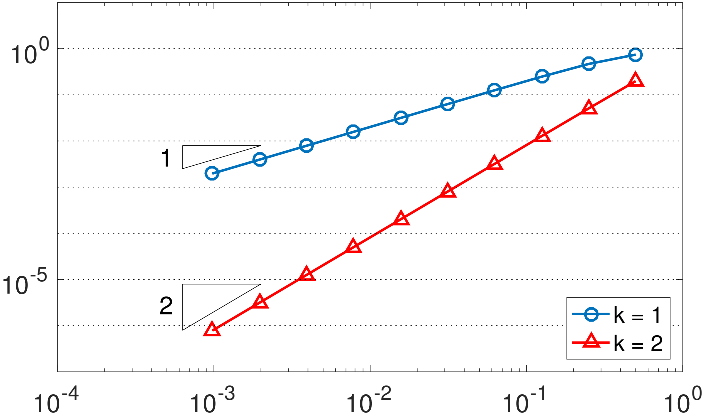


Figure: Convergence history for Example