# Finite Element Methods

FEM for Poisson problem in 1D

2016-2 CSE6820

# Outline

# 1D Poisson problem

Consider the one dimensional Poisson problem

$$\begin{aligned} -u''(x) &= f(x) \qquad \text{in } \Omega = [x_\ell, x_r] \\ u(x_\ell) = u(x_r) &= 0. \end{aligned}$$

# Weak formulation

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

$$\int_\Omega u'(x)v'(x)\,dx = \int_\Omega f(x)v(x)\,dx,$$

# Variational formulation

Find $u_h \in V_h^k$ such that

$$\int_\Omega u_h' v_h' \, dx = \int_\Omega f v_h \, dx, \qquad \forall v_h \in V_h^k$$

where

$$V_h^k = \{v_h \in H_0^1(\Omega) \mid v_h|_I \in P_k(I),\ \forall I \in \mathcal{T}_h\},$$

and $P_k(I)$ is the polynomial function space of degrees $\leq k$.

# Affine mapping

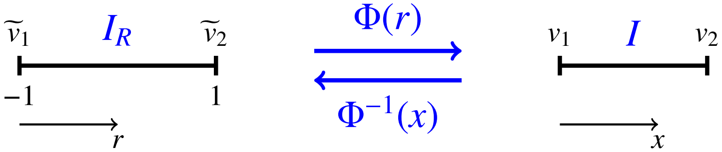



Figure: An affine mapping from the reference interval $I_R$ to an interval $I$.

# Barycentric coordinates

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

$$\begin{cases} 0 \le \widetilde{\lambda}_i(r) \le 1, & i = 1,\ 2, \\ \widetilde{\lambda}_1(r) + \widetilde{\lambda}_2(r) = 1. \end{cases}$$

Then we have an affine mapping $\Phi$ such that

$$\Phi(r) = v_1\widetilde{\lambda}_1(r) + v_2\widetilde{\lambda}_2(r) = x,$$

where $x \in I$.

# Properties

1. $\widetilde{\lambda}_1(r) = \dfrac{1-r}{2}, \qquad \widetilde{\lambda}_2(r) = \dfrac{1+r}{2}$

2. $\dfrac{dx}{dr} = \dfrac{v_2 - v_1}{2}$

3. $r_x = \dfrac{1}{J}$ where $J = \dfrac{v_2 - v_1}{2}$

4. $\lambda_1(x) = \dfrac{v_2 - x}{v_2 - v_1}, \qquad \lambda_2(x) = \dfrac{x - v_1}{v_2 - v_1}$

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

6. $\dfrac{d}{dx}\lambda_i(x) = r_x \dfrac{d}{dr}\widetilde{\lambda}_i(r)$

# Notations

- $\mathcal{T}_h$ : Triangulation with $h = \max_{T\in\mathcal{T}_h} \mathrm{diam}(T)$
- $N$ : the number of nodes
- $M$ : the number of elements
- $I_j$ : the $j$-th element in $\mathcal{T}_h$
- $\xi_i$ : the $i$-th node in $\mathcal{T}_h$
- $\xi_i^j$ : the $i$-th node in $I_j$

# Nodes in $\mathcal{T}_h$

$$N = kM + 1,$$
$$\xi_i^j = \xi_{k(j-1)+i},$$
$$\xi_{k+1}^j = \xi_1^{j+1}.$$

Here the indices $i$ and $j$ satisfy $1 \le j \le M, \ 1 \le i \le k+1$.

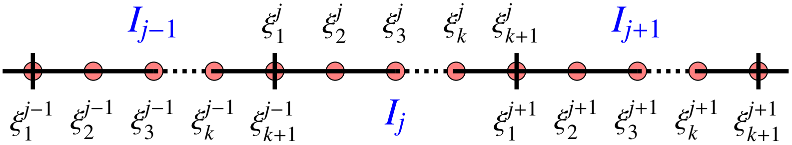
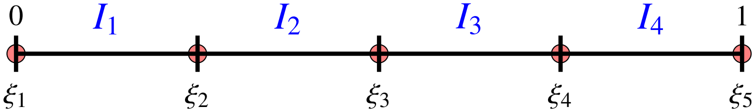



Figure: Nodes in $\mathcal{T}_h$ with $k$.

# Triangulation data ($P_1$)

If $\Omega = [0,\ 1]$, $M = 4$ and $k = 1$, data are stored as follows.



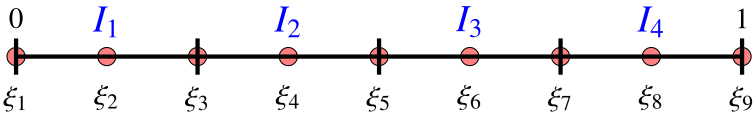

$$\mathrm{c4n} = (0,\ 1/4,\ 1/2,\ 3/4,\ 1)$$

$$\mathrm{n4e} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \end{pmatrix} = \mathrm{ind4e}$$

$$\mathrm{n4db} = (1,\ 5)$$

# Triangulation data ($P_2$)

If $\Omega = [0,\ 1]$, $M = 4$ and $k = 2$, data are stored as follows.




$$\text{c4n} = (0,\ 1/8,\ 1/4,\ 3/8,\ 1/2,\ 5/8,\ 3/4,\ 7/8,\ 1)$$

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

$$\text{n4db} = (1,\ 9)$$

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

# mesh_FEM1D

The following Matlab code generates an uniform mesh on the domain $\Omega = [a, b]$ in $\mathbb{R}$ with mesh size $h = 1/M$. Also this code returns an index matrix for continuous $k$-th order polynomial approximations.

```
function [c4n, n4e, n4db, ind4e] = mesh_FEM1D(a,b,M,k)
nrNodes = k*M+1;
c4n = linspace(a,b,nrNodes);
n4e = [1:k:(nrNodes-1);(k+1):k:nrNodes];
n4db = [1, nrNodes];
ind4e = repmat(n4e(1,:),k+1,1)+repmat((0:k)',1,M);
end
```

# Basis functions

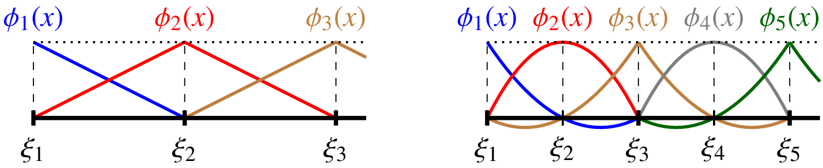
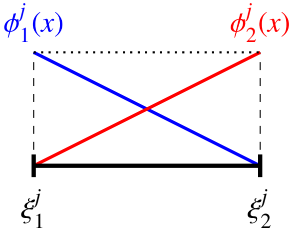



Figure: Global basis functions of $V_h^1$ (left) and $V_h^2$ (right) on an interval.

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

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

# Local basis functions



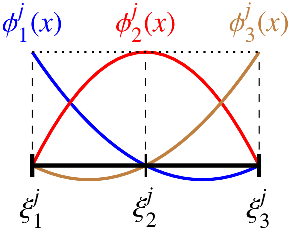




Figure: Local basis functions of $V_h^1$ (left) and $V_h^2$ (right) on the $j$-th element.

$$\phi_i^j(x) = \phi_{k(j-1)+i}(x)\Big|_{I_j}$$

$$\phi_i^j(x) = 0 \quad \text{if } x \in \Omega \setminus I_j$$

$$\sum_{i=1}^{k+1} \phi_i^j(x) = 1 \quad \forall x \in I_j.$$

# Interpolation

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

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

where $f_i = f(\xi_i)$.

# Numerical solutions

- Numerical solution

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

- Local solution

$$u_h(x)\Big|_{I_j} = \sum_{i=1}^{k+1} u_i^j \phi_i^j(x)$$

- Derivatives

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

$$\frac{d}{dx} u_h(x)\Big|_{I_j} = \sum_{i=1}^{k+1} u_i^j \frac{d}{dx} \phi_i^j(x).$$

# Variational formulation

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

$$\sum_{j=1}^{N} u_j \int_\Omega \phi_i'(x)\phi_j'(x)\,dx = \int_\Omega f(x)\phi_i(x)\,dx$$

# Finite element system

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

where

$$(A)_{ij} = \int_\Omega \phi_i'(x)\phi_j'\,dx$$
$$(\boldsymbol{b})_i = \int_\Omega f(x)\phi_i(x)\,dx$$
$$(\boldsymbol{u})_j = u_j.$$

# Finite element system

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

where

$$\begin{aligned}
(A)_{ij} &= \int_\Omega \phi_i'(x)\phi_j'\,dx \\
(\boldsymbol{u})_j &= u_j \\
(M)_{ij} &= \int_\Omega \phi_i(x)\phi_j(x)\,dx \\
(\boldsymbol{f})_j &= f_j.
\end{aligned}$$

# Matrices

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

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

where $(k+1)$-by-$(k+1)$ matrices $A_{I_\ell}$ and $M_{I_\ell}$ are defined as

$$(A_{I_\ell})_{ij} = \int_{I_\ell} \frac{d}{dx}\phi_i^\ell(x)\frac{d}{dx}\phi_j^\ell(x)\,dx$$

$$(M_{I_\ell})_{ij} = \int_{I_\ell} \phi_i^\ell(x)\phi_j^\ell(x)\,dx$$

where $1 \le i,\ j \le k+1$.

# Lemma 1.1

For a given interval $I = [v_1, v_2]$ with barycentric coordinates $\lambda_1$ and $\lambda_2$, it holds for $a,\ b \in \mathbb{N}_0$ that

$$\int_I \lambda_1^a \lambda_2^b \, dx = |I| \frac{a!\, b!}{(a+b+1)!}$$

where $|I| = v_2 - v_1$.

# Differentiation matrix

$$(D)_{ij} = \phi_j'(\xi_i).$$

$$\Rightarrow \qquad \phi_i'(x) = \sum_{j=1}^{N} \phi_i'(\xi_j)\phi_j(x) = (D^t)_i \boldsymbol{\phi}$$

where

$$\begin{aligned}(D^t)_i &= [\phi_i'(\xi_1) \ \cdots \ \phi_i'(\xi_N)] \\ \boldsymbol{\phi} &= [\phi_1(x) \ \cdots \ \phi_N(x)]^t\end{aligned}$$

# Properties of $D$

- $D = \sum_{\ell=1}^{M} D_{I_\ell}, \qquad (D_{I_\ell})_{ij} = \frac{d\phi_j^\ell}{dx}(\xi_i^\ell)$

- $\frac{d}{dx}\phi_i^\ell(x) = \sum_{j=1}^{k+1} \frac{d\phi_i^\ell}{dx}(\xi_j^\ell)\phi_j^\ell(x) = (D_{I_\ell}^t)_i \boldsymbol{\phi}^\ell$

- $u_h'(\xi_m) = (D)_m \boldsymbol{u}$

- $u_h'(\xi_m^\ell) = (D_{I_\ell})_m \boldsymbol{u}^\ell$

# Lemma 1.2

For a given interval $I = [v_1, v_2]$ with barycentric coordinates $\lambda_1$ and $\lambda_2$, it holds for $a,\ b \in \mathbb{N}_0$ that

$$\frac{d}{dx}(\lambda_1^a(x)\lambda_2^b(x)) = r_x \frac{d}{dr}(\widetilde{\lambda}_1^a(r)\widetilde{\lambda}_2^b(r))$$

where $r = \Phi^{-1}(x)$, $r_x = 1/J$ and $J = (v_2 - v_1)/2$.

# Mass matrix

$$(M)_{ij} = \int_I \phi_i(x)\phi_j(x)\, dx = J \int_{I_R} \widetilde{\phi}_i(r)\widetilde{\phi}_j(r)\, dr = J(M_R)_{ij}$$

$$\Rightarrow \qquad M = JM_R$$

# Stiffness matrix

$$(S)_{ij} = \int_I \phi_i'(x)\phi_j'(x)\,dx = \frac{1}{J}(S_R)_{ij}$$

where

$$(S_R)_{ij} = (D_R^t M_R D_R)_{ij}$$

$$\Rightarrow \qquad S = \frac{1}{J} S_R$$

# $P_1$ matrices

- $\widetilde{\phi}_1(r) = \widetilde{\lambda}_1(r), \qquad \widetilde{\phi}_2(r) = \widetilde{\lambda}_2(r)$

- $\widetilde{\phi}_1'(r) = -\frac{1}{2}, \qquad \widetilde{\phi}_2'(r) = \frac{1}{2}$

- $M_R = \frac{1}{3}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$

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

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

# $P_2$ matrices

- $\widetilde{\phi}_1(r) = \widetilde{\lambda}_1(r)(2\widetilde{\lambda}_1(r) - 1),\ \widetilde{\phi}_2(r) = 4\widetilde{\lambda}_1(r)\widetilde{\lambda}_2(r),\ \widetilde{\phi}_3(r) = \widetilde{\lambda}_2(r)(2\widetilde{\lambda}_2(r) - 1)$
- $\widetilde{\phi}_1'(r) = -2\widetilde{\lambda}_1(r) + \dfrac{1}{2}, \quad \widetilde{\phi}_2'(r) = 2\widetilde{\lambda}_1(r) - 2\widetilde{\lambda}_2(r), \quad \widetilde{\phi}_3'(r) = 2\widetilde{\lambda}_2(r) - \dfrac{1}{2}$
- $M_R = \dfrac{1}{15}\begin{pmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{pmatrix}$
- $D_R = \dfrac{1}{2}\begin{pmatrix} -3 & 4 & -1 \\ -1 & 0 & 1 \\ 1 & -4 & 3 \end{pmatrix}$
- $S_R = D_R^t M_R D_R = \dfrac{1}{6}\begin{pmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{pmatrix}$

# MatrixforPoisson_1D

The following Matlab code generates the mass matrix $M_R$, the stiffness matrix $S_R$ and the differentiation matrix $D_R$ for continuous $k$-th order polynomial approximations on the reference interval $I_R$.

```
function [M_R, S_R, D_R] = MatrixforPoisson_1D(k)
if k==1
    M_R = [2 1; 1 2]/3;
    S_R = [1 -1; -1 1]/2;
    D_R = [-1 1; -1 1]/2;
elseif k==2
    M_R = [4 2 -1; 2 16 2; -1 2 4]/15;
    S_R = [7 -8 1; -8 16 -8; 1 -8 7]/6;
    D_R = [-3 4 -1; -1 0 1; 1 -4 3]/2;
elseif ...
end
end
```

# FEMforPoisson_1D

The following Matlab code solves the Poisson problem. In order to use this code, mesh information (c4n, n4e, n4db, ind4e), matrices ($M_R, S_R$), 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_1D(c4n,n4e,n4db,ind4e,M_R,S_R,f,u_D)
A = sparse(length(c4n),length(c4n));
b = zeros(length(c4n),1);
u = b;
for j=1:length(n4e)
    J=(c4n(n4e(2,j))-c4n(n4e(1,j)))/2;
    A(ind4e(:,j),ind4e(:,j)) = A(ind4e(:,j),ind4e(:,j)) + S_R/J;
    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_1D

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

```
function error = ComputeErrorFEM_1D(c4n,ind4e,M_R,D_R,u,Du)
error = 0;
for j=1:size(ind4e,2)
    J=(c4n(ind4e(end,j))-c4n(ind4e(1,j)))/2;
    De=Du(c4n(ind4e(:,j))') - D_R*u(ind4e(:,j))/J;
    error=error+J*De'*M_R*De;
end
error=sqrt(error);
end
```

# main_FEMforPoisson_1D

The following Matlab code solves the Poisson problem by using several matlab codes such as mesh_FEM1D, MatrixforPoisson_1D, FEMforPoisson_1D and ComputeErrorFEM_1D.

```
iter = 10;
a = 0; b = 1; k = 2; M = 2.^(1:iter);
f=@(x) pi^2*sin(pi*x);
u_D=@(x) x*0;
Du=@(x) pi*cos(pi*x);

error=zeros(1,iter);
h=1./M;
for j=1:iter
    [c4n, n4e, n4db, ind4e]=mesh_FEM1D(a,b,M(j),k);
    [M_R, S_R, D_R] = MatrixforPoisson_1D(k);
    u=FEMforPoisson_1D(c4n,n4e,n4db,ind4e,M_R,S_R,f,u_D);
    error(j) = ComputeErrorFEM_1D(c4n,ind4e,M_R,D_R,u,Du);
end
```

# Example

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

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

is the analytical solution to the Poisson problem.

# Convergence history

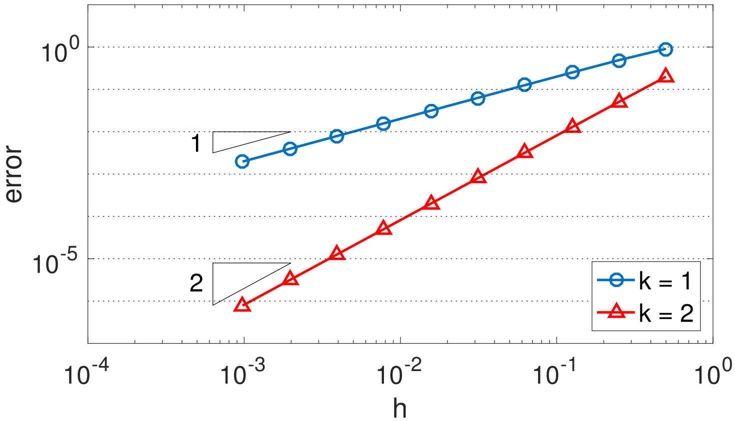



Figure: Convergence history for Example