Finite Element Methods for PDEs

FEM for Poisson's problem in 1D

2024 AJOU-KYUSHU SUMMER SCHOOL ON APPLIED MATHEMATICS

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

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

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

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

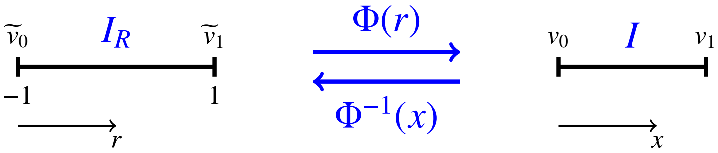


Figure: An affine mapping from the reference interval I_R to an interval I.

Barycentric coordinates

Barycentric coordinates $(\widetilde{\lambda}_0,\ \widetilde{\lambda}_1)$ have the properties

$$\begin{cases} 0 \le \widetilde{\lambda}_i(r) \le 1, & i = 0,\ 1, \\ \widetilde{\lambda}_0(r) + \widetilde{\lambda}_1(r) = 1. \end{cases}$$

Affine Mapping

Let us consider an interval I,

$$I = \{x \mid v_0 \leq x \leq v_1\}.$$

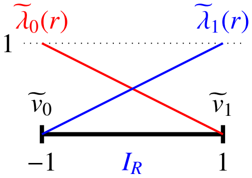
Then we have an affine mapping Φ such that

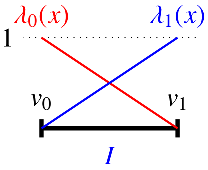
$$\Phi(r) = v_0\widetilde{\lambda}_0(r) + v_1\widetilde{\lambda}_1(r) = x,$$

where $x \in I$.

Properties

1. $\widetilde{\lambda}_0(r) = \dfrac{1-r}{2}, \qquad \widetilde{\lambda}_1(r) = \dfrac{1+r}{2}$
2. $\dfrac{dx}{dr} = \dfrac{v_1 - v_0}{2} \quad \left(\because \;\; x = v_0\widetilde{\lambda}_0(r) + v_1\widetilde{\lambda}_1(r)\right)$
3. $r_x = \dfrac{1}{J} \quad \text{where} \quad J = \dfrac{v_1 - v_0}{2}$
4. $\lambda_0(x) = \dfrac{v_1 - x}{v_1 - v_0}, \qquad \lambda_1(x) = \dfrac{x - v_0}{v_1 - v_0}$
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 $h = \max_{T \in \mathcal{T}_h} \operatorname{diam}(T)$ (e.g. uniform triangulation: $h = 1/M$)
- N : the number of nodes
- M : the number of elements
- I_j : the j-th element in $\mathcal{T}_h$
- ξ_i : the i-th node in $\mathcal{T}_h$
- ξ_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 $0 \le j \le M-1$ and $0 \le i \le k$, respectively.

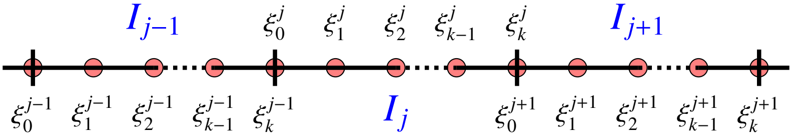
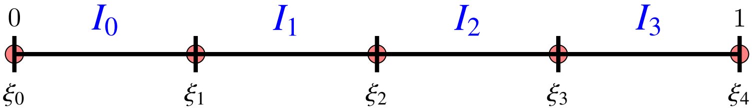


Figure: Nodes in $\mathcal{T}_h$ with k.

Triangulation data (P_1)

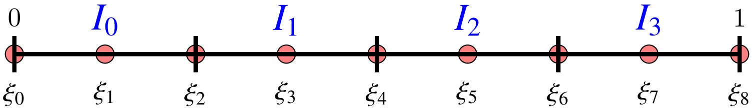
If $\Omega = [0,\ 1]$, $M = 4$ and $k = 1$, data are stored as follows.



$$
\begin{aligned}
\text{c4n} &= [0,\ 1/4,\ 1/2,\ 3/4,\ 1], \\
\text{n4e} &= [[0,1],\ [1,2],\ [2,3],\ [3,4]], \\
\text{ind4e} &= [[0,1],\ [1,2],\ [2,3],\ [3,4]], \\
\text{n4db} &= [0,4]
\end{aligned}
$$

Triangulation data (P_2)

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



$$
\begin{aligned}
\text{c4n} \quad &= [0,\ 1/8,\ 1/4,\ 3/8,\ 1/2,\ 5/8,\ 3/4,\ 7/8,\ 1], \\
\text{n4e} \quad &= [[0,2],\ [2,4],\ [4,6],\ [6,8]], \\
\text{ind4e} \quad &= [[0,1,2],\ [2,3,4],\ [4,5,6],\ [6,7,8]], \\
\text{n4db} \quad &= [0,8]
\end{aligned}
$$

mesh_fem_1d

The following python 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.

```
def mesh_fem_1d(a,b,M,k):
    nrNodes = k*M + 1
    c4n = np.linspace(a, b, nrNodes)
    n4e = np.array([[i*k, (i+1)*k] for i in range(M)])
    n4db = np.array([0, nrNodes-1])
    ind4e = np.array([list(range(i*k, (i+1)*k+1)) for i in range(M)])
    return (c4n,n4e,n4db,ind4e)
```

Basis functions

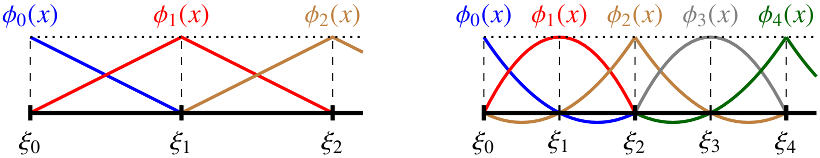


Figure: Global basis functions of V_h^1 (left) and V_h^2 (right) on an interval.

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

$$\sum_{i=0}^{N-1} \phi_i(x) = 1 \qquad \forall x \in \Omega$$

Local basis functions

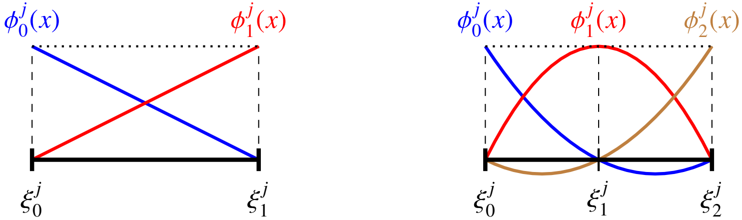


Figure: Local basis functions of V_h^1 (left) and V_h^2 (right) on the j-th element.

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

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

$$\sum_{i=0}^{k} \phi_i^j(x) = 1 \quad \forall x \in I_j.$$

Interpolation

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

$$\mathcal{I} f(x) = \sum_{i=0}^{N-1} f_i \phi_i(x)$$

where $f_i = f(\xi_i)$.

Numerical solutions

- Numerical solution

$$u_h(x) = \sum_{i=0}^{N-1} u_i \phi_i(x)$$

- Local solution

$$u_h(x)\Big|_{I_j} = \sum_{i=0}^{k} u_i^j \phi_i^j(x)$$

- Derivatives

$$\frac{d}{dx} u_h(x) = \sum_{i=0}^{N-1} u_i \frac{d}{dx} \phi_i(x),$$

$$\frac{d}{dx} u_h(x)\Big|_{I_j} = \sum_{i=0}^{k} u_i^j \frac{d}{dx} \phi_i^j(x).$$

Variational formulation

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

$$\sum_{j=0}^{N-1} u_j \int_\Omega \phi_i'(x)\phi_j'(x)\ dx = \int_\Omega f(x)\phi_i(x)\ dx$$

Finite element system

$$\sum_{j=0}^{N-1} u_j \int_\Omega \phi_i'(x)\phi_j'(x)\ dx = \int_\Omega f(x)\phi_i(x)\ dx$$

$$\Rightarrow \quad \underbrace{\begin{bmatrix} \int_\Omega \phi_i'(x)\phi_0'(x)\ dx & \int_\Omega \phi_i'(x)\phi_1'(x)\ dx & \cdots & \int_\Omega \phi_i'(x)\phi_{N-1}'(x)\ dx \end{bmatrix}}_{(\boldsymbol{A})_{i+1}^*} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{\boldsymbol{u}} = \underbrace{\int_\Omega f(x)\phi_i(x)\ dx}_{\boldsymbol{b}_{i+1}}$$

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

Finite element system

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

where

$$\begin{aligned}
(\boldsymbol{A})_{ij} &= \int_{\Omega} \phi'_{i-1}(x)\phi'_{j-1} \; dx \\
(\boldsymbol{b})_i &= \int_{\Omega} f(x)\phi_{i-1}(x) \; dx \\
(\boldsymbol{u})_j &= u_{j-1}.
\end{aligned}$$

Finite element system

In order to compute the load vector, f is replaced by the interpolate function $\mathcal{I}f(x)$ in general.

$$\boldsymbol{b} = \int_\Omega f(x)\phi_i(x)\ dx \approx \int_\Omega \mathcal{I}f(x)\phi_i(x)\ dx = \int_\Omega \phi_i(x)\left(\sum_{i=0}^{N-1} f_j\phi_j(x)\right)\ dx = \sum_{i=0}^{N-1} f_j \int_\Omega \phi_i(x)\phi_j(x)\ dx$$

$$\Rightarrow \quad \underbrace{\begin{bmatrix} \int_\Omega \phi_i(x)\phi_0(x)\ dx & \int_\Omega \phi_i(x)\phi_1(x)\ dx & \cdots & \int_\Omega \phi_i(x)\phi_{N-1}(x)\ dx \end{bmatrix}}_{(\boldsymbol{M})^*_{i+1}} \underbrace{\begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{bmatrix}}_{\boldsymbol{f}} \approx \underbrace{\int_\Omega f(x)\phi_i(x)\ dx}_{\boldsymbol{b}_{i+1}}$$

$$\Rightarrow \qquad \boldsymbol{M}\boldsymbol{f} \approx \boldsymbol{b}$$

Finite element system

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

where

$$(\boldsymbol{A})_{ij} = \int_\Omega \phi'_{i-1}(x)\phi'_{j-1}\ dx$$
$$(\boldsymbol{u})_j = u_{j-1}$$
$$(\boldsymbol{M})_{ij} = \int_\Omega \phi_{i-1}(x)\phi_{j-1}(x)\ dx$$
$$(\boldsymbol{f})_j = f_{j-1}.$$

Matrices

$$\boldsymbol{A} = \sum_{\ell=0}^{M-1} \boldsymbol{A}_{I_\ell},$$

$$\boldsymbol{M} = \sum_{\ell=0}^{M-1} \boldsymbol{M}_{I_\ell}.$$

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

$$(\boldsymbol{A}_{I_\ell})_{ij} = \int_{I_\ell} \frac{d}{dx}\phi_{i-1}^{\ell}(x)\frac{d}{dx}\phi_{j-1}^{\ell}(x)\ dx$$

$$(\boldsymbol{M}_{I_\ell})_{ij} = \int_{I_\ell} \phi_{i-1}^{\ell}(x)\phi_{j-1}^{\ell}(x)\ dx$$

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

Lemma 1.1

For a given interval $I = [v_0, v_1]$ with barycentric coordinates λ_0 and λ_1, it holds for $a,\ b \in \mathbb{N}_0$ that

$$\int_I \lambda_0^a \lambda_1^b \ dx = |I| \frac{a!\ b!}{(a+b+1)!}$$

where $|I| = v_1 - v_0$.

Differentiation matrix

$$(\boldsymbol{D})_{ij} = \phi'_{j-1}(\xi_{i-1})$$

$$\Rightarrow \qquad \phi'_{i-1}(x) = \sum_{j=0}^{N-1} \phi'_{i-1}(\xi_j)\phi_j(x) = (\boldsymbol{D}^t)_i \boldsymbol{\phi}$$

where

$$(\boldsymbol{D}^t)_i = [\phi'_{i-1}(\xi_0) \ \cdots \ \phi'_{i-1}(\xi_{N-1})]$$
$$\boldsymbol{\phi} = [\phi_0(x) \ \cdots \ \phi_{N-1}(x)]^t$$

Properties of D

- $\boldsymbol{D} = \sum_{\ell=0}^{M-1} \boldsymbol{D}_{I_\ell}, \qquad (\boldsymbol{D}_{I_\ell})_{ij} = \frac{d\phi_{j-1}^{\ell}}{dx}(\xi_{i-1}^{\ell})$

- $\frac{d}{dx}\phi_{i-1}^{\ell}(x) = \sum_{j=0}^{k} \frac{d\phi_{i-1}^{\ell}}{dx}(\xi_j^{\ell})\phi_j^{\ell}(x) = (\boldsymbol{D}_{I_\ell}^{t})_i \boldsymbol{\phi}^{\ell}$

- $u_h'(\xi_{m-1}) = (\boldsymbol{D})_m \boldsymbol{u}$

- $u_h'(\xi_{n-1}^{\ell}) = (\boldsymbol{D}_{I_\ell})_n \boldsymbol{u}^{\ell}$

Lemma 1.2

For a given interval $I = [v_0, v_1]$ with barycentric coordinates λ_0 and λ_1, it holds for $a,\ b \in \mathbb{N}_0$ that

$$\frac{d}{dx}(\lambda_0^a(x)\lambda_1^b(x)) = r_x \frac{d}{dr}(\widetilde{\lambda}_0^a(r)\widetilde{\lambda}_1^b(r))$$

where $r = \Phi^{-1}(x)$, $r_x = 1/J$ and $J = (v_1 - v_0)/2$.

Mass matrix

$$(\boldsymbol{M})_{ij} = \int_I \phi_{i-1}(x)\phi_{j-1}(x)\ dx = J \int_{I_R} \widetilde{\phi}_{i-1}(r)\widetilde{\phi}_{j-1}(r)\ dr = J(\boldsymbol{M}_R)_{ij}$$

$$\Rightarrow \qquad \boldsymbol{M} = J\boldsymbol{M}_R$$

Stiffness matrix

$$
\begin{aligned}
(\boldsymbol{S})_{ij} &= \int_I \phi'_{i-1}(x)\phi'_{j-1}(x)\ dx \\
&= \int_I \Big(\sum_{\ell=0}^{k} \frac{d\phi_{i-1}}{dx}(\xi_\ell)\phi_\ell(x) \Big) \Big(\sum_{m=0}^{k} \frac{d\phi_{j-1}}{dx}(\xi_m)\phi_m(x) \Big)\ dx \\
&= J \int_{I_R} \Big(\sum_{\ell=0}^{k} r_x \frac{d\widetilde{\phi}_{i-1}}{dr}(\Phi^{-1}(\xi_\ell))\widetilde{\phi}_\ell(r) \Big) \Big(\sum_{m=0}^{k} r_x \frac{d\widetilde{\phi}_{j-1}}{dr}(\Phi^{-1}(\xi_m))\widetilde{\phi}_m(r) \Big)\ dr \\
&= r_x^2 J(\boldsymbol{S}_R)_{ij} = \frac{1}{J}(\boldsymbol{S}_R)_{ij}
\end{aligned}
$$

Stiffness matrix

$$(\boldsymbol{S})_{ij} = \frac{1}{J}(\boldsymbol{S}_R)_{ij}$$

where

$$\begin{aligned}(\boldsymbol{S}_R)_{ij} &= \int_{I_R} \Big((\boldsymbol{D}_R^t)_i\widetilde{\boldsymbol{\phi}}\Big)\Big((\boldsymbol{D}_R^t)_j\widetilde{\boldsymbol{\phi}}\Big)\, dr \\ &= (\boldsymbol{D}_R^t)_i\boldsymbol{M}_R(\boldsymbol{D}_R)_j \\ &= (\boldsymbol{D}_R^t\boldsymbol{M}_R\boldsymbol{D}_R)_{ij}\end{aligned}$$

$$\Rightarrow \qquad \boldsymbol{S} = \frac{1}{J}\boldsymbol{S}_R$$

P_1 matrices

- $\widetilde{\phi}_0(r) = \widetilde{\lambda}_0(r), \qquad \widetilde{\phi}_1(r) = \widetilde{\lambda}_1(r)$
- $\widetilde{\phi}_0'(r) = -\dfrac{1}{2}, \qquad \widetilde{\phi}_1'(r) = \dfrac{1}{2}$
- $\boldsymbol{M}_R = \dfrac{1}{3}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$
- $\boldsymbol{D}_R = \dfrac{1}{2}\begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix}$
- $\boldsymbol{S}_R = \boldsymbol{D}_R^t \boldsymbol{M}_R \boldsymbol{D}_R = \dfrac{1}{2}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$

P_2 matrices

- $\widetilde{\phi}_0(r) = \widetilde{\lambda}_0(r)(2\widetilde{\lambda}_0(r) - 1), \ \widetilde{\phi}_1(r) = 4\widetilde{\lambda}_0(r)\widetilde{\lambda}_1(r), \ \widetilde{\phi}_2(r) = \widetilde{\lambda}_1(r)(2\widetilde{\lambda}_1(r) - 1)$
- $\widetilde{\phi}'_0(r) = -2\widetilde{\lambda}_0(r) + \frac{1}{2}, \quad \widetilde{\phi}'_1(r) = 2\widetilde{\lambda}_0(r) - 2\widetilde{\lambda}_1(r), \quad \widetilde{\phi}'_2(r) = 2\widetilde{\lambda}_1(r) - \frac{1}{2}$
- $\boldsymbol{M}_R = \frac{1}{15}\begin{pmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{pmatrix}$
- $\boldsymbol{D}_R = \frac{1}{2}\begin{pmatrix} -3 & 4 & -1 \\ -1 & 0 & 1 \\ 1 & -4 & 3 \end{pmatrix}$
- $\boldsymbol{S}_R = \boldsymbol{D}_R^t \boldsymbol{M}_R \boldsymbol{D}_R = \frac{1}{6}\begin{pmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{pmatrix}$

MatrixforPoisson_1D

The following python 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.

```
def get_matrices_1d(k=1):
    if k == 1:
        M_R = np.array([[2, 1],[1, 2]], dtype=np.float64) / 3.
        S_R = np.array([[1, -1],[-1, 1]], dtype=np.float64) / 2.
        D_R = np.array([[-1, 1],[-1, 1]], dtype=np.float64) / 2.
    elif k == 2:
        M_R = np.array([[4, 2, -1],[2, 16, 2],[-1, 2, 4]], dtype=np.float64) / 15.
        S_R = np.array([[7, -8, 1],[-8, 16, -8],[1, -8, 7]], dtype=np.float64) / 6.
        D_R = np.array([[-3, 4, -1],[-1, 0, 1],[1, -4, 3]], dtype=np.float64) / 2.
    elif: ...
    return (M_R, S_R, D_R)
```

Programming

Assignments

1. Add the matrices for the cubic approximations ($k = 3$) in $\boxed{\text{get\_matrices\_1d}}$ and check the convergence rate.
2. Modify $\boxed{\text{fem\_for\_poisson\_1d\_ex2}}$ to solve the Poisson problem with non-homogeneous Dirichlet boundary condition,

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

3. Modify $\boxed{\text{fem\_for\_poisson\_1d\_ex3}}$ to solve the Poisson problem with mixed boundary condition,

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

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