3 Finite Element Methods

3.1 Galerkin Method for 1-D Problem

Consider the two-point boundary value problem:

$$\begin{cases} -u''(x) + \mu u(x) = f(x), & x \in I = (a,b) \\ u(a) = 0, u'(b) = 0. \end{cases} \tag{3.1}$$

Set

$$\begin{aligned} V &\triangleq \left\{ v | v, v \in L^2(a,b), \int_a^b (v^2 + v'^2) dx < +\infty, v(0) = 0 \right\}, \\ a(u,v) &= \int_a^b u'v' dx + \mu \int_a^b uv dx, \\ \langle f, v \rangle &= \int_a^b fv dx. \end{aligned} \tag{3.2}$$

The variational problem to find $u \in V$ such that

$$a(u,v) = \langle f, v \rangle \quad \forall v \in V, \tag{3.3}$$

Let V_h be a subspace of V which is finite dimensional, h stands for a discretization parameter. The Galerkin method of the variation problem is then to find $u_h \in V_h$ such that

$$a(u_h, v_h) = \langle f, v_h \rangle \quad \forall v \in V_h. \tag{3.4}$$

Suppose that $\{\phi_1, \cdots, \phi_N\}$ is a basis for V_h, Then (3.4) is equivalent to

$$a(u_h, \phi_i) = \langle f, \phi_i \rangle, \quad i = 1, \cdots, N. \tag{3.5}$$

Writing u_h in the form

$$u_h = \sum_{j=1}^{N} u_j \phi_j, \tag{3.6}$$

we are led to the system of equations

$$\sum_{j=1}^{N} a(\phi_j, \phi_i) u_j = \langle f, \phi_i \rangle, \quad i = 1, \cdots, N, \tag{3.7}$$

which we can write in the matrix-vector form as

$$A\boldsymbol{u} = \boldsymbol{b} \tag{3.8}$$

where $A_{ij} = a(\phi_j, \phi_i)$, and $b_i = \langle f, \phi_i \rangle$.

$$A\boldsymbol{u} \triangleq \begin{pmatrix} a(\phi_1,\phi_1) & a(\phi_2,\phi_1) & \cdots & a(\phi_n,\phi_1) \\ a(\phi_1,\phi_2) & a(\phi_2,\phi_2) & \cdots & a(\phi_n,\phi_2) \\ \vdots & \vdots & \vdots & \vdots \\ a(\phi_1,\phi_n) & a(\phi_2,\phi_n) & \cdots & a(\phi_n,\phi_n) \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$$

$$\boldsymbol{b} \triangleq \begin{pmatrix} (f,\phi_1) \\ (f,\phi_2) \\ \vdots \\ (f,\phi_n) \end{pmatrix}$$

Mesh splitting, the nodes: $a = x_0 < x_1 < \cdots < x_n = b$

Element: $I_i = [x_{i-1}, x_i]$, $h_i = x_i - x_{i-1}, h = \max_i h_i$

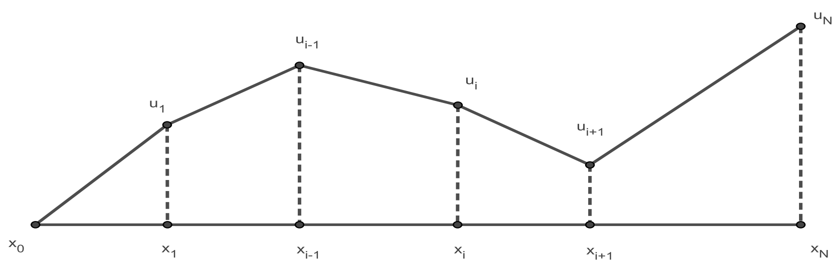
The test function space U_h is composed of piecewise linear functions. Its set of values on the node

$$u_0, u_1, u_2, \cdots, u_n,$$

Linear interpolation formula

$$u_h(x) = \frac{x_i - x}{h_i} u_{i-1} + \frac{x - x_{i-1}}{h_i} u_i, x \in I_i, i = 1, 2, \cdots, n. \tag{3.9}$$

Element shape function Affine transform



$$\xi = \frac{x - x_{i-1}}{h_i},$$

Change I_i to the reference unit $[-1, 1]$,

$$N_{-1}(\xi) = \frac{1-\xi}{2}, \quad N_1(\xi) = \frac{1-\xi}{2}$$

$$\Rightarrow u_h(x) = N_{-1}(\xi) u_{i-1} + N_1(\xi) u_i, \quad x \in I_i$$

Every local element have two element shape function:

$$\Phi_1^{I_i}(x) = \begin{cases} \dfrac{x_i - x}{h_i}, & x \in [x_{i-1}, x_i]; \\ 0, & otherwise. \end{cases}$$

$$\Phi_2^{I_i}(x) = \begin{cases} \dfrac{x - x_{i-1}}{h_i}, & x \in [x_{i-1}, x_i]; \\ 0, & otherwise. \end{cases}$$

Basis function

$$\varphi_1 = \frac{1}{2}(\Phi_2^{I_1} + \Phi_1^{I_2}), \quad \varphi_2 = \frac{1}{2}(\Phi_2^{I_2} + \Phi_1^{I_3}), \quad \cdots$$
$$\varphi_i = \frac{1}{2}(\Phi_2^{I_i} + \Phi_1^{I_{i+1}}), \quad \cdots \quad \varphi_n = \Phi_2^{I_n}.$$

In local unit I_i , element stiffness matrix $K_{2\times 2}^{I_i}$.

$$K_{11}^{I_i} = a(\Phi_1^{I_i}, \Phi_1^{I_i}) = \int_{x_{i-1}}^{x_i} (p\Phi_1^{I_i\prime} \cdot \Phi_1^{I_i\prime 2} + q\Phi_1^{I_i} \cdot \Phi_2^{I_i})dx$$

$$K_{22}^{I_i} = a(\Phi_2^{I_i}, \Phi_2^{I_i})$$

$$K_{12}^{I_i} = a(\Phi_2^{I_i}, \Phi_1^{I_i})$$

$$K_{21}^{I_i} = a(\Phi_1^{I_i}, \Phi_2^{I_i})$$

Global element of stiffness matrix A consist of

$$K_{ij} = \sum_{k=1}^{n} K_{ij}^{I_k}$$

Example 3.1 Consider the two-point boundary value problem:

$$\begin{cases} -u''(x) + \alpha u(x) = f(x), \quad x \in I = (-1, 1) \\ u(-1) = 0, u(1) = 0. \end{cases}$$

Exact solution: $u = x(1-x)\sin(x)$, $f = (4x-2)\cos(x) + (2 + 2x - 2x^2)\sin(x)$.

```
% FEM1D.m
% Finite Element Method
% -u_xx+u=f in (0,1) with boundary condition u(0)=u(1)=0;
% exact : u=x*(1-x)*sin(x)
% RHS: f=(4*x-2).*cos(x)+(2+2*x-2*x^2).*sin(x);
% Thanks to the code from Shuangshuang Li & Qian Tong
clear all
Num=[16 32 64 128 256 512];     % Number of splits
Err=[];  DOF=[];
for j=1:length(Num)
    N=Num(j);    h=1/N;    x=0:h:1;
    % The global node number corresponds to element local node number
    M=[1:N;2:N+1];
    [xv,wv]=jags(2,0,0);  % nodes and weights of gauss quadrature

    K=zeros(N+1);          %  global stiffness matrix
    F=zeros(N+1,1);        %  RHS load vector
    for i=1:N   % loop for each element
        K(M(1,i),M(1,i))=K(M(1,i),M(1,i))
                 +((h/2)*(((1/4)*(2/h)^2+((1-xv)/2).^2)))'*wv;
        K(M(1,i),M(2,i))=K(M(1,i),M(2,i))+((h/2)*((-1/4)*(2/h)^2
                 +((1-xv)/2).*((1+xv)/2)))'*wv;
        K(M(2,i),M(1,i))=K(M(2,i),M(1,i))+((h/2)*((-1/4)*(2/h)^2
                 +((1-xv)/2).*((1+xv)/2)))'*wv;
        K(M(2,i),M(2,i))=K(M(2,i),M(2,i))+((h/2)*(((1/4)*(2/h)^2
                 +((1+xv)/2).^2)))'*wv;

        t=h*xv/2+(x(i+1)+x(i))/2;
        F(M(1,i))=F(M(1,i))+(h/2*((1-xv)/2).*((4*t-2).*cos(t)
                 +(2+2*t-2*t.^2).*sin(t)))'*wv;
        F(M(2,i))=F(M(2,i))+(h/2*((1+xv)/2).*((4*t-2).*cos(t)
                 +(2+2*t-2*t.^2).*sin(t)))'*wv;
    end
    % Dirichlet boundary condition
    K(1,: )=zeros(1,N+1);
    K(:,1)=zeros(1,N+1);
    K(N+1,: )=zeros(1,N+1);
    K(:, N+1)=zeros(1,N+1);
    K(1,1)=1;   K(N+1,N+1)=1;
    F(1)=0;     F(N+1)=0;

    U=K\F;         % numerical solution at the value of the node
    error=max(abs(U'-x.*(1-x).*sin(x)));  % node error
    doff=N+1;  % degrees of freedom, number of unknowns
    Err=[Err, error];
```

```
    DOF=[DOF, doff];
end
plot(log10(DOF),log10(Err),'ro-','MarkerFaceColor','w','LineWidth',1.5),
hold on,
plot(log10(DOF),log10(DOF.^(-2)),'--')
grid on,
xlabel('log_{10}N','fontsize', 16), ylabel('log_{10}Error','fontsize',16),
title('Convergence of Finite Element Method','fontsize',14)
set(gca,'fontsize',14)
```

```
% FEM1DP.m
% FEM for 1D elliptic problem
% -u_xx+u=f in [0,1] with  boundary condition u(0)=u(1)=0;
% exact solution: u=x*(1-x)*sin(x);
% RHS: f=(4*x-2).*cos(x)+(2+2*x-2*x^2).*sin(x)
% Thanks to the code from Shuangshuang Li & Qian Tong
clear all
Num=[16 32 64 128 256 512]
node_Err=[];   L2_Err=[];  H1_Err=[];  DOF=[];
for j=1:length(Num)
    N=Num(j);    h=1/N;    x=0:h:1;
    % The global node number corresponds to element local node number
    M=[1:N;2:N+1];
    [xv,wv]=jags(3,0,0);   % nodes and weights of gauss quadrature
    K=zeros(N+1);          % global stiffness matrix
    F=zeros(N+1,1);       % RHS load vector

    for i=1:N   % loop for each element
        K(M(1,i),M(1,i))=K(M(1,i),M(1,i))
                 +((h/2)*(((1/4)*(2/h)^2+((1-xv)/2).^2)))'*wv;
        K(M(1,i),M(2,i))=K(M(1,i),M(2,i))+((h/2)*((-1/4)*(2/h)^2
                 +((1-xv)/2).*((1+xv)/2)))'*wv;
        K(M(2,i),M(1,i))=K(M(2,i),M(1,i))+((h/2)*((-1/4)*(2/h)^2
                 +((1-xv)/2).*((1+xv)/2)))'*wv;
        K(M(2,i),M(2,i))=K(M(2,i),M(2,i))+((h/2)*(((1/4)*(2/h)^2
                 +((1+xv)/2).^2)))'*wv;

        t=h*xv/2+(x(i+1)+x(i))/2;
        F(M(1,i))=F(M(1,i))+(h/2*((1-xv)/2).*((4*t-2).*cos(t)
                 +(2+2*t-2*t.^2).*sin(t)))'*wv;
        F(M(2,i))=F(M(2,i))+(h/2*((1+xv)/2).*((4*t-2).*cos(t)
                 +(2+2*t-2*t.^2).*sin(t)))'*wv;
    end
    % Handling Dirichlet boundary condition
```

```
    K(1,: )=zeros(1,N+1);
    K(:,1)=zeros(1,N+1);
    K(N+1,: )=zeros(1,N+1);
    K(:, N+1)=zeros(1,N+1);
    K(1,1)=1;   K(N+1,N+1)=1;
    F(1)=0;     F(N+1)=0;

    U=K\F;       % numerical solution at the value of the nodes
    node_error=max(abs(U'-x.*(1-x).*sin(x)));  % node error
    for i=1:N
        tt=h*xv/2+(x(i+1)+x(i))/2;
        % value of finite element solution at Gauss point
        uh=U(i)*(1-xv)/2+U(i+1)*(1+xv)/2;
        % derivative value of finite element solution at Gauss point
        duh=-U(i)/2+U(i+1)/2;
        L2_error(i)=h/2*((tt.*(1-tt).*sin(tt)-uh).^2)'*wv;
        % the square of the L2 error of the i-th interval
        H1_error(i)=h/2*((sin(tt)-2*tt.*sin(tt)...
                    +tt.*(1-tt).*cos(tt)-duh*2/h).^2)'*wv;
        % the square of the H1 semi-norm error of the i-th interval
    end
    node_Err=[node_Err, node_error];
    L2_Err=[L2_Err, sqrt(sum(L2_error))];
    H1_Err=[H1_Err, sqrt(sum(L2_error)+sum(H1_error))];
    doff=N+1;    % degrees of freedom, number of unknowns
    DOF=[DOF, doff];
end
loglog(DOF,node_Err,'r+-','LineWidth',1.5)
hold on
loglog(DOF,L2_Err,'bo-','MarkerFaceColor','w','LineWidth',1.5)
hold on
loglog(DOF,H1_Err,'b*-','LineWidth',1.5)
hold on, grid on
xlabel('log_{10}N','fontsize', 16), ylabel('log_{10}Error','fontsize',16),
title('Convergence of Finite Difference Method','fontsize',14)
set(gca,'fontsize',14)

for i=1:length(Num)-1    % calculating of convergence order
    node_order(i)=log(node_Err(i)/node_Err(i+1))/(log(DOF(i)/DOF(i+1)));
    L2_order(i)=log(L2_Err(i)/L2_Err(i+1))/(log(DOF(i)/DOF(i+1)));
    H1_order(i)=log(H1_Err(i)/H1_Err(i+1))/(log(DOF(i)/DOF(i+1)));
end
node_order
L2_order
H1_order
```