Desai Manish
21101022

Assignment -3

1. Strong form :

(S) $\Rightarrow$ { Given $f : \Omega \to R$ $\quad g : \Gamma_g \to \mathbb{R}$

$h : \Gamma_h \to \mathbb{R}$

Find $u : \Omega \to R$ such that

$q_{i,i} = f$ on Ω

$u = g$ on Γ_g

$-q_i n_i = h$ on Γ_h

So, In the above,

Let us assume $f = 0$

~~Γ_{g_1} let~~

$u = 0$ on Γ_{g_1}

$u = 1$ on Γ_{g_2}

~~we get~~

Given ~~g~~

we get,

Given $f=0$

~~Find u on Ω~~

Find $u : \Omega \to R$ such that

$q_{i,i} = 0$ on Ω

$u = 0$ on Γ_{g_1}

$u = 1$ on Γ_{g_2}

$-q_i n_i = 0$ there is no Γ_n

Weak form:

Assuming u is the sol^n to the strong form

$\Rightarrow$ $u \in S$ and choose a $w \in V$

such that

$S = \{u \mid u \in H^1, u = 0$ for Γ_{g_1} & $u = 1$ on $\Gamma_{g_2}\}$

$V = \{w \mid w \in H^1$ $w = 0$ on Γ_g

$$\int_{\Omega} w\,(q_{i,i})\,d\Omega = 0$$

$$= \int_{\Omega} -w_{,i}\, q_i\, d\Omega + \int_{\Gamma} w\, q_i\, d\Gamma = 0$$

$$\Gamma = \Gamma_g = \Gamma_{g_1} \cup \Gamma_{g_2}$$

$$= \int_{\Omega} -w_{,i}\, q_i\, d\Omega + 0 = 0$$

$$\Rightarrow -\int_{\Omega} w_{,i}\, q_i\, d\Omega = 0$$

$$\boxed{\int_{\Omega} w_{,i}\, q_i\, d\Omega = 0}$$

Let $a(w, u) = \int w_{,i}\, q_i\, d\Omega$

Then

$$a(w, u) = 0$$

Galerkin form:

Let $S^h \subset S \quad V^h \subset V$

$u^h = v^h + g^h$

where $v^h \in V^h$

$\Leftrightarrow a(w^h, v^h + g^h) = 0$

$a(w^h, v^h) = -a(w^h, g^h)$

Let η be the set of all nodes

$\{1, 2, \ldots n_{np}\}$

$n_{np} \rightarrow$ total no of nodal points

$n_{el} \longrightarrow$ total no of elements that belongs to Γ_g

Γ_g nodes doesn't have an eqn

Total nodes with eqn

$\Rightarrow \eta - \eta_g$

n_{eq} be size of $\eta - \eta_g$

· Basis functions : N_A's

$$w^h(\underset{\sim}{x}) = \sum_{A \in \eta - \eta_g} c_A N_A(x)$$

$$g^h(x) = \sum_{B \in \eta - \eta_g} d_B N_B(x)$$

$$a\left(\sum_{A \in \eta - \eta_g} c_A N_A(x), \sum_{B \in \eta - \eta_g} d_B N_B\right)$$

$$= - a\left(\sum_{A \in \eta - \eta_g} c_A N_A(x), \sum_{B \in \eta_g} g_B N_B(x)\right)$$

Using Bilinearity of $a(\cdot,)$ we get

$$\sum_{A \in \eta - \eta_g} C_A \left(\sum_{B \in \eta - \eta_g} a(N_A, N_B) d_B + \sum_{B \in \eta_g} a(N_A, N_B) g_B \right)$$

$$= 0$$

$$\sum_{B \in \eta - \eta_g} a(N_A, N_B) d_B + \sum_{B \in \eta_g} (N_A, N_B) g_B = 0$$

$$\boxed{K_{PQ} = a(N_A, N_B)}$$

$$\boxed{F_P = -\sum_{B \in \eta_g} a(N_A, N_B) g_B}$$

$$\boxed{K_{PQ} d = F_P}$$

$$l_a^{nen-1}(\xi_1) = \frac{\prod_{\substack{b=1 \\ b\neq a}}^{nen} (\xi_1 - \xi_{1b})}{\prod_{\substack{b=1 \\ b=a}}^{nen} (\xi_{1a} - \xi_{1b})}$$

Property : $l_a(\xi_{1b}) = \delta_{ab}$

$\cancel{N_a(\xi_1, \eta)}$.

$$l_1^1 = \frac{(\xi_1 - \xi_2)}{(\xi_1 - \xi_{12})} = \frac{\xi_1 - 1}{-2} = \frac{1-\xi}{2}$$

$$l_2^1 = \frac{\xi_1 + 1}{2} + \frac{1}{2}(\xi_1 + 1$$

$$N_1 = l_1^1(\xi_1)\, l_1^1(\eta) = \frac{1}{4}(1-\xi_1)(1-\eta)$$

$$N_2 = l_2^1(\xi_1)\, l_1^1(\eta) = \frac{1}{4}(1+\xi_1)(1-\eta)$$

$$N_3 = l_2^1(\xi_1)\, l_2^1(\eta) = \frac{1}{4}(1+\xi_1)(1+\eta)$$

$$N_4 = l_1^1(\xi_1)\, l_2^1(\eta) = \frac{1}{4}(1-\xi_1)(1+\eta)$$

```
%% initializing the parameters
clear all;
clc;
close all;
N=9;
a=(N-1)/4+1;
b=3*(N-1)/4+1;
nel=(N-1)^2-((N-1)/2)^2;
Xcords=ones(N,1)*[0:(N-1)];
Ycords=[(N-1):-1:0]'*ones(1,N);

%% writing the equation numbers to the mesh
dvars=zeros(N,N);
k=1; % initialzing the variables
for i=N:-1:1
    for j=1:N
        if((i>a&&i<b&&j>a&&j<b))
        else
            dvars(i,j)=k; % placing the values at
corresponding location
            k=k+1;
        end
    end
end
% number of elemental points nnp
nnp=max(max(dvars));
%% IEN matrix
ien=zeros(4,nel); % initializing the variables
PM=zeros(2,nel); % initializing the variables
k=1; % initializing the variables
for i=N:-1:2
    for j=1:N-1
         if((i>a&&i<=b&&j>=a&&j<b)) % the central part
of the mesh is not counted using this condition
         else
             ien(1,k)=dvars(i,j); % placing the
variables
             PM(1,k)=i;
             PM(2,k)=j;
             k=k+1;
         end
```

```
    end
end
k=1;
for i=(N-1):-1:1
    for j=2:N
         if((i>=a&&i<b&&j>a&&j<=b))
         else
             ien(3,k)=dvars(i,j);
             k=k+1;
         end
    end
end
k=1;
for i=N:-1:2
    for j=2:N
         if((i>a&&i<=b&&j>a&&j<=b))
         else
             ien(2,k)=dvars(i,j);
             k=k+1;
         end
    end
end
k=1;
for i=(N-1):-1:1
    for j=1:(N-1)
         if((i>=a&&i<b&&j>a&&j<=b))
         else
             ien(4,k)=dvars(i,j);
             k=k+1;
         end
    end
end

%% ID array
ID=zeros(1,nnp);
dum1=zeros(N,N);
k=1;
l=1;
for i=N:-1:1
    for j=1:N
        if (not(i>=3&&j>=3&&i<=7&&j<=7))
```

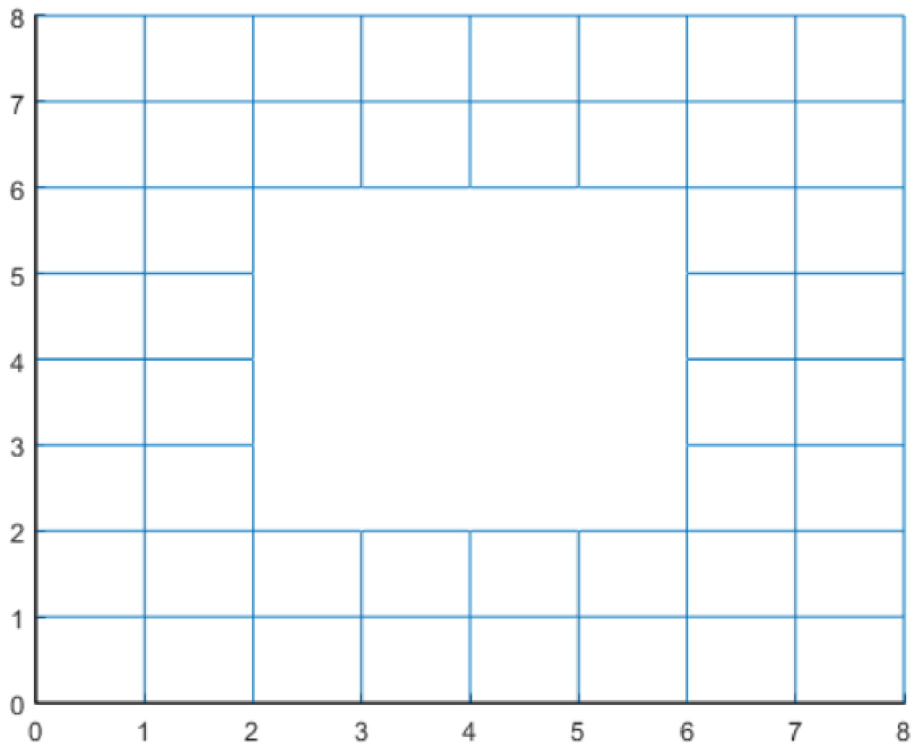
```
                if(i>=2&&j>=2&&i<=8&&j<=8)
                    ID(l)=k;
                    dum1(i,j)=k;
                    k=k+1;
                end

            end
        if (not(i>=4&&j>=4&&j<=6&&i<=6))
        l=l+1;
        end
        end

end

neq=max(ID);
K=zeros(neq,neq);
F=zeros(neq,1);
%% LM array
LM=zeros(4,nel);
for i=1:4
    for j=1:nel
        LM(i,j)=ID(ien(i,j));
    end
end

%% Mesh
hold on;
rectangle('Position',[0 0 N-1 N-1])
for i=1:(N-1)
    line([0, N-1],[i i]);
    line([i i],[0 N-1])
end
for i=a:b-2
    line([a-1,b-1],[i i],'Color','white');
    line([i i],[a-1,b-1],'Color','white');
end
%% Using bilinear shape functions in 2D, write the
element level matrices and vector

kele=zeros(4,4);
f=zeros(4,1);
```

```
c2=-1/2;
b1=1/2;
j=c2*b1;
syms eta neu x y
n1=@(eta,neu) (1-eta)*(1-neu)/4; %N1 bilinear function
n2=@(eta,neu) (1+eta)*(1-neu)/4; %N2 bilinear function
n3=@(eta,neu) (1+eta)*(1+neu)/4; %N3 bilinear function
n4=@(eta,neu) (1-eta)*(1+neu)/4; %N4 bilinear function
n1e=@(neu) neu/4 - 1/4;N1n=@(eta) eta/4 - 1/4; %
differentiating n1
n2e=@(neu) -neu/4 + 1/4;N2n=@(eta) -eta/4 - 1/4; %
differentiating n2
n3e=@(neu) neu/4 + 1/4;N3n=@(eta) +eta/4 + 1/4; %
differentiating n3
n4e=@(neu) -neu/4 - 1/4;N4n=@(eta) -eta/4 + 1/4; %
differentiating n4
force11= @(eta,neu)
(n1e(neu).^2/b1.^2+N1n(eta).^2/c2.^2)*j;
kele(1,1)=integral2( force11,-1,1,-1,1);
force22= @(eta,neu)
(n2e(neu).^2/b1.^2+N2n(eta).^2/c2.^2)*j;
kele(2,2)=integral2( force22,-1,1,-1,1);
force33= @(eta,neu)
(n3e(neu).^2/b1.^2+N3n(eta).^2/c2.^2)*j;
kele(3,3)=integral2( force33,-1,1,-1,1);
force44= @(eta,neu)
(n4e(neu).^2/b1.^2+N4n(eta).^2/c2.^2)*j;
kele(4,4)=integral2( force44,-1,1,-1,1);
force12= @(eta,neu)
(n1e(neu).*n2e(neu)/b1.^2+N1n(eta).*N2n(eta)/c2.^2)*j;
kele(1,2)=integral2( force12,-1,1,-1,1);
kele(2,1)=kele(1,2);
force13= @(eta,neu)
(n1e(neu).*n3e(neu)/b1.^2+N1n(eta).*N3n(eta)/c2.^2)*j;
kele(1,3)=integral2( force13,-1,1,-1,1);
kele(3,1)=kele(1,3);
force14= @(eta,neu)
(n1e(neu).*n4e(neu)/b1.^2+N1n(eta).*N4n(eta)/c2.^2)*j;
kele(1,4)=integral2( force14,-1,1,-1,1);
kele(4,1)=kele(1,4);
```

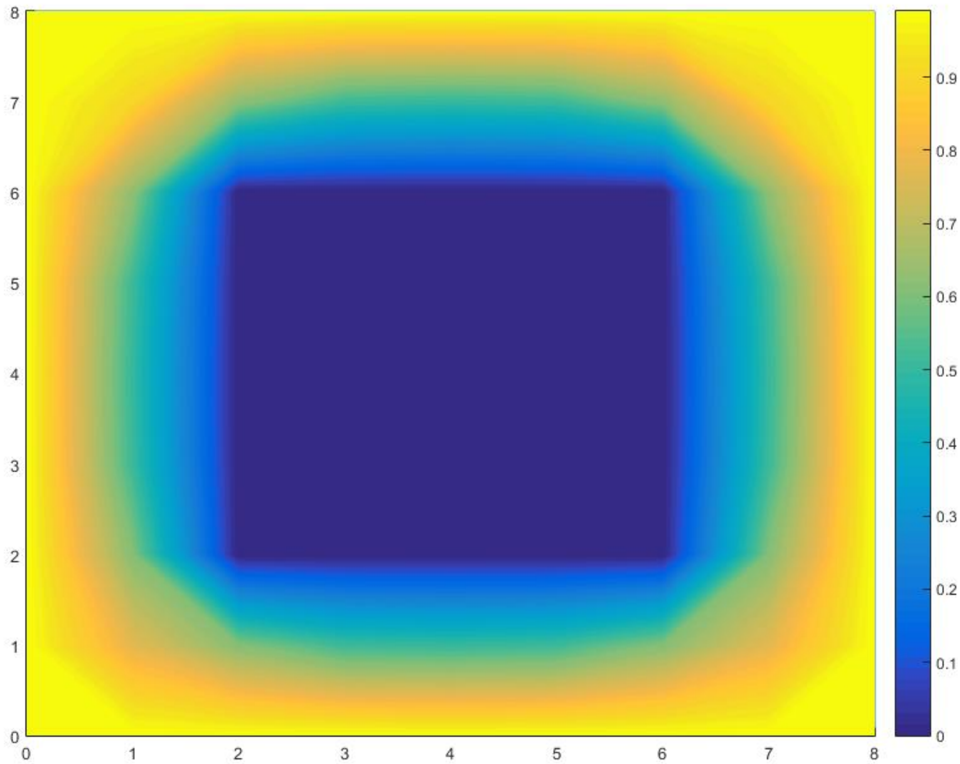
```
force23= @(eta,neu)
(n2e(neu).*n3e(neu)/b1.^2+N2n(eta).*N3n(eta)/c2.^2)*j;
kele(2,3)=integral2( force23,-1,1,-1,1);
kele(3,2)=kele(2,3);
force24= @(eta,neu)
(n2e(neu).*n4e(neu)/b1.^2+N2n(eta).*N4n(eta)/c2.^2)*j;
kele(2,4)=integral2( force24,-1,1,-1,1);
kele(4,2)=kele(2,4);
force34= @(eta,neu)
(n3e(neu).*n4e(neu)/b1.^2+N3n(eta).*N4n(eta)/c2.^2)*j;
kele(3,4)=integral2( force34,-1,1,-1,1);
kele(4,3)=kele(3,4);
%% finite element program to assemble the global
stiffness matrix and force vector

for i=1:nel
if (i==1)
    ke1=kele;
end
y=PM(1,i);
x=PM(2,i);
for z=1:4
    m=0;
    if (x==1)
        m=m-kele(z,1)-kele(z,4);
    end
    if (x==(N-1))
        m=m-kele(z,2)-kele(z,3);
    end
    if (y==2)
        m=m-kele(z,3)-kele(z,4);
    end
    if (y==N)
        m=m-kele(z,1)-kele(z,2);
    end
    if(i==1&&y==N)
        m=m+kele(z,1);
    end
    if(i==(N-1)&&y==N)
        m=m+kele(z,2);
    end
```

```
    if(i==1&&y==2)
        m=m+kele(z,4);
    end
    if(i==(N-1)&&y==2)
        m=m+kele(z,3);
    end
    f(z)=m;
end
for r=1:4
    for s=1:4
        if (LM(r,i)&&LM(s,i))

K(ID(ien(r,i)),LM(s,i))=K(ID(ien(r,i)),LM(s,i))+kele(r,
s);
        end
    end
    if (LM(r,i))
        F(ID(ien(r,i)))=F(ID(ien(r,i)))+f(r);
    end
end
end


d=K^-1*(F);
 %% updating u at every node

 u=zeros(N,N);
 for i=1:N
     for j=1:N
         if (i==1||i==9||j==1||j==9) % at the outer
boundary the condition is applied
             u(i,j)=1;
         end
     end
 end
 k=1;
for i=(N-1):-1:2
    for j=2:(N-1)
        if(not(i>=a&&j>=a&&i<=b&&j<=b)) % other than
the middle hole other values are updated
        u(i,j)=(d(k));
```

```
        k=k+1;
        end
    end
end

contourf(Xcords,Ycords,u,100,'LineColor','non')
```



Mesh of the Domain



Contour Plot of the Linear Heat conduction equation