AE 618

Shrikant Dalmia
180739

April 2021

Choice of Shape

The choice of shape is square plate with a square hole. The temperature of the outer boundary is 0 while it is 1 at the boundary of the hole.The dimension of the outer boundary is 100*100 and the dimension of the inner boundary is 60*60 and their centre is coincided.

Assumption

- The heat conductivity is constant.
- There is no heat source.

1. The strong form of the problem

Let u be Temperature field
and q be the heat flux that can be given as K ∇u
Where K is the heat conductivity.

Given
f: $\Omega \rightarrow \mathbb{R}$ is the given heat source i.e 0
g1: $\Gamma_{g1} \rightarrow \mathbb{R}$ be the outer boundary
g1: $\Gamma_{g2} \rightarrow \mathbb{R}$ be the inner boundary
find u ($\Omega \rightarrow \mathbb{R}$) such that
∇q =0 on Ω
u(Γ_{g2})=1,u(Γ_{g1})=0

2. The weak form of the problem

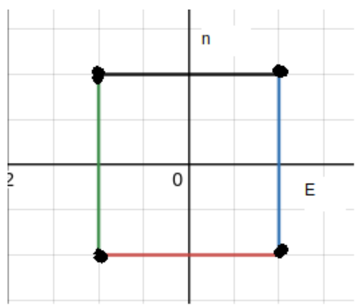
given f,g1,g2 as above
we define two spaces as following:

S: {u $\|$u $\in H^1$,u=1 on Γ_{g2},u=0 on Γ_{g1}}
V: {w $\|$w $\in H^1$,w=0 on Γ_{g2},w=0 on Γ_{g1}}

Find u$\in$S such that for all w$\in$V
a(u,w)=(w,f) i.e
a(u,w)=0

3.Using bi-linear shape functions in 2D,the element level matrices and force vector.

For a bi-linear shape function we will use a 4 noded element which can be represented by (ξ,η) as shown below



We can define shape functions $N_a \in$ V such that

$$x(\xi,\eta) = \sum_{a=1}^{n_{en}} N_a(\xi,\eta) {x_a}^e$$

$$y(\xi,\eta) = \sum_{a=1}^{n_{en}} N_a(\xi,\eta) {y_a}^e$$

$$N_a(\xi,\eta) = \frac{(1+\xi_a\xi)(1+\eta_a\eta)}{4}$$

where ξ_a,η_a are given as below,

a	ξ_a	η_a
1	-1	-1
2	1	-1
3	1	1
4	-1	1

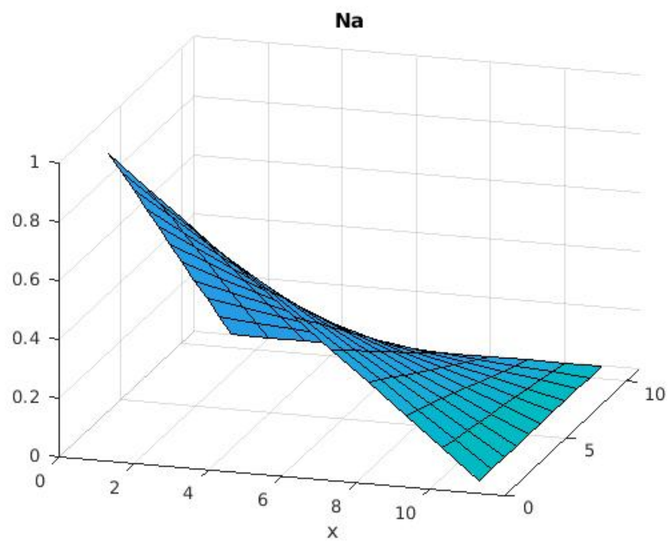


Figure 1: Na_1

Thus the element matrix can be given by

$$K^e_{ab} = \int_{\Omega^e} (\nabla N_a)^T K (\nabla N_b)\, d\Omega$$

And the force vector can be given by

$$f^e_a = \int_{\Omega^e} N_a f \, d\Omega - \sum_{a=1}^{n_{en}} K^e_{ab} * g^e_b$$

we can simplify both the above equation and write as

$$K^e_{ab} = K \int\int < N_{a,x} N_{a,y} >< N_{b,x} N_{b,y} >^t \, dxdy$$

we can write the following expression as

$$< N_{a,x} N_{a,y} >=< N_{a,\xi} N_{a,\eta} > \begin{bmatrix} \xi_{,x} & \eta_{,x} \\ \xi_{,y} & \eta_{,y} \end{bmatrix}$$

we know that,

$$x(\xi,\eta)_{,\xi} = \sum_{a=1}^{n_{en}} N_a(\xi,\eta)_{\xi} {x_a}^e$$

$$x(\xi,\eta)_{,\eta} = \sum_{a=1}^{n_{en}} N_a(\xi,\eta)_{,\eta} {x_a}^e$$

$$y(\xi,\eta)_{,\xi} = \sum_{a=1}^{n_{en}} N_a(\xi,\eta)_{,\xi} {y_a}^e$$

$$y(\xi,\eta)_{,\eta} = \sum_{a=1}^{n_{en}} N_a(\xi,\eta)_{,\eta} {y_a}^e$$

Thus we get,

$$K^e_{ab} = K\int_{-1}^{1}\int_{-1}^{1} \frac{< N_{a,\xi} N_{a,\eta} > \begin{bmatrix} y(\xi,\eta)_{,\eta} & -x(\xi,\eta)_{,\eta} \\ -y(\xi,\eta)_{,\xi} & x(\xi,\eta)_{,\xi} \end{bmatrix} [< N_{b,\xi} N_{b,\eta} > \begin{bmatrix} y(\xi,\eta)_{,\eta} & -x(\xi,\eta)_{,\eta} \\ -y(\xi,\eta)_{,\xi} & x(\xi,\eta)_{,\xi} \end{bmatrix}]^t}{J} d\xi d\eta$$

Where J is the Jacobian,
can be given by J= $x(\xi,\eta)_{,\xi} * y(\xi,\eta)_{,\eta} - x(\xi,\eta)_{,\eta} * x(\xi,\eta)_{,\eta}$
We can write the above expression as,

$$K^e_{ab} = \int_{-1}^{1}\int_{-1}^{1} G(\xi,\eta) d\xi d\eta$$

where $G(\xi,\eta) = K \frac{< N_{a,\xi} N_{a,\eta} > \begin{bmatrix} y(\xi,\eta)_{,\eta} & -x(\xi,\eta)_{,\eta} \\ -y(\xi,\eta)_{,\xi} & x(\xi,\eta)_{,\xi} \end{bmatrix} [< N_{b,\xi} N_{b,\eta} > \begin{bmatrix} y(\xi,\eta)_{,\eta} & -x(\xi,\eta)_{,\eta} \\ -y(\xi,\eta)_{,\xi} & x(\xi,\eta)_{,\xi} \end{bmatrix}]^t}{J}$

Now using Gauss quadrature rules we can apply numerical integration we will get the following:

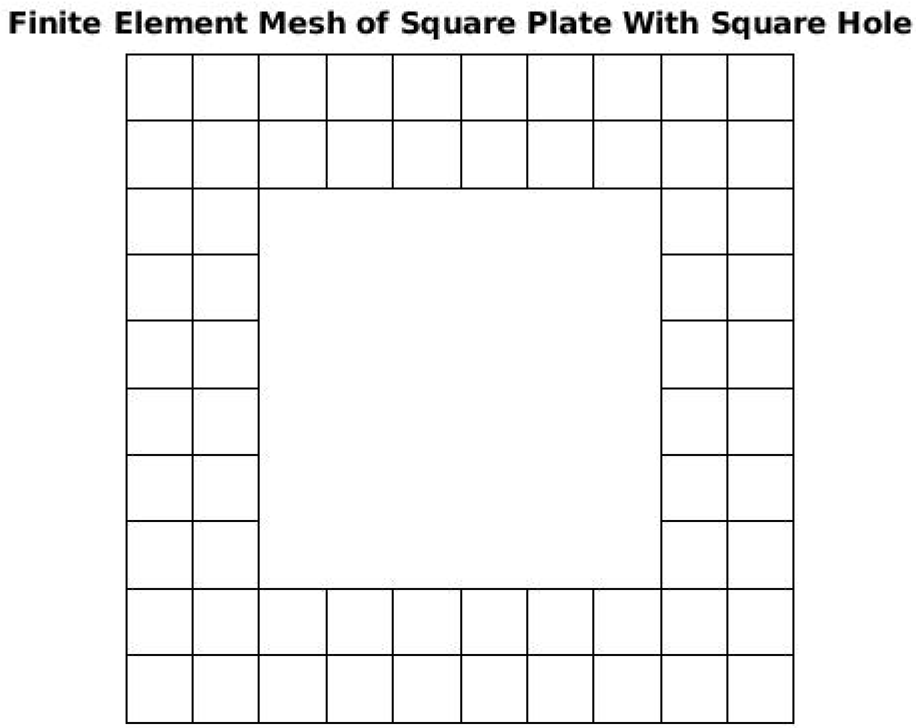
$$K^e_{ab} = G(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}) + G(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}) + G(\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}) + G(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$$

and Since f of the give problem is zero the f^e_a can be given as follows,

$$f^e_a = -\sum_{a=1}^{n_{en}} K^e_{ab} * g^e_b$$

4.Generate a finite element mesh

I have generated a mesh where each element is square and the figure is shown below.




```
clear;
clc;
L=100;
B=100;
divx=L/10;
divy=B/10;
nel=divx*divy;
npx=divx+1;
npy=divy+1;
nnode = npx*npy ;
nx=linspace(0,L,npx);
ny=linspace(0,B,npy);

nnel = 4 ;
[xx, yy] = meshgrid(nx,ny) ;
coordinates = [xx(:) yy(:)] ;
NodeNo = zeros(npx,npy) ;

```

```
for i=1:npx*npy
        if(coordinates(i,1)>L/4 &&  coordinates(i,1)<(3*L)/4 && coordinates(i,2)>L/4 && coordinates(i,2)<(3*L)/4)
            coordinates(i,1)=-1;
            coordinates(i,2)=-1;

        end
end
nodes = zeros(nel,nnel) ; %IEN array

count=1;
for i=1:npx
    for j=1:npy
        if(coordinates((i-1)*npx+j,1)== -1 && coordinates((i-1)*npx+j,2)==(-1) && i>1 &&j>1)
             NodeNo(i,j)=-1;



        else
             NodeNo(i,j)=count;
             count=count+1;
        end

    end
end

for i= size(coordinates(:,1)):-1:1

        if(coordinates(i,1)==-1||coordinates(i,2)==-1   )
            coordinates((i),:)=[];
        end

end

count=1;
    for i=1:npx-1
        for j=1:npy-1
            if(NodeNo(i,j)~=-1 && NodeNo(i,j+1)~=-1  && NodeNo(i+1,j)~=-1 && NodeNo(i+1,j+1)~=-1 )
                 nodes(count,1)=NodeNo(i,j);
                 nodes(count,2)=NodeNo(i+1,j);
                 nodes(count,3)=NodeNo(i+1,j+1);
                 nodes(count,4)=NodeNo(i,j+1);
                 count=count+1;
```

```
                end
            end
        end

        nel=count-1;
        ien = nodes(1:nel,1:4);
        X = zeros(nnel,nel) ;
        Y = zeros(nnel,nel) ;
% Extract X,Y coordinates for the (iel)-th element
    for iel = 1:nel
        X(:,iel) = coordinates(nodes(iel,:),1) ;
        Y(:,iel) = coordinates(nodes(iel,:),2) ;
    end
% Figure
fh = figure ;
set(fh,'name','Preprocessing for FEA','numbertitle','off','color','w') ;
patch(X,Y,'w')
title('Finite Element Mesh of Square Plate With Square Hole') ;
axis([0. L*1.01 0. B*1.01])
axis off ;
if L==B
    axis equal ;
end
```

5.A shape function subroutine

A shape function subroutine descries Na and its derivatives and jacobian J at a point (ξ,η).
As described earlier Na can be given as

$$N_a(\xi,\eta) = \frac{(1+\xi_a\xi)(1+\eta_a\eta)}{4}$$

$$J = x(\xi,\eta)_{,\xi} * y(\xi,\eta)_{,\eta} - x(\xi,\eta)_{,\eta} * x(\xi,\eta)_{,\eta}$$

$$< N_{a,x} N_{a,y} > = < N_{a,\xi} N_{a,\eta} > \begin{bmatrix} \xi_{,x} & \eta_{,x} \\ \xi_{,y} & \eta_{,y} \end{bmatrix}$$

and using the above expression we can calculate higher order derivatives if required.
Code for shape function subroutine and elemental k matrix

```
for ea=[-1,1]
    for na=[-1,1]
```

```
        N{index}=symfun(((1+ea*l)*(1+na*m))/4, [l m]);
        index=index+1;
    end
end

Nx = cell(nnel,1);
Ny = cell(nnel,1);
for i=1:nnel
    Nx{i} = diff(N{i},l);
    Ny{i} = diff(N{i},m);
end

for i=1:nel
    k=zeros(nnel);
    f=zeros(nnel,1);
    sz=size(K);
    F=zeros(sz(1),1);
    jacobian = cell(2);
    jacobian{1,1} = symfun(Nx{1}*coordinates(ien(i,1),1)+Nx{2}*coordinates(ien(i,2),1)+Nx{3}*coordinates(ien(i,3),1)+Nx{4}*coordinates(ien(i,4),1), [l m]);
    jacobian{1,2} = symfun(Ny{1}*coordinates(ien(i,1),1)+Ny{2}*coordinates(ien(i,2),1)+Ny{3}*coordinates(ien(i,3),1)+Ny{4}*coordinates(ien(i,4),1), [l m]);
    jacobian{2,1} = symfun(Nx{1}*coordinates(ien(i,1),2)+Nx{2}*coordinates(ien(i,2),2)+Nx{3}*coordinates(ien(i,3),2)+Nx{4}*coordinates(ien(i,4),2), [l m]);
    jacobian{2,2} = symfun(Ny{1}*coordinates(ien(i,1),2)+Ny{2}*coordinates(ien(i,2),2)+Ny{3}*coordinates(ien(i,3),2)+Ny{4}*coordinates(ien(i,4),2), [l m]);
    det=jacobian{1,1}*jacobian{2,2}-jacobian{1,2}*jacobian{2,1};

    for p=1:nnel
        for q=1:nnel
            g = (Nx{p}*jacobian{2,2}-Ny{p}*jacobian{2,1})*(Nx{q}*jacobian{2,2}-Ny{q}*jacobian{2,1})+(Ny{p}*jacobian{1,1}-Nx{p}*jacobian{1,2})*(Ny{q}*jacobian{1,1}-Nx{q}*jacobian{1,2});%Nx{p}*Nx{q} + Ny{p}*Ny{q};
            k(p,q) = g((-1/sqrt(3)),(-1/sqrt(3)))/ det((-1/sqrt(3)), (-1/sqrt(3))) + g((1/sqrt(3)),(-1/sqrt(3)))/det((1/sqrt(3)), (-1/sqrt(3))) + g((-1/sqrt(3)),(1/sqrt(3)))/det((-1/sqrt(3)),(1/sqrt(3))) + g((1/sqrt(3)),(1/sqrt(3)))/det((1/sqrt(3)), (1/sqrt(3)))
```

```
;
        end
    end
    for p=1:nnel
        for q=1:nnel
            f(p)=f(p)-k(p,q)*lmf(ien(i,q));
        end
    end
end
```

6.Program to assemble the global stiffness matrix and force vector

K= $A_{e=1}^{nel} k_{ab}^{e}$

F= $A_{e=1}^{nel} f_{b}^{e}$
we can easily Assemble the elemental k matrix using lm array.

```
K=zeros(p);
F=zeros(p,1);

for i=1:nel

   for p=1:nnel
        for q=1:nnel
            if(lm(i,p)==0 || lm(i,q)==0)
                continue;
            else
                K(lm(i,p),lm(i,q)) = K(lm(i,p),lm(i,q)) + k(i,p,q);
                %k(i,:,:) is the elemental k matrix for element i
            end
        end

   end
    for x=1:nnel
        if(lm(i,x)~=0)
            F(lm(i,x)) = F(lm(i,x))+f(i,x);
            %f(i,:) is the elemental f matrix for element i
        end
    end
end
```

7.The solution

In matrix form we can write

$$[K][d] = [F]$$

so our solution can be given by

$$[d] = [K]^{-1}[F]$$

```
d = inv(K)*F;
```

8.Plot

Now we have temperature at all nodes which we have defined earlier.we can use the plot3 function in MATLAB and obtain the graph

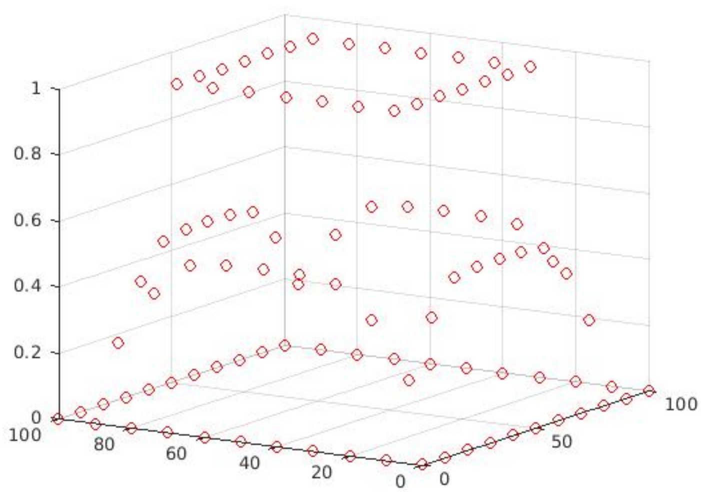


Figure 2: Plot

I have also run this code for Higher Mesh Resolution the plot is given below

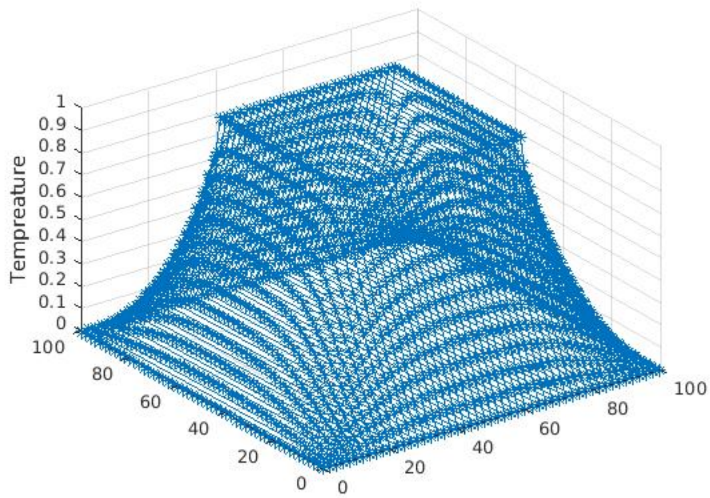


Figure 3: Plot with Higher Mesh Resolution where the [L=2]

9.Discussion

- According to conduction theory the temperature should decrease from inner boundary to outer boundary,Hence Theory agrees with our FEM model.
- If we have good computational processors we can increase the number of nodes in our mesh and can come closer to the actual solution.
- Also,We can see that our steady state temperature is independent of heat conductivity (constant) which also agrees with the conduction theory.