AE 675A : Introduction to Finite Element Methods

Assignment Report

2D – Steady State Heat Conduction Problem

AMAL M S

22101262

Department of Aerospace Engineering, IIT Kanpur

CONTENTS

- Question
- MATLAB Code
- Output
- Inference

QUESTION

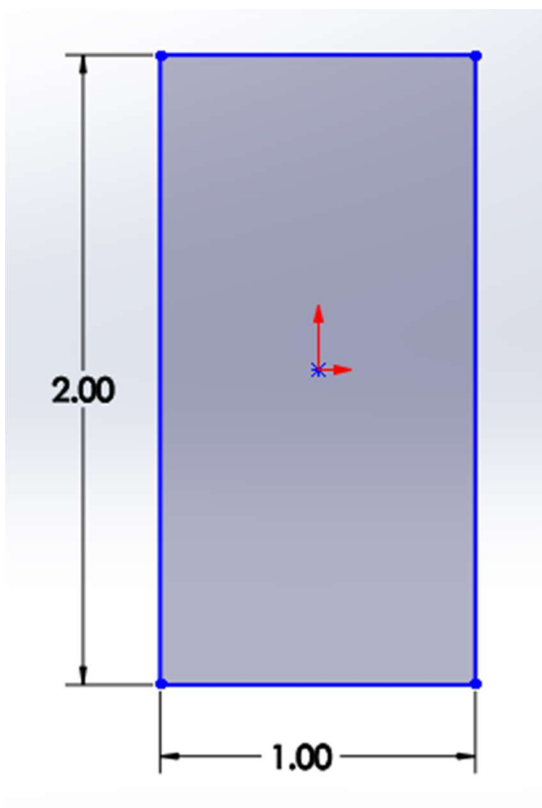
Given below is the figure of a rectangular 2D geometry of length 2 units and breadth 1 units.

$k_{11} = k_{22} = k_{12} = k_{21} = 1;$

$f = 10;$

The temperature across all the four boundaries is zero.

Plot the temperature distribution in the geometry after solving using FEM.



MATLAB CODE

```
% Settings
clear all
format short


% Define variables
syms eta tsi


% Define the parameters
Lx = 1; % Length of the rectangle
Ly = 2; % Width of the rectangle
k = 1; % Thermal conductivity
f = 10; % Heat generation rate
Tleft = 0; % Boundary condition on the left edge
Tright = 0; % Boundary condition on the right edge
Tbottom = 0; % Boundary condition on the bottom edge
Ttop = 0; % Boundary condition on the top edge
nelx = input('Enter the number of elements along x direction: '); % Number of
elements in the x-direction
nely = input('Enter the number of elements along y direction: '); % Number of
elements in the y-direction


% Create the mesh
x = linspace(0, Lx, nelx+1);
y = linspace(0, Ly, nely+1);
[X,Y] = meshgrid(x,y);
x = reshape(X',[],1);
y = reshape(Y',[],1);
nv = (nelx+1)*(nely+1); % Number of vertices
nt = 2*nelx*nely; % Number of triangles
iv = (reshape(1:nv,nelx+1,nely+1))'; % Index of each vertex
i_tri = zeros(nt,3); % vertex of each triangle row-wise
count = 0;
for i = 1:nely
    for j = 1:nelx
        n1 = iv(i,j);
        n2 = iv(i,j+1);
        n3 = iv(i+1,j+1);
        n4 = iv(i+1,j);
        i_tri(count+1,:) = [n1 n2 n3];
```

```
        i_tri(count+2,:) = [n3 n4 n1];
        count = count + 2;
    end
end


% Stiffness matrix and load vector
K = zeros(nv,nv);
F = zeros(nv,1);
    N1 = 1-eta-tsi;
    N2 = tsi;
    N3 = eta;
    N = [N1 N2 N3];
for i = 1:nt
    % Extract the vertices of the current element
    n1 = i_tri(i,1);
    n2 = i_tri(i,2);
    n3 = i_tri(i,3);
    x1 = x(n1);
    x2 = x(n2);
    x3 = x(n3);
    y1 = y(n1);
    y2 = y(n2);
    y3 = y(n3);

    X_master = (x1*N(1))+(x2*N(2))+(x3*N(3));
    Y_master = (y1*N(1))+(y2*N(2))+(y3*N(3));
    J = [diff(X_master,tsi) diff(X_master,eta); diff(Y_master,tsi)
diff(Y_master,eta)];
    det_J = det(J);
    JJ = inv(J);
    dtsi_dx = JJ(1,1);
    dtsi_dy = JJ(1,2);
    deta_dx = JJ(2,1);
    deta_dy = JJ(2,2);
    dN1_dx = diff(N1,tsi)*dtsi_dx + diff(N1,eta)*deta_dx;
    dN1_dy = diff(N1,tsi)*dtsi_dy + diff(N1,eta)*deta_dy;
    dN2_dx = diff(N2,tsi)*dtsi_dx + diff(N2,eta)*deta_dx;
    dN2_dy = diff(N2,tsi)*dtsi_dy + diff(N2,eta)*deta_dy;
    dN3_dx = diff(N3,tsi)*dtsi_dx + diff(N3,eta)*deta_dx;
    dN3_dy = diff(N3,tsi)*dtsi_dy + diff(N3,eta)*deta_dy;
    dN_dx = [dN1_dx dN2_dx dN3_dx];
    dN_dy = [dN1_dy dN2_dy dN3_dy];


    % Calculate the element stiffness matrix and load vector
    for i = 1:3
        for j = 1:3
```

```
            Klocal(i,j) = 
int(int((dN_dx(i)*dN_dx(j)+dN_dy(i)*dN_dy(j))*det_J,tsi,0,1-eta),eta,0,1);
        end
        Flocal(i) = (int(int((N(i))*(f)*det_J,tsi,0,1-eta),eta,0,1));
    end
    % Assemble the element stiffness matrix and load vector into the global 
matrix and vector
    indices = [n1 n2 n3];
    K(indices,indices) = K(indices,indices) + Klocal;
    F(indices) = F(indices) + Flocal';
end
K;
F;
% Apply the boundary conditions
fixed_nodes = [];
for i = 1:length(x)
    if x(i) == 0 % Left edge
        fixed_nodes = [fixed_nodes i];
    elseif x(i) == Lx % Right edge
        fixed_nodes = [fixed_nodes i];
    elseif y(i) == 0 % Bottom edge
        fixed_nodes = [fixed_nodes i];
    elseif y(i) == Ly % Bottom edge
        fixed_nodes = [fixed_nodes i];
    end
end


for i = fixed_nodes
    K(:,i) = 0;
    K(i,:) = 0;
    K(i,i) = 1;
    F(i) = 0;
end
K
F
T = K\F


% Plot the temperature distribution
trisurf(i_tri, x, y, T);
xlabel('x');
ylabel('y');
zlabel('Temperature');
title('Temperature Distribution');
```

OUTPUT

```
Enter the number of elements along x direction: 3
Enter the number of elements along y direction: 5
```

K = 24×24

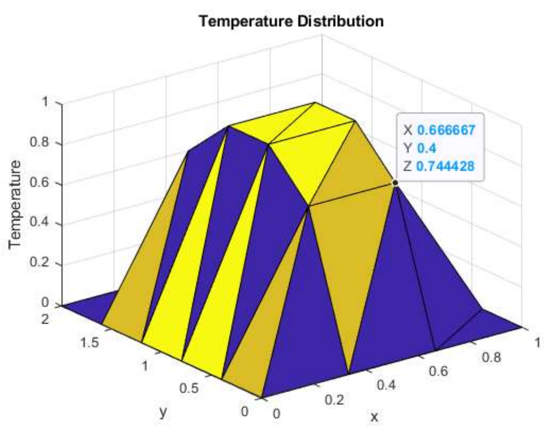
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1.0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1.0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1.0000	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1.0000	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1.0000	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	4.0667	-1.2000	0	0	-0.8333	-0.0000	0	0	0	0	0
7	0	0	0	0	0	-1.2000	4.0667	0	0	0	-0.8333	0	0	0	0	0
8	0	0	0	0	0	0	0	1.0000	0	0	0	0	0	0	0	0

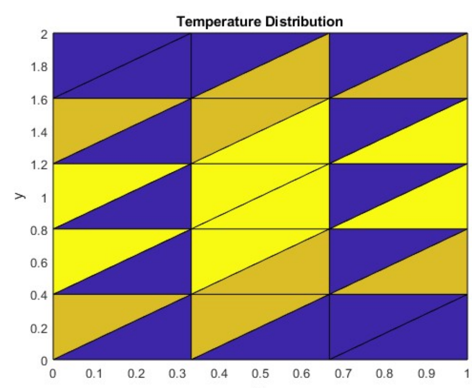
F = 24×1

	1
1	0
2	0
3	0
4	0
5	0
6	1.3333
7	1.3333
8	0

T = 24×1

	1
1	0
2	0
3	0
4	0
5	0
6	0.7444
7	0.7444
8	0





INFERENCE

- It can be seen from the graph that the temperature distribution is zero at the boundaries and reaches peak value at around the mid-point of the geometry.
- We expect a symmetric distribution of temperature as the boundary temperatures were all zero. But the deviation from symmetrical distribution can be attributed to the rectangular shape and majorly the way the mesh was made.
- On trying to make the mesh finer, a distribution as shown below is obtained. This is a more symmetric distribution.