Project # 2

Solution of Heat Conduction Equation in Finite Elements

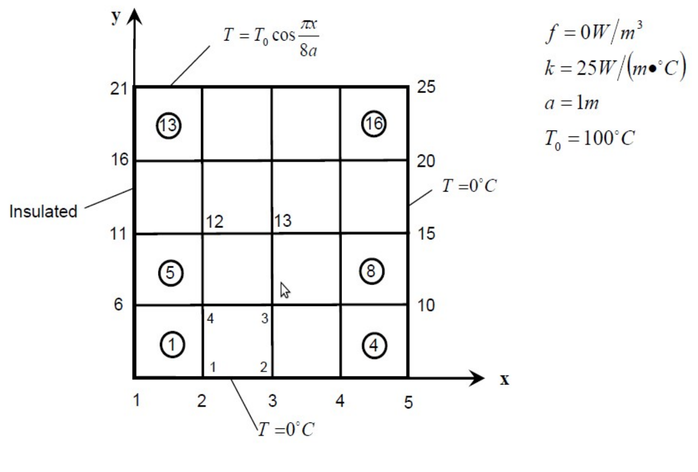
Muhammad Baqui[1]
Computational and Data Sciences Department,George Mason University,Fairfax,VA,22030.

In this report, a steady state heat conduction equation is solved with finite element method to obtain temperature distribution along a two dimensional rectangular domain. The boundaries of the domain is set to specific temperature values. Linear rectangular and triangular elements are used for finite elements. An unstructured grid composed of triangles are also used to solve the same problem. The code is implemented in MATLAB. To solve the resulting linear system, MATLAB linear equation solver is used. The results are shown in the form of contour plots. The solution when compared with actual solution showed same result.

I. Introduction

The steady state heat conduction equation can be used to obtain temperature profile for different geometries. The steady state heat conduction equation is a second order partial differential equation. It involves the term thermal conductivity (k). In this problem the value of thermal conductivity is defined as 25 W/(mC). For solving the problem, a rectangular domain is selected. The sides of the domain is maintained at specified conditions. The left boundary is insulated, the right and bottom boundaries are kept at zero degree C. The top boundary is maintained to follow a $T_0 \cos\frac{\pi x}{8a}$ profile. The domain is shown in Figure [1]. The steady state heat conduction equation is shown as:

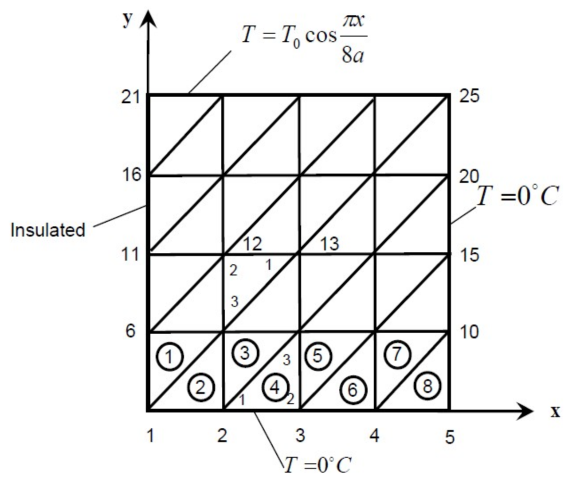
$$-k\nabla^2 T(x,y) = f(x,y)$$



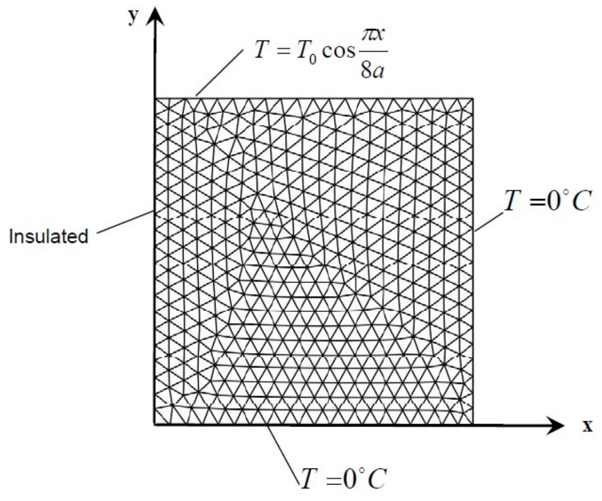
(a)

[1] Graduate Student, Computational and Data Sciences Department, George Mason University, Fairfax,VA

The value of T_0 is set at 100 deg C. The domain is rectangular with sides as 4a for this problem a is set at a=1 m. The domain do not have internal heat generation so f = 0 W/m^3 . Galerkin Finite Element method is used to obtain temperature profile of the domain using rectangular (Figure [1a]) , triangular (Figure [1b]) and unstructured triangular (Figure [1c]) grids. For Figure [1c] grid a separate data file is provided where element connectivities and node coordinates are specified. In this case element k matrices are calculated whereas for all other cases constant element matrices are used as discussed in section 2.2.



(b)



(c)

Figure 1. Different grids used in the finite element method.

II. Problem Formulation

In this section, step by step description is provided of how the finite element model is formulated, what element matrices are used how are they obtained. Also, the section provides specific nodal equations for triangular and rectangular elements.

2. 1. Formulation of finite element model:

The steady state heat conduction equation is:

$$-k\nabla^2 T(x,y)=f(x,y)$$

the equation can be expanded to

$$-k\left[\frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial y}\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)\right]=f$$

In the first step doing integration by parts and applying weight function w we get

$$0=-\int_{\Omega e} k\,w\left[\frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial y}\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)-f\right]dx\,dy$$

$$0=\int_{\Omega e} k\left[\frac{\partial w}{\partial x}\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)+\frac{\partial w}{\partial y}\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)-wf\right]dx\,dy-\oint k\,w\left[n_x\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)+n_y\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)\right]ds$$

The second term of this equation will be zero as $\frac{\partial T}{\partial n}=0$ along the boundary, the secondary variable q$_n$ will be zero. In this case $q_n=\left(n_x\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)+n_y\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)\right)$. So it will vanish along ds. Eventually we get.

$$0=\int_{\Omega e} k\left[\frac{\partial w}{\partial x}\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)+\frac{\partial w}{\partial y}\left(\frac{\partial T}{\partial x}+\frac{\partial T}{\partial y}\right)-wf\right]dx\,dy$$

Lagrange family of interpolation functions, T approximated over elements as

$$T(x,y)\approx T_h^e(x,y)=\sum_{j=1}^{n} T_j^e\psi_j^e(x,y)$$

substituting the value of T in weak formulation leads to

$$0=\int_{\Omega e}\left[k\frac{\partial w}{\partial x}\left(\sum_{j=1}^{n} T_j^e\frac{\partial w}{\partial x}+\sum_{j=1}^{n} T_j^e\frac{\partial \psi_j}{\partial y}\right)+k\frac{\partial w}{\partial y}\left(\sum_{j=1}^{n} T_j^e\frac{\partial \psi_j}{\partial x}+\sum_{j=1}^{n} T_j^e\frac{\partial \psi_j}{\partial y}\right)-kf\,w\right]dx\,dy$$

now replace w as $w=\psi_i^e$ we get

$$0=\int_{\Omega e}\left[k\frac{\partial \psi_i}{\partial x}\left(\sum_{j=1}^{n} T_j^e\frac{\partial \psi_j}{\partial x}+\sum_{j=1}^{n} T_j^e\frac{\partial \psi_j}{\partial y}\right)+k\frac{\partial \psi_i}{\partial y}\left(\sum_{j=1}^{n} T_j^e\frac{\partial \psi_j}{\partial x}+\sum_{j=1}^{n} T_j^e\frac{\partial \psi_j}{\partial y}\right)-kf\,\psi_i\right]dx\,dy$$

$$\sum_{j=1}^{n}\left\{\int_{\Omega e} k\frac{\partial \psi_i}{\partial x}\left(\frac{\partial \psi_j}{\partial x}+\frac{\partial \psi_j}{\partial y}\right)+k\frac{\partial \psi_i}{\partial y}\left(\frac{\partial \psi_j}{\partial x}+\frac{\partial \psi_j}{\partial y}\right)\right\}dx\,dy=\int_{\Omega e} kf\,\psi_i\,dx\,dy$$

this equation can be expressed as

$\sum_{j=1}^{n} K_{ij}^e T_j^e=f_i^e(i=1,2,3.\dots n)$ where

$$K_{ij}^e=\left\{\int_{\Omega e} k\frac{\partial \psi_i}{\partial x}\left(\frac{\partial \psi_j}{\partial x}+\frac{\partial \psi_j}{\partial y}\right)+k\frac{\partial \psi_i}{\partial y}\left(\frac{\partial \psi_j}{\partial x}+\frac{\partial \psi_j}{\partial y}\right)\right\}T_j^e\,dx\,dy \quad \text{and} \quad f_i^e=\int_{\Omega e} kf\,\psi_i\,dx\,dy$$

so in matrix form: $[K^e][T^e]=[f^e]$

2.2 Element coefficient matrices

in this project, linear right triangular and rectangular elements are considered. The element matrix for a generalized rectangular element is [1] :

$$[K^e]=\frac{k_e}{6}\begin{bmatrix} 2(a^2+b^2) & a^2-2b^2 & -(a^2+b^2) & b^2-2a^2 \\ a^2-2b^2 & 2(a^2+b^2) & b^2-2a^2 & -(a^2+b^2) \\ -(a^2+b^2) & b^2-2a^2 & 2(a^2+b^2) & a^2-2b^2 \\ b^2-2a^2 & -(a^2+b^2) & a^2-2b^2 & 2(a^2+b^2) \end{bmatrix}$$

when a/b=1 then the matrix takes the form as :

$$[K^e]=\frac{k_e}{6}\begin{bmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & 2 & -1 & 4 \end{bmatrix}$$

this formulation has been used in current project for the construction of global coefficient matrix.

similarly a generalized representation of triangular element matrix is

$$[K^e]=\frac{k_e}{2}\begin{bmatrix} b^2 & -b^2 & 0 \\ -b^2 & (a^2+b^2) & -a^2 \\ 0 & -a^2 & a^2 \end{bmatrix}$$

when a=b the matrix takes the form of

$$[K^e]=\frac{k_e}{2}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

this formulation has been used in current project for the construction of global coefficient matrix.

2.3 Finite Element Equations for specific nodes:

For rectangular elements the finite element equation for node 13, 16 and 19 are as follows:
node 13:

$$k_{31}^{6}T_7+(k_{32}^{6}+k_{41}^{7})T_8+k_{42}^{7}T_9+(k_{21}^{10}+k_{34}^{6})T_{12}+(k_{22}^{10}+k_{11}^{11}+k_{33}^{6}+k_{44}^{7})T_{13}$$
$$+(k_{12}^{11}+k_{43}^{7})T_{14}+k_{24}^{10}T_{17}+(k_{23}^{10}+k_{14}^{11})T_{18}+k_{13}^{11}T_{19}=0$$

node 16:

$$k_{14}^{13}T_{21}+(k_{11}^{13}+k_{44}^{9})T_{16}+k_{41}^{9}T_{11}+(k_{12}^{13}+k_{43}^{9})T_{17}+k_{42}^{9}T_{12}+k_{13}^{13}T_{22}=0$$

node 19:

$$k_{31}^{11}T_{13}+(k_{32}^{11}+k_{41}^{11})T_{14}+k_{42}^{12}T_{15}+(k_{21}^{15}+k_{34}^{11})T_{18}+(k_{22}^{15}+k_{11}^{16}+k_{33}^{11}+k_{44}^{12})T_{19}+(k_{43}^{12}+k_{12}^{16})T_{20}+k_{24}^{25}T_{23}$$
$$+(k_{23}^{15}+k_{14}^{16})T_{24}+k_{13}^{16}T_{25}=0$$

for triangular elements specific nodal equations for nodes 13, 16, 19 are:

node 13:

$$(k_{13}^{11}+k_{31}^{12})T_7+(k_{32}^{12}+k_{23}^{13})T_8+(k_{12}^{11}+k_{21}^{20})T_{12}+(k_{22}^{20}+k_{33}^{21}+k_{11}^{22}+k_{11}^{11}+k_{33}^{12}+k_{22}^{13})T_{13}$$
$$+(k_{12}^{22}+k_{21}^{13})T_{14}+(k_{23}^{20}+k_{32}^{21})T_{18}+(k_{13}^{22}+k_{31}^{21})T_{19}=0$$

node 16:

$$k_{23}^{17}T_{11}+(k_{21}^{17}+k_{12}^{26})T_{17}+(k_{22}^{17}+k_{11}^{26}+k_{33}^{25})T_{16}+(k_{31}^{25}+k_{13}^{26})T_{22}+k_{32}^{25}T_{21}=0$$

node 19:

$$(k_{13}^{21}+k_{31}^{22})T_{13}+(k_{32}^{22}+k_{23}^{23})T_{14}+(k_{21}^{30}+k_{12}^{21})T_{18}+(k_{11}^{21}+k_{33}^{22}+k_{22}^{23}+k_{22}^{30}+k_{33}^{31}+k_{11}^{32})T_{19}$$
$$+(k_{12}^{32}+k_{21}^{23})T_{20}+(k_{23}^{30}+k_{32}^{31})T_{24}+(k_{31}^{31}+k_{13}^{32})T_{25}=0$$

2.4 Primary and secondary variables on boundary nodes:

the right and bottom boundary is set to T =0 so primary variables associated with these nodes are set to zero :

$$T_1=T_2=T_3=T_4=T_5=T_{10}=T_{15}=T_{20}=0$$

the top boundary nodal values follow the equation $T(x)=T_0\cos\frac{\pi x}{8}$ so the following node values can be specified :

$$T_{21}=T_0 \quad T_{22}=T_0\cos\frac{\pi}{8} \quad T_{23}=T_0\cos\frac{\pi x}{4} \quad T_{24}=T_0\cos\frac{3\pi}{8}$$

the left boundary is insulated so secondary variables at that boundary will be zero

$$F_6=F_{11}=F_{16}=0$$

for the balance of internal heat flow along the domain, the following secondary variables will be zero:

$$F_7=F_8=F_9=F_{12}=F_{13}=F_{14}=F_{17}=F_{18}=F_{19}=0$$

Hence, the unknown primary variables are :

$$T_6,T_7,T_8,T_9,T_{11},T_{12},T_{13},T_{14},T_{16},T_{17,}T_{18},T_{19}$$

unknown secondary variables are:

$$F_1,F_{2,}F_{3,}F_4,F_5,F_{10},F_{15},F_{20},F_{21,}F_{22},F_{23},F_{24},F_{25}$$

III. Code Implementation

Four different MATLAB scripts are written for four requirements of the project. Linear triangular and rectangular elements are implemented in “triangular.m” and “rectangular.m” file. In these scripts the node coordinates and connectivities are defined manually inside the script in the form of matrix. Once the connectivities are defined, the element matrices are directly incorporated in the code as mentioned in the previous section. After the element matrix is defined the global K matrix is formed by three nested loops that iterates over each element in the outer stage and in the inner stage they loop over each nodes of the elements. For rectangular elements we have 4 nodes per element so the variables irow and icol loop from 1 to 4. For triangular they loop from 1 to 3. Once the global K matrix is formed the boundary nodes are identified in a row vector. Two vectors are written, the “zeroNodeIndex” are nodes with zero values i.e. the right and bottom boundary. The vector “topBoundaryNodeIndex” and “topBoundaryNodeValues” are for top boundary nodes and nodal values respectively. The right hand vector or the force vector is formed as the same row number of global K matrix. The vector is populated with top boundary nodes and their corresponding values. Once boundary nodes are identified, the row and columns corresponding to these boundary nodes are first deleted for nodes with zero values from global K matrix and force vector. After the global matrix and the force vector is formed the MATLAB inbuilt function is used to find the solution of the linear system. The values are stored in felSol vector. This vector does not have the boundary node values so a loop over each nodes are done to put the boundary nodal values in the solution. The final solution is put in a separate array as “solutionVector”. The values of this vector is used in the contour plot. The unstructured grid solution is formed very similar to the structured rectangular/triangular grid solution. However, in this case processing of the input file is needed to identify the element connectivities and node coordinates. For this purpose loops are used that go over the rows of the input file to first form the connectivities and then to get the node coordinates. The connectivity loops go from 1 to number of elements (nelem) and node loops run from 1 to number of nodes (nnode). For the unstructured grid the formulation of element matrix is also different. In this case we need the values of α, β,γ to form the element matrix. These values are calculated in each element using the following formula:

$$\alpha_i = x_i y_k - x_i y_j$$
$$\beta_i = y_j - y_k$$
$$\gamma_i = -(x_j - x_k)$$
$$i \neq j \neq k\, ; i\, ,\ j\, , k \text{ permute by natural order.}$$

using α, β,γ and area of triangle, the element k matrix is formed by the following formula:

$$K_{ij}^{e} = \frac{k}{4A_e}\left(\beta_i^e \beta_j^e + \gamma_i^e \gamma_j^e\right)$$

once the element matrix is formed the rest of the code is similar to the structured grid where global K matrix is formed, boundary nodes are identified, force vector is formed and using MATLAB inbuilt linear equation solver, the solution vector is calculated.

IV. Results and Analysis

The actual solution of the partial differential equation is given as

$$T(x,y)=\frac{T_0 \sinh\frac{\pi y}{8a}\cos\frac{\pi x}{8a}}{\sinh\frac{\pi}{2}}$$

The temperature contour obtained from the actual solution can be seen from Figure [2]. From Figure [2] it can be seen that the temperature is highest in the top left corner and because of the other boundaries is being insulated or maintained at 0 deg C, the temperature distribution gradually decreases from the top left corner. From the finite element solution similar temperature distribution should be obtained. If similar patterns are obtained, that would ensure the correctness of the numerical approach taken.

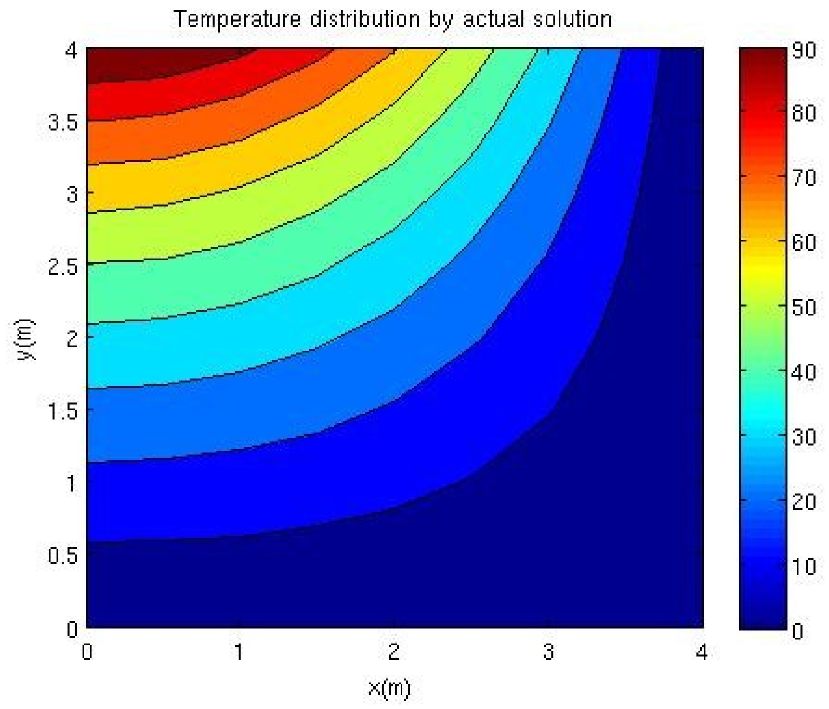


Figure 2. Temperature distribution obtained by actual solution

The temperature distribution for rectangular elements with finite element method can be seen in Figure [3]. From the figure it can be seen the the distribution profile is very similar to the actual solution. The levels are a little discontinuous that is because of the number of elements for rectangular elements are only 16. Similar profile is seen for triangular elements shown in Figure [4]. The number of elements are significantly increased for unstructured grid. As a result we see a temperature distribution Figure [5] identical to the actual solution in Figure [2]. To verify that the unstructured grid produces the same result as of the structured triangular or rectangular the unstructured code was run with structured triangular elements node and connectivities. The data file is stored as 'tridomain.data'. Similar results are obtained. The temperature distribution is shown in Figure [6].

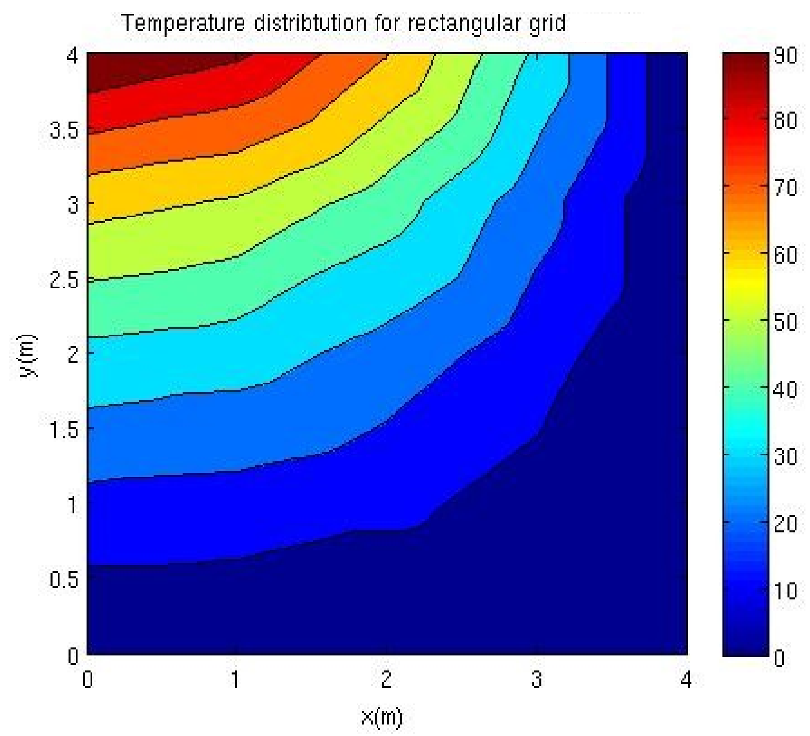


Figure 3. Temperature distribution obtained from rectangular finite elements

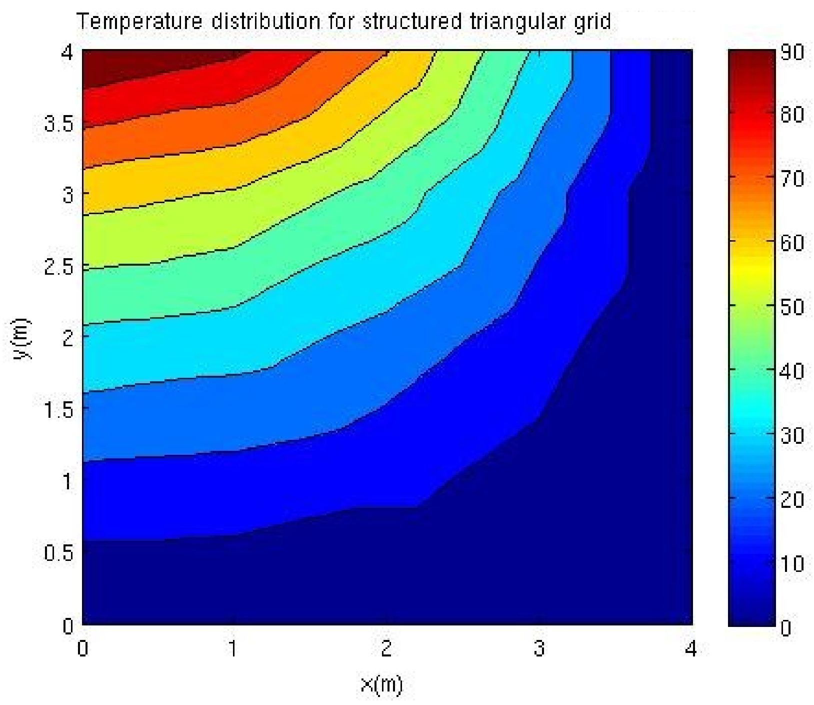


Figure 4. Temperature distribution obtained from right triangular finite elements

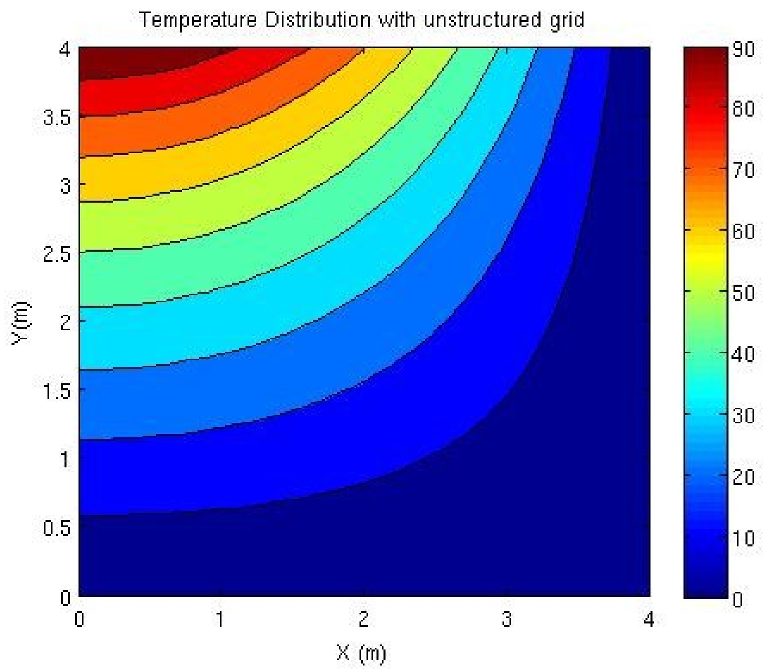


Figure 5. Temperature distribution obtained from unstructured grid finite elements

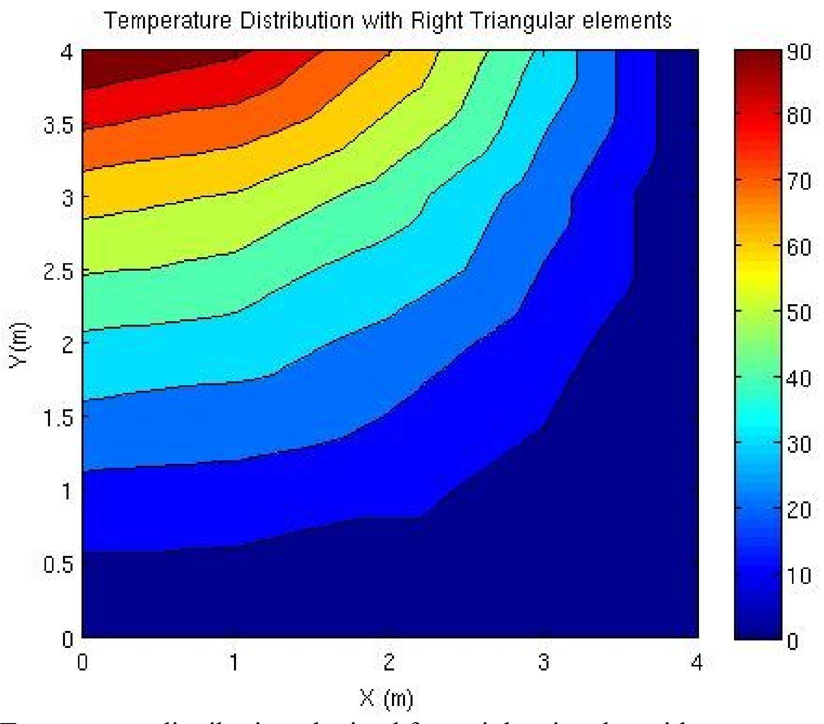


Figure 6. Temperature distribution obtained from right triangles with unstructured grid approach

From Figure [6] we see that the temperature levels are not as smooth as of in Figure [5]. The reason behind is the number of elements are considerably small than that of the original unstructured grid where 864 element triangles are used. However, instead of such a wide variation in number of elements and nodes, still the result obtained following the finite element approach are very similar to each other.

V. Conclusion

Second order heat conduction equation is solved for a rectangular 2d domain using galerkin finite elements methods. For the finite element model triangular and rectangular elements are considered. The element matrices are manually constructed. A separate model is also developed to determine the solution using unstructured grid. The code is implemented in MATLAB. Specific boundary conditions are used for the nodes that reside on the physical boundary. The result obtained using finite element model is compared with actual solution. The Numerical result showed agreement with actual solution.

References

[1]Reddy, J. N., “An Introduction to the Finite Element Method.” London: Mcgraw Hill Higher Education, 2006. Print.

Appendix A

Matlab codes for finite element method

triangularUnstructured.m

```
%
% Finite element solution for steady state heat conduction equation %
% with unstructured triangular element
%
clc;
clear all;
close all;
T0=100;
a=1;
conductivity=25;
fid=fopen('domain.data');
line=fgetl(fid);
temp=fscanf(fid,'%d');
nelem=temp(1);
nnode=temp(2);
line=fgetl(fid);  %bface
connectivity=zeros(nelem,4);
nodePoints=zeros(nnode,3);
beta=zeros(nelem,3);
gamma=zeros(nelem,3);
k=zeros(3,3,nelem);
area=zeros(nelem,1);
globalK =zeros(nnode,nnode) ;


for i=1:nelem
    line=fgetl(fid);
    temp=sscanf(line,'%d')';
    connectivity(i,1)=temp(1);
    connectivity(i,2)=temp(2);
    connectivity(i,3)=temp(3);
    connectivity(i,4)=temp(4);
end

line=fgetl(fid); %coordinates of the nodePoints

for i = 1: nnode
    tline = fgetl(fid);
    temp = sscanf(tline, '%f')';
    nodePoints(i,1) = temp(2);
    nodePoints(i,2)=temp(3);
    nodePoints(i,3)=temp(4);
%     nodePoints(i,1) = temp(1);
%     nodePoints(i,2)=temp(2);
%     nodePoints(i,3)=temp(3);
end


%calculate element matrix and store in k 3d arrayt
for n=1:nelem
    xi=nodePoints(connectivity(n,2),1);
    xj=nodePoints(connectivity(n,3),1);
    xk=nodePoints(connectivity(n,4),1);
    yi=nodePoints(connectivity(n,2),2);
    yj=nodePoints(connectivity(n,3),2);
    yk=nodePoints(connectivity(n,4),2);
    beta(n,2)=yk-yi;
    beta(n,3)=yi-yj;
    beta(n,1)=yj-yk;
    gamma(n,1)=-(xj-xk);
    gamma(n,2)=-(xk-xi);
    gamma(n,3)=-(xi-xj);
    area(n)=polyarea([xi,xj,xk],[yi,yj,yk]);

end

for e=1:nelem
    for i=1:3
        for j=1:3
                        k(i,j,e)=conductivity/(4*area(e))*(beta(e,j)*beta(e,i)
+gamma(e,j)*gamma(e,i));
%            fprintf('i=%d j=%d out=%d\n',i,j,e);
        end
    end
end

B=connectivity(:,2:4);

%assemble global matrix
for i=1:nelem
    for irow=1:3
        for icol=1:3
            globalK(B(i,irow),B(i,icol))=  globalK(B(i,irow),B(i,icol))+ ...
                k(irow,icol,i);
        end;
    end;
end;
%initialize variables for ease of calculation
i=1;
count1=1;
count2=1;
left=0;
right=4.0;
top=4.0;
bottom=0;

%define boundary nodes
for n=1:nnode
    if nodePoints(n,1)==right ||nodePoints(n,2)==bottom
        zeroNodeIndex(count1)=n;
        count1=count1+1;
    end
    if nodePoints(n,2)==top  %top boundary,y=4
      topBoundaryNodeIndex(count2)=n;
      topBoundaryNodeValues(count2,1)=T0*cos(pi*nodePoints(n,1)/(8*a));
      count2=count2+1;
    end

end

%right hand vector
rhsVector=-sum(globalK(:,topBoundaryNodeIndex)*topBoundaryNodeValues,2);

%remove rows and columns

deleteColumn=cat(2,zeroNodeIndex,topBoundaryNodeIndex);
globalK(:,deleteColumn(1,:) )=[]; % delete columns
globalK(deleteColumn(1, :), :)=[]; % delete rows
rhsVector(deleteColumn(1,:))=[];

feSol=globalK\rhsVector;

count=1;
for i=1:nnode
    zeroIndex=find(zeroNodeIndex==i);
    nonZeroIndex=find(topBoundaryNodeIndex==i);
    if zeroIndex>0
        solutionVector(i)=0.0;
    elseif nonZeroIndex>0
        solutionVector(i)=topBoundaryNodeValues(nonZeroIndex);
    else
        solutionVector(i)=feSol(count);
        count=count+1;
    end;
end;

figure(1);
[xp,yp]=meshgrid(0:a/10:4*a,0:a/10:4*a);
T=griddata(nodePoints(:,1),nodePoints(:,2),solutionVector,xp,yp);
contourf(xp,yp,T);
title('Temperature distribution using unstructrued triangular elements');
xlabel('x(m)');
ylabel('y(m)');
colorbar;
figure(2);
x=0:0.5:4;
y=0:0.5:4;
[X,Y]=meshgrid(x,y);
actual_sol = (T0.*sinh(pi.*Y./(8*a)) .*cos(pi.*X./(8*a))) ./(sinh(pi/2));
contourf(X,Y,actual_sol);
title('Temperature distribution by actual solution');
xlabel('x(m)');
ylabel('y(m)');
colorbar;
```

triangular.m

```
%
% Finite element solution for steady state heat conduction equation %
% with structured right triangular element
%
clc;
clear all;
close all;
T0=100;
a=1;
conductivity=25;
nelem=32;
nnode=25;
globalK =zeros(nnode,nnode) ;

B = [7, 6, 1;
1, 2, 7;
8, 7, 2;
2, 3, 8;
9, 8, 3;
3, 4, 9;
10, 9, 4;
4, 5, 10;
12, 11, 6;
6, 7, 12;
13,12, 7;
7, 8, 13;
14,13, 8;
8, 9, 14;
15, 14, 9;
9, 10, 15;
17, 16, 11;
11, 12, 17;
18, 17, 12;
12, 13, 18;
19, 18, 13;
13, 14, 19;
20, 19, 14;
14, 15, 20;
22, 21, 16;
16, 17, 22;
23, 22, 17;
17, 18, 23;
24, 23, 18;
18, 19, 24;
25, 24, 19;
19, 20, 25];


k=conductivity/2*[1 -1 0;
                  -1 2 -1;
                  0 -1 1];
```

```
%assemble global matrix
for i=1:nelem
    for irow=1:3
        for icol=1:3
            globalK(B(i,irow),B(i,icol))=  globalK(B(i,irow),B(i,icol))+ ...
                k(irow,icol);
        end;
    end;
end;

zeroNodeIndex=[1 2 3 4 5 10 15 20 25];
topBoundaryNodeIndex=[21 22 23 24];
topBoundaryNodeValues=[T0;T0*cos(pi/8);T0*cos(pi/4);T0*cos(3*pi/8)];
rhsVector=-sum(globalK(:,topBoundaryNodeIndex(1,:)                           )
*topBoundaryNodeValues,2);

%remove rows and columns

deleteColumn=cat(2,zeroNodeIndex,topBoundaryNodeIndex);
globalK(:,deleteColumn(1,:) )=[]; % delete columns
globalK(deleteColumn(1, :), :)=[]; % delete rows
rhsVector(deleteColumn(1,:))=[];

feSol=globalK\rhsVector;

count=1;
for i=1:nnode
    zeroIndex=find(zeroNodeIndex==i);
    nonZeroIndex=find(topBoundaryNodeIndex==i);
    if zeroIndex>0
        solutionVector(i)=0.0;
    elseif nonZeroIndex>0
        solutionVector(i)=topBoundaryNodeValues(nonZeroIndex);
    else
        solutionVector(i)=feSol(count);
        count=count+1;
    end;
end;

 x=[0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4];
 y=[0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4];
 [xp,yp]=meshgrid(0:a/10:4*a,0:a/10:4*a);
 T=griddata(x,y,solutionVector,xp,yp);
 contourf(xp,yp,T);
 title('Temperature distribution using triangular elements');
 xlabel('x(m)');
 ylabel('y(m)');
 colorbar;
```

rectangular.m

```
%
% Finite element solution for steady state heat conduction equation %
% with structured rectangular element
%
clc;
clear all;
close all;
T0=100;
a=1;
conductivity=25;
nelem=16;
nnode=25;
globalK =zeros(nnode,nnode) ;

B=[1 2 7 6;
   2 3 8 7;
   3 4 9 8;
   4 5 10 9;
   6 7 12 11;
   7 8 13 12;
   8 9 14 13;
   9 10 15 14;
   11 12 17 16;
   12 13 18 17;
   13 14 19 18;
   14 15 20 19;
   16 17 22 21;
   17 18 23 22;
   18 19 24 23;
   19 20 25 24];


k=conductivity/6*[4 -1 -2 -1;
        -1 4 -1 -2;
        -2 -1 4 -1;
        -1 -2 -1 4];




%assemble global matrix
for i=1:nelem
    for irow=1:4
        for icol=1:4
            globalK(B(i,irow),B(i,icol))=  globalK(B(i,irow),B(i,icol))+ ...
                k(irow,icol);
        end;
    end;
end;

zeroNodeIndex=[1 2 3 4 5 10 15 20 25];
topBoundaryNodeIndex=[21 22 23 24];
topBoundaryNodeValues=[T0;T0*cos(pi/8);T0*cos(pi/4);T0*cos(3*pi/8)];
rhsVector=-1*sum(globalK(:,topBoundaryNodeIndex(1,:)                         )
*topBoundaryNodeValues,2);

%remove rows and columns

deleteColumn=cat(2,zeroNodeIndex,topBoundaryNodeIndex);
globalK(:,deleteColumn(1,:) )=[]; % delete columns
globalK(deleteColumn(1, :), :)=[]; % delete rows
rhsVector(deleteColumn(1,:))=[];

feSol=globalK\rhsVector;

count=1;
for i=1:nnode
    zeroIndex=find(zeroNodeIndex==i);
    nonZeroIndex=find(topBoundaryNodeIndex==i);
    if zeroIndex>0
        solutionVector(i)=0.0;
    elseif nonZeroIndex>0
        solutionVector(i)=topBoundaryNodeValues(nonZeroIndex);
    else
        solutionVector(i)=feSol(count);
        count=count+1;
    end;
end

 x=[0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4];
 y=[0 0 0 0 0 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4];
 [xp,yp]=meshgrid(0:a/10:4*a,0:a/10:4*a);
 T=griddata(x,y,solutionVector,xp,yp);
 contourf(xp,yp,T);
 title('Temperature distribution using rectangular elements');
 xlabel('x(m)');
 ylabel('y(m)');
 colorbar;
```