Finite Element- 2 Dimensional Model for Thermal Distribution

Vinh The Nguyen

August 4, 2010
Abstract

Computerized thermal modeling is vital in engineering designs nowadays. Finite element method has been applied to give highly accurate approximate results. In this paper we will discuss about using finite element method, specifically triangular elements, with Matlab to generate a 2 dimensional model for thermal distribution. We will also compute the maximum error of the model as the number of elements increases, discuss about temperature on side of the model, and observe the change in temperature distribution if there is a defect in the model.

0.1 Introduction

Finite element method is a numerical technique to find approximate results of partial differential equations (PDE). This method uses a complex system of points called nodes which make a grid called a mesh. This mesh is programmed to contain the material and structural properties which define how the structure will react to certain loading conditions. [2]

The challenge of this method is to create equations to approximate the equations to be studied. The Finite Element Method is a good choice for solving partial differential equations over complicated domains (like cars and oil pipelines), when the domain changes (as during a solid state reaction with a moving boundary), when the desired precision varies over the entire domain, or when the solution lacks smoothness. [1]

In our problem, we want to construct a two dimensional plate model (at steady state) for thermal distribution using finite element method. We also use finite element to construct a defected model and study temperature distribution throughout it. The temperature distribution is a PDE problem as following:

\[
\frac{\delta^2 \vartheta}{\delta x^2} + \frac{\delta^2 \vartheta}{\delta y^2} = 0, \quad \Omega = \{0 < x < 5; 0 < y < 10\}
\]

(1)

With boundary conditions:

\[
\begin{align*}
\vartheta(x, 0) &= 0 & 0 < x < 5 \\
\vartheta(y, 0) &= 0 & 0 < y < 10 \\
\vartheta(x, 10) &= 100 \sin(\frac{\pi x}{10}) & 0 < x < 5 \\
\frac{\delta \vartheta}{\delta x}(5, y) &= 0 & 0 < y < 10
\end{align*}
\]

(2) (3) (4) (5)

This PDE problem is approached by using Matlab to build triangular mesh as shown in figure 1. Also the exact solution of this problem is given as:

\[
\vartheta(x, y) = \frac{100 \sinh(\frac{\pi y}{10}) \sin(\frac{\pi x}{10})}{\sinh(\pi)}
\]

(6)

In this problem we also need to compare the numerical solution generated by Matlab and the exact analytical solution.

0.2 2D Plate Model

We started up using 2D model in order to generate new understanding and improve computer method for calculating temperature distribution along the model. 2D model has great advantages because it is cheap, fast to process, give accurate numerical result, and could be processed more efficient and faster with parallel method.
0.2.1 Building Triangular Mesh

In this problem, the plate model is assumed to be infinitely thin and has width of 5 \((0 < x < 5)\) and length of 10 \((0 < y < 10)\). Initially, the plate is constructed with 25 nodal points from bottom up. Then, 32 triangular elements are created by making connectivity between nodes (figure 1). Furthermore, every element has three local nodes (1, 2, and 3) which are referred to to specific nodal points. Matching up all the local nodes of elements will create the expecting rectangle mesh.

![Figure 1: 32 triangular element mesh](image)

As in the figure above, the number in circle is element number (32 elements in this case). The whole mesh is built by stacking up couples of triangular element facing opposite direction bottom up to top. The boundary conditions are also shown in this figure. Temperature distributed on the top of the plate follow a sinusoidal function (7). The temperatures of the left side and the bottom of the plate are assumed to be zero initially, and the heat flux on the right side of the plate is also zero (no heat coming into or escapes from the right side of the plate or heavily insulated) (8).

\[
\Theta = 100 \sin\left(\frac{\pi x}{10}\right) \quad (7)
\]

\[
\frac{\delta \Theta}{\delta y} = 0 \quad (8)
\]
The mesh can be reconstructed with the same length and width but with more elements by increasing the number of nodal points which will be discussed later.

0.2.2 Temperature Distribution In The Plate

After building the mesh, we want to see how temperature distributes throughout the model as can be seen in figure 2 below:

![Temperature Distribution](image)

Figure 2: Matlab’s numerical result for temperature distribution of 32 element plate

The material studied in this model has thermal conductivity of $k = 10$ which in family of ceramics. By real life experiences, if heat is applied to material with low thermal conductivity, the expected result would be that the heat concentrates at applied spot and doesn’t spread out very far throughout the specimen. As in the figure, we can see that heat concentrates highly at the top right of the model (highest of 100 degree at the tip) and spreading downward. In figure above, we don’t have a very smooth result due to small amount of elements and nodes.
0.2.3 Matlab Code for 25 Nodes and 32 Elements

This is our first approach to build 32 elements. This way of approach is not really efficient and we will introduce the alternative after this. Here we create connectivity between local nodes of elements in order to construct the mesh

```
% zeros matrix (2 by 25)
xn=zeros(nsd,nnp);

Using repmat to replicate matrices
A=[0:1.25:5];
B=repmat(A,1,5);

Inputing x and y value for nodal points
xn(1,:)=B;
xn(2,1:1:5)=0;
xn(2,6:1:10)=2.5;
xn(2,16:1:20)=7.5;
xn(2,21:1:25)=10;

Connectivity between nodes
Local nodes are matched up with global nodes
```

Row 1 of matrix— LOCAL NODE 1
- Local node 1 of elements 1 to 4
- Local node 1 of elements 7 to 10
- Local node 1 of elements 6 to 9
- Local node 1 of elements 13 to 16
- Local node 1 of elements 9 to 12
- Local node 1 of elements 17 to 20
- Local node 1 of elements 21 to 24
- Local node 1 of elements 25 to 28
- Local node 1 of elements 29 to 32

Row 2 of matrix— LOCAL NODE 2
- Local node 2 of elements 1 to 8
- Local node 2 of elements 9 to 16
- Local node 2 of elements 17 to 24
- Local node 2 of elements 25 to 32

Row 3 of matrix— LOCAL NODE 3
- Local node 3 of elements 1 to 4
- Local node 3 of elements 5 to 8
- Local node 3 of elements 9 to 12
- Local node 3 of elements 13 to 16
- Local node 3 of elements 17 to 20
- Local node 3 of elements 21 to 24
- Local node 3 of elements 25 to 28
- Local node 3 of elements 29 to 32

0.2.4 Numerical Results as Number of Elements Increases

What would we expect for the numerical results of temperature distribution as the number of elements and nodal points increase? The answer will be shown in figure 3 below. As the number of elements increases from 128 to 1682, the result gets smoother and smoothers because there is less gap between data as in the original plate of 32 elements.
0.2.5 Matlab Code for Increasing Elements and Nodal Points

As we revised the code again and again, we finally come up with the final efficient code to increase number of nodal points and elements as shown below:

```matlab
nsd = 2;            % number of space dimension
ndf = 1;            % number of degree of freedom per node
nen = 3;            % number of element nodes
nxd = 5;            % number of points in x direction
nyd = 5;            % number of points in y direction
xinc = 5/(nxd-1);   % Increment in x direction
yinc = 10/(nyd-1);  % Increment in y direction
nel = (nxd-1)*(2*(nyd-1))-2; % number of elements
nnp = nxd*nyd;      % number of nodal points

% Nodal coordinates

% xn(i,N) = coordinate i for node N
% N = 1, ..., nnp
% i = 1, ..., nsd
xn = zeros(nsd,nnp); %2 by nnp matrix
pnumb = 0;
for j = 1:nyd
    for i = 1:nxd
        pnumb = pnumb + 1;
        % inputing x and y value for nodal points
        xn(1,pnumb) = (i-1)*xinc;
        xn(2,pnumb) = (j-1)*yinc;
    end
end
```

Figure 3: Matlab’s numerical results as number of elements increases from left to right (a), (b), and (c)
0.2.6 Temperature Distribution on Right Side of The Model

As we already succeed in thermal distribution throughout the plate model, we want to analyze the temperature distribution on the right side of the model, where flux is assumed to be zero. The result is expected to match with the exactly solution:

\[
\vartheta(x, y) = \frac{100 \sinh \left( \frac{x \pi}{10} \right) \sin \left( \frac{\pi y}{10} \right)}{\sinh(\pi)}
\]  
(9)

Holding \(x = 5\) for all \(y\), will give us the temperature distribution on the right side of the plate. Then, equation (9) become:

\[
\vartheta(5, y) = \frac{100 \sinh \left( \frac{y \pi}{10} \right) \sin \left( \frac{5 \pi}{10} \right)}{\sinh(\pi)} ; \quad 0 < y < 10
\]  
(10)

Back to our 2D model, we want to see how our result would match up with the exact solution above (10). Starting with 32 elements, the temperature at the bottom is zero and rises up to 100 at the top. However, because there are big gaps between data of nodal points on the side of the model, the result shown in figure below is not very smooth. But, again as we start to increase the number of elements, we observe the same phenomenon as in the above section, the result start to get smoother and smoother due to less and less data gap.

![Temperature distribution as number of elements increases](image)

As we observe that the result get smoother and smoother, we also want to see if they converge to the exact solution. To do that, we plot all the result in one graph as in figure below:
0.2.7 Matlab Code for Plotting Side Temperature

```matlab
A = load('32elements.mat');  %Load temperature of nodal points in 32 element plate
B = load('128elements.mat');  %Load temperature of nodal points in 128 element plate
C = load('512elements.mat');  %Load temperature of nodal points in 512 element plate
D = load('1682elements.mat');  %Load temperature of nodal points in 1682 element plate
x=5;  %holding x constant
y=linspace(0,10);  %Creating domain for y
\[ g = 100 \times \frac{\sinh(\pi y/10) \times \sin(\pi x/10)}{\sinh(\pi)} \]  %exact solution

Ucomp1 = A.Ucomp(5:5:25);  %Reading temperature on the right side of 32 element plate
Ucomp2 = B.Ucomp(9:9:81);  %Reading temperature on the right side of 128 element plate
Ucomp3 = C.Ucomp(17:17:289);  %Reading temperature on the right side of 512 element plate
Ucomp4 = D.Ucomp(30:30:900);  %Reading temperature on the right side of 1682 element plate
X1 = A.xn(2,5:5:25);  \quad X2 = B.xn(2,9:9:81);  
X3 = C.xn(2,17:17:289);  \quad X4 = D.xn(2,30:30:900);

%PLOT RESULTS
plot(X1, Ucomp1, 'p-');  \% 32 elements  
hold on
plot(X2, Ucomp2, 'mv-');  \% 128 elements
hold on
plot(X3, Ucomp3, 'h-k');  \% 512 elements
hold on
plot(X4, Ucomp4, 'r*');  \% 1682 elements
```

Figure 5: Convergence of results
0.2.8 Computing Error in The Model

Finite element method gives approximate results, so the model we built definitely has error compared to the exact numerical result. Before computing the error (in percentage) of the model, we used Matlab to construct the exact result based on the function below:

\[ \vartheta(x, y) = \frac{100 \sinh(\frac{\pi y}{10}) \sin(\frac{\pi x}{10})}{\sinh(\pi)} \]  

(11)

Next, we construct a parameter to calculate temperature at every nodal point. After we have the exact solution and the nodal point temperature, the error at each nodal point is computed by this equation (12):

\[ \text{Error} = \frac{\text{Exact Solution} - \text{Nodal Point temperature}}{\text{Exact Solution}} \times 100 \]

(12)

After calculating the error, we observe a phenomenon in which, the maximum percentage error dramatically drops as the number of elements increases. This phenomenon is shown in figure below.

Figure 6: Matlab’s numerical result for maximum percentage error as number of elements increases. Maximum percentage error drop from 6 (32 elements) to 1.6 (128 elements) to 0.43 (512 elements) to 0.13 (1682 elements)
0.2.9 Matlab Code for Error Computing

```matlab
% Exact Solution %
for i= 1:nnp
    Exact(i)= 100* sinh(pi*xn(2,i)/10)* sin(pi*xn(1,i)/10)/ sinh(pi);
end

% Error Computing %
Error=abs((Exact-Ucomp))/(Exact)*100;

for i=1:nnp
    if Exact(i)+Ucomp(i)==0
        if Exact(i)-Ucomp(i)==0
            Error(i)=0;
        else
            Error(i)=100;
        end
    end
end
```

0.3 Temperature Distribution In a Defected Plate Model

After computing the error in the model, we want to move forward to another important part in the project, temperature distribution of the model with a hole. What the temperature distribution would be like if there is a hole in the model? To approach this problem, we start with building the mesh.

0.3.1 Building the Mesh

The mesh is built in Matlab nearly the same way as with the original mesh, except for the middle part of it. 8 elements are taken out leaving an empty space in the mesh. In order to take away the elements, the number of nodal points and elements has to be reduced. Figure below shows the mesh with a hole in the middle.

![Mesh of the defected plate model](image-url)

Figure 7: Mesh of the defected plate model

9
As in figure 7 above, the nodal point is built in the same pattern from left to right and from bottom up. The hole is built by building the nodes so that they surround certain areas and by setting up no connectivity between nodes there.

0.3.2 Matlab Code of The Mesh With Hole

Since we haven’t figured out the relationship of nodes at the hole, the whole mesh is built manually and this is not an efficient way to study the model when the number of elements increases.

```matlab
%Building Nodal Points
xn(1,1:1:5)=0:1.25:5; %X value of elements 1 to 5
xn(1,6:1:10)=0:1.25:5; %X value of elements 6 to 10
xn(1,11)=0; xn(1,12)=1.25; %X value of elements at the hole (11, 12)
xn(1,13)=3.75;xn(1,14)=5; %X value of elements at the hole (13, 14)
xn(1,15:1:19)=0:1.25:5; %X value of elements 15 to 19
xn(1,20:1:24)=0:1.25:5; %X value of elements 20 to 24

xn(2,6:1:10)=2.5; %Y value of elements 6 to 10
xn(2,1:1:5)=0; %Y value of elements 1 to 5
xn(2,11:1:12)=5; %Y value of elements at the hole (11, 12)
xn(2,13:1:14)=5; %Y value of elements at the hole (13, 14)
xn(2,15:1:19)=7.5; %Y value of elements 15 to 19
xn(2,20:1:24)=10; %Y value of elements 20 to 24

%Building Local Nodal Points
ien(1,1:1:4)=1:1:4; %Local nodes 1 and 2 at hole
ien(1,5:1:8)=7:1:10;
ien(1,9)=6; ien(1,10)=9;
ien(1,11)=12; ien(1,12)=14;
ien(1,13)=11; ien(1,14)=13;
ien(1,15)=16; ien(1,16)=19;
ien(1,17:1:20)=15:18; %Local node 3 at hole
ien(1,21:24)=21:24;
ien(2,1:1:4)=2:5;
ien(2,5:1:8)=6:9;
ien(3,1:4)=7:10;
ien(3,5:8)=1:4;
ien(3,9)=12; ien(3,10)=14;
ien(3,11)=6; ien(3,12)=9;
ien(3,13)=16; ien(3,14)=19;
ien(3,15)=11; ien(3,16)=13;
ien(3,17:1:20)=21:24;
ien(3,21:1:24)=15:18;

After constructing the mesh, we want to set up boundary conditions for the model and observe its steady state temperature distribution.

0.3.3 Temperature Distribution in The Defected Model

The boundary conditions for this plate stays the same for all the sides as in the original plate. Heat is applied on the top of the plate as a sinusoidal function (7). There is no heat being applied to the left side
and the bottom side, the temperature at these places is considered to be absolute zero. There is no heat flux through the right side of the plate because it’s heavily insulated. The only difference in this model is that there is boundary condition at the hole and the temperature there is also considered to be zero.

After setting up all the boundary conditions, we have been able to make a thermal distribution graph for the model with Matlab as in figure 8 below.

As in figure 8 above, temperature spreading to almost everywhere on the upper part and the right side of the plate. The highest heat concentration is still at the top right corner. Because there is no heat flux through the hole, thermal energy has to be spread in different direction. In this case, heat is forced to spread throughout the top part and the right side part of the model.

Comparing Temperature Distribution of Original Plate Model and Defected Plate Model

After finishing the temperature distribution of the defected model, we want to compare the result to that of the original plate.
Figure 9: Matlab’s numerical results for the defected model and the original model from left to right (a), (b).

As in the figure we can see that heat is spreading further to the left at the top part and further down on the right side of the plate with hole.

0.3.4 Matlab Code for Temperature Distribution of The Defected Model

The temperature distribution in this model is controlled by boundary conditions as below:

```matlab
% Boundary conditions
% prescribed displacement (essential boundary condition)
% Idb(i,N)=1 if the degree of freedom i of the node N is prescribed
% =0 otherwise
% 1) initialize Idb to 0
idb=zeros(ndf,nnp);
% 2) enter the flag for prescribed displacement boundary conditions
for i = 1:nxd
    idb(1,i)=1;
end

for i = 1:nxd:(nyd*(nxd-1)+1)
    idb(1,i)=1;
end

for i = nxd*(nyd-1)+1:nxd*(nyd)
    idb(1,i)=1;
end
```

12
prescribed nodal temperature boundary conditions

% \( g(i,N) \): prescribed temperature for the dof \( i \) of node \( N \)

% initialize \( g \)
\( g = \text{zeros}(\text{ndf}, \text{nnp}); \)
% enter the values
\begin{verbatim}
for i=nxd*(nyd-1):nxd*nyd-1
    g(1,i)=100*\sin(pi*xn(1,i)/10); \%heat equation on top of the plate
end
\end{verbatim}
%Temperature at the hole
\( g(1,7:1:9)=0; \)
\( g(1,12:1:13)=0 \)
\( g(1,16:1:18)=0; \)

0.4 conclusion

Overall, finite element method gives very good approximate results for the PDE problem in this project as the number of elements increases. The maximum percentage of error also drop dramatically at very big amount of elements. Furthermore, this method is a useful tool to study the temperature distribution of a defected model, especially the plate model with a hole in the middle, where we have no exact solution to compare to. Clearly, using this method in Matlab give lots of advantages in numerical approximation.

0.5 Acknowledgements

I would like to thank Professor Nima Rahbar who guided us throughout the project, from proposing this project to providing us the background we need in finite element method. I also want to thank Professor Gottlieb, Professor Kim and Csums staff for their encouragement and support throughout the program.
Bibliography
