Project Report

FEM: 598 Finite Element Methods

Project Guide: Dr. Jay Oswald

Present by: Tanmay Dhanote

ASU ID: 1217423922

Contents

Problem Statement

The problem states that there is an aluminum rod with varying cross-section, and a current is flowing through it, and both the ends of the rod are at the same fixed temperature. Due to which there is heating in the rod based on the joule's heating law, and thus for the same, we must determine the steady-state temperature and heat flux by developing a finite element code for the system. After that, also compare the finite element code solution with the exact solution of the problem and simulating it in Abaqus or any commercial finite element code.

Problem Definition

Rod Geometry and Parameters

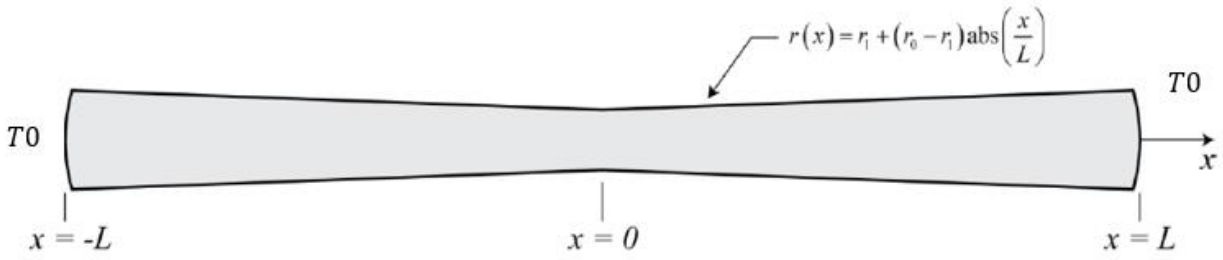


Figure 1: Rod Geometry and Parameters

Specified Input Variables

$$Current, I = 1000\ A$$

$$Temperature\ at\ end\ of\ Rod, T0 = 20°C$$

$$Length\ of\ Rod, 2L\ = 0.02\ m$$

$$Length, L\ = 0.01\ m$$

$$Radius\ at\ x = 0, r1 = 0.001\ m$$

$$Radius\ at\ x = L, r0 = 0.002\ m$$

$$Melting\ Temperature\ T_{Melt} = 660℃$$

Material and It's Properties

$$Material: Aluminium$$

$$Thermal\ Conductivity, k = 205\ {}^{W}/_{mK}$$

$$Electrical\ Resitivity, \rho = 2.82e^{-8}\ \Omega m$$

$$Electrical\ Resitivity\ as\ a\ function\ of\ Temperature, \rho(T) = 2.60e^{-8} + 1.1e^{-10}T\ \Omega m$$

Parameters and Input Variable Equations

$$Radius\ as\ a\ function\ of\ distance\ x, r(x) = r1 + (r0 - r1)abs\left(\frac{x}{L}\right)\ m$$

$$Joule's\ Heating, J = \frac{I^2\rho}{A^2}\ W/m^3$$

Analytical Solution

Problem Simplification

As the geometry is symmetric and imposed by symmetry, thermal boundary condition, i.e., the same fixed temperature at both the ends of the rod in our case. Thus, this led to the symmetry of temperature profile about the central plan, and therefore the center plan acts as an insulated wall or heat flux/temperature gradient is zero. Now to solve the problem, we can consider half geometry as our problem.

Figure 2: Simplified Problem with Boundary Conditions

NOTE: Temperature and heat flux plot present below are of Half geometry as the problem domain is symmetric about the center, i.e., about x=0.

Boundary Condition

$$BC: 1\ at\ x = 0, \frac{dT}{dx} = 0$$

$$BC: 2\ at\ x = L, T0 = 20℃$$

Governing Equation

The problem is steady-state heat conduction with heat generation due to Joule's heating effect

Assumption

Considering the problem as 1-D. The governing equation is as follows

$$1 - D\ Heat\ condction\ equation\ with\ heat\ generation, \frac{d}{dx}\left(kA\frac{dT}{dx}\right) + s = 0$$

$$Where\ "s"\ is\ heat\ genration\ per\ unit\ length$$

Solving the equation with prescribed boundary conditions

Input equations

$$Heat\ generation\ per\ unit\ length, s = JA = \frac{I^2\rho}{A}\ W/m$$

As now the x is positive for our defined simplified problem, we can remove the absolute from the radius equation, and thus the equation becomes:

$$Radius\ as\ a\ function\ of\ distance\ x, r(x) = r1 + (r0 - r1)\left(\frac{x}{L}\right)\ m$$

$$Area\ as\ a\ function\ of\ x, A(x) = \pi\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)^2\ m^2$$

Derivation

Input the function of Area and heat generation into the heat conduction equation to derive the equation for Heat Flux, Temperature Gradient, and Temperature as a function of distance.

$$\frac{d}{dx}\left(kA\frac{dT}{dx}\right) + \frac{I^2\rho}{\pi\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)^2} = 0$$

$$Integrating\ the\ above\ equation\ with\ respect\ to\ x\ to\ find\ heat\ flux\ and\ temperature\ gradient$$

$$Heat\ flux, k\frac{dT}{dx} = \frac{-\int \frac{I^2\rho}{\pi\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)^2}dx}{A} - \frac{c1}{A}$$

$$Where\ c1\ is\ constant\ of\ integration$$

$$Heat\ flux, k\frac{dT}{dx} = \frac{I^2\rho\ \mathrm{L}}{\pi^2(r0 - r1)\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)^3} - \frac{c1}{\pi\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)^2}$$

$$Temperature\ Gradient, \frac{dT}{dx} = \frac{I^2\rho\ \mathrm{L}}{\pi^2(r0 - r1)k\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)^3} - \frac{c1}{\pi k\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)^2}$$

$$Integratimg\ the\ temperature\ gradient\ to\ find\ the\ tempurature\ as\ function\ of\ x$$

$$Temperature, T = \frac{-I^2\rho\ \mathrm{L}^2}{2\pi^2 k(r0 - r1)^2\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)^2} + \frac{c1L}{\pi k(r0 - r1)\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)} - c2$$

$$Where\ c2\ is\ constant\ of\ integration$$

Now to determine the constant of integration, we use the prescribed boundary conditions.

1st Boundary Condition to Determine c1

$$BC: 1\ at\ x = 0, \frac{dT}{dx} = 0$$

Imputing the boundary condition

$$c1 = \frac{I^2 \rho\ \mathrm{L}}{\pi(r0 - r1)(r1)}$$

2nd Boundary Condition to determine c2

$$BC: 2\ at\ x = L, T0 = 20℃$$

Imputing the boundary condition and c1 value

$$c2 = \frac{I^2 \rho\ \mathrm{L}^2}{\pi^2 k(r0 - r1)^2 r0}\left(\frac{1}{r1} - \frac{1}{2r0}\right) - 20$$

Now inputting c1 and c2 values in Heat flux, temperature gradient, and temperature

$Heat\ flux,$

$$k\frac{dT}{dx} = \frac{I^2 \rho\ \mathrm{L}}{\pi^2 (r0 - r1)\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)^2}\left(\frac{1}{\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)} - \frac{1}{(r1)}\right)$$

$Temperature\ Gradient,$

$$\frac{dT}{dx} = \frac{I^2 \rho\ \mathrm{L}}{\pi^2 k(r0 - r1)\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)^2}\left(\frac{1}{\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)} - \frac{1}{(r1)}\right)$$

$Temperature,$

$$T = \left(\frac{I^2 \rho\ \mathrm{L}^2}{\pi^2 k(r0 - r1)^2}\right)\left(\frac{1}{\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)}\left(\frac{1}{r1} - \frac{1}{2\left(r1 + (r0 - r1)\left(\frac{x}{L}\right)\right)}\right) - \left(\frac{2\mathrm{r0} - \mathrm{r1}}{2r1r0^2}\right)\right) + 20$$

MATLAB code for Analytical Solution

Code

```
syms r0a r1a ka Ia La rhoa xa ra c1 c2
% parameters equations
ra=r1a +(r0a-r1a)*(xa/La);
Aa=pi()*ra^2;
%Derivation
pa=int(1/Aa,xa);
HF(r0a,r1a,ka,Ia,rhoa,La,xa,c1)=-(pa*Ia^2*rhoa)/Aa-c1/Aa;
dTdxa(r0a,r1a,ka,Ia,rhoa,La,xa,c1)=-((pa*Ia^2*rhoa)/(ka*Aa))-
c1/(ka*Aa);
Ta(r0a,r1a,ka,Ia,rhoa,La,xa,c1,c2)=int(dTdxa,xa)+c2;
%Inputs
k=205;
r0=0.002;
r1=0.001;
L=0.01;
I=1000;
rho=2.82e-8;
x_T=L;
x_dTdx=0;
% solving for integration constant
c1=solve(dTdxa(r0,r1,k,I,rho,L,x_dTdx,c1));
c2=solve(Ta(r0,r1,k,I,rho,L,x_T,c1,c2)==20,c2);
% Temperatre and Heat Flux calcuation
x=linspace(0,0.01,100);
Temp=Ta(r0,r1,k,I,rho,L,x,c1,c2);
Heat_flux=-HF(r0,r1,k,I,rho,L,x,c1);
% Plot of Temperature and Heat Flux
figure('name',"Temprature Profile");
plot(x,Temp)
xlim([0 L])
title('Temperature Profile')
xlabel('x distance (m)')
ylabel('Temeperature ( )')
figure('name',"Heat Flux Profile");
plot(x,Heat_flux)
xlim([0 L])
title('Heat Flux Profile')
xlabel('x distance (m)')
ylabel('Heat Flux (W/m^2)')
```

Output

Plot Temperature Profile Analytical

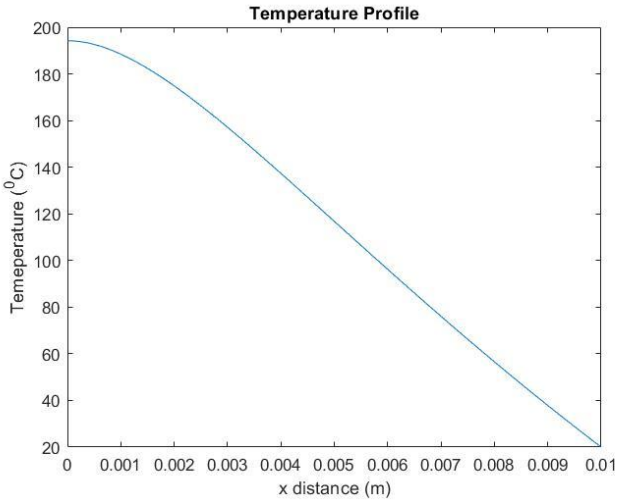


Figure 3: Temperature Profile Analytical Solution

Plot Heat Flux Profile Analytical

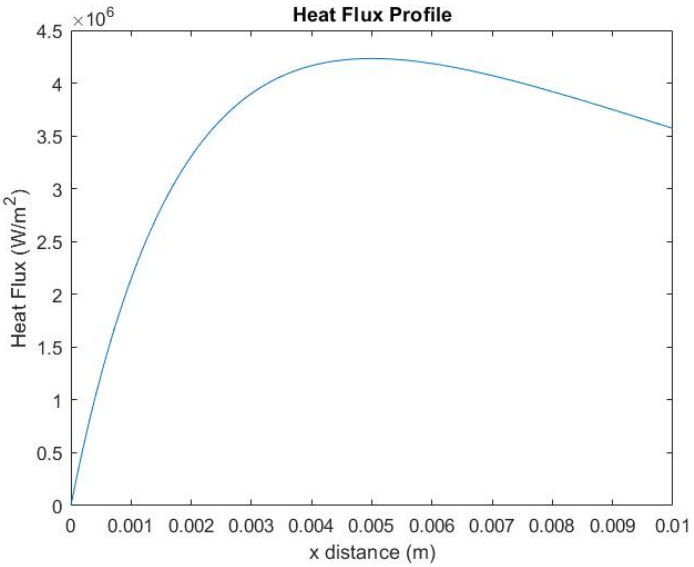


Figure 4: Heat Flux Profile Analytical Solution

Conclusion:

The maximum temperature (194.22°C) is observed at the central symmetry plan of the rod. The temperature equation shows that $T \propto \frac{2x-1}{x^2}$ Thus as x increases, the temperature will decrease.

The maximum Heat Flux (4.2328e+06 W/m^2) is observed at x=0.005 m

MATLAB Code for Finding Maximum Current Limit

Code

```
syms r0a r1a ka Ia La rhoa xa ra c1 c2 C1 C2
% parameters equations
ra=r1a +(r0a-r1a)*(xa/La);
Aa=pi()*ra^2;
%Derivation
pa=int(1/Aa,xa);
HF(r0a,r1a,ka,Ia,rhoa,La,xa,c1)=-(pa*Ia^2*rhoa)/Aa-c1/Aa;
dTdxa(r0a,r1a,ka,Ia,rhoa,La,xa,c1)=-((pa*Ia^2*rhoa)/(ka*Aa))-
c1/(ka*Aa);
Ta(r0a,r1a,ka,Ia,rhoa,La,xa,c1,c2)=int(dTdxa,xa)+c2;
%Inputs
k=205;
r0=0.002;
r1=0.001;
L=0.01;
I=1000;
rho=2.82e-8;
x_T=L;
x_dTdx=0;
c1_I_max=solve(dTdxa(r0,r1,k,Ia,rho,L,x_dTdx,C1),C1);
c2_I_max=solve(Ta(r0,r1,k,Ia,rho,L,x_T,c1_I_max,C2)==20,C2);
I_max=solve(Ta(r0, r1, k, Ia, rho, L, x_dTdx, c1_I_max,
c2_I_max)==660,Ia);
I_max=max(double(I_max));
disp(['Maximum Current,Imax = ',num2str(I_max),' A'])
```

Output

Maximum Current, Imax = 1916.6 A

Effect of Assumptions on the Predicted Value

1. There is no heat dissipation considered from the surface of the rod. Therefore the complete heat generated is absorbed by the rod and increasing its temperature. However, if there is some heat dissipation from the surface, more heat is required to reach the melting temperature. Which, in turn, would require more Current flow as heat generation is proportional to the square of the current.
2. We assumed that thermal conductivity and electrical conductivity is constant for the present model. As we know this, both are temperature-dependent and which makes current dependent on the two mentioned properties as follows: $I \propto \sqrt{\frac{k}{\rho}}$. Based on the cumulative effect of the two properties, an overall increase or decrease determines the more accurate maximum current limit.

Finite Element Method MATLAB Solution

MATLAB code

```
function [T, Heat_flux] =
tdhanote_Project01_FEM(nn_per_element,number_of_elements,k,r0,
r1,L,rho,no_qpts)
%% input defination

% nn_per_element = Number of nodes per element
% number_of_elements = Number of elements
% k = Thermal conductivity
% r0 = Radius at the ends of bar
% r1 = radius at the center of bar
% roh = Electrical Resistivity
% no_qpts = Number of quadrature Points

%% Inpts
if nargin==0
nn_per_element=2;
number_of_elements=4;
k=205;
r0=0.002;
r1=0.001;
L=0.01;
I=1000;
rho=2.82e-8;
no_qpts=20;
end
%% Heat Generation as a function of x

s=@(x)(((I^2)*rho)/(pi()*(r1 + (r0-r1)*(x/L))^2));
%% Arear as a function of x

A=@(x)(pi()*(r1 + (r0-r1)*(x/L))^2);

%% Quadture points selection

qpts=quadrature(no_qpts);

%% Total nodes and connectivity matrix determination Corresponding
shape function
%Linear
if nn_per_element==2
    nodes_total=number_of_elements+1;
    conn=[1:number_of_elements;2:number_of_elements+1];
    shape=@shape2;
% Quadratic
elseif nn_per_element==3
     nodes_total=2*number_of_elements+1;
```

```
     conn=[1:2:2*number_of_elements-
1;2:2:2*number_of_elements;3:2:2*number_of_elements+1];
     shape=@shape3;
%Cubic
else
 conn=[1:3:3*number_of_elements-2;2:3:3*number_of_elements-
1;3:3:3*number_of_elements;4:3:3*number_of_elements+1];
 nodes_total=3*number_of_elements+1;
 shape=@shape4;
end
%% Coordiantes of each Nodes

x=linspace(0, L,nodes_total);

%% Stiffness matrix  K Calculation

K=zeros(nodes_total);
for c=conn
    xe=x(:,c);
    Ke=zeros(length(c));
    for q=qpts
        [N,dNdp]=shape(q(1));
        J=xe*dNdp;
        B=dNdp/J;
        xq=xe*N;
        Ke=Ke+B*k*A(xq)*B'*J*q(2);
    end
    sctr=c;
    K(sctr,sctr)=K(sctr,sctr)+Ke;
end

%% Heat Flux Matrix calculations
fe=zeros(nodes_total,1);

for c=conn
    xe=x(:,c);
        for i=1:no_qpts
    [N,DNDp]=shape(qpts(1,i));
    J=xe*DNDp;
    w=qpts(2,i);
    xp=xe*N;
    F=s(xp)*N*J*w;
    fe(c)=fe(c)+F;
        end

end
%% Temperature Calculation at nodes

K(nodes_total, :)= 0;
K(nodes_total,nodes_total)=1;
fe(nodes_total,1)=20;
T=K\fe;
```

```
DTDX=[];
x1=[];
Heat_flux(1)=0;
%% Heat Flux
for c=conn
    xe=x(:,c);
    for p=linspace(-1,1,nn_per_element)
        [N,dNdp]=shape(p);
        J=xe*dNdp;
        B=dNdp/J;
        DTDX(end+1)=-B'*T(c);
        x1(end+1)=xe*N;
    end
end
Heat_flux(1:(size(DTDX,2)))=DTDX*k;
%% Plots
figure('name',"Temprature Profile");
plot(x,T)
hold on
xlim([0 L])
title(['Temperature Profile with ',num2str(number_of_elements), '
linear elements'])
xlabel('x Distance (m)')
ylabel('Temeperature (^0C)')
legend([num2str(number_of_elements), ' linear elements'])
hold off
figure('name',"Heat Flux Profile");
plot(x1,Heat_flux)
xlim([0 L])
title(['Heat Flux Profile with ',num2str(number_of_elements), ' linear
elements'])
xlabel('x Distance (m)')
ylabel('Heat Flux (W/m^2)')
legend([num2str(number_of_elements), ' linear elements'])
end
%% Shape functions for different elements
function [N,dNdp] = shape2(p)
N=0.5*[1-p(1);1+p(1)];
dNdp=[-0.5;0.5];
end
function [N,dNdp] = shape3(p)
N=[p(1)*(1-p(1))*-0.5;1-p(1)^2;0.5*p(1)*(p(1)+1)];
dNdp = [(p(1)-0.5);(-2*p(1));(p(1)+0.5)];
end
function [N,dNdp] = shape4(p)
N=[(p(1)^2-1/9)*(p(1)-1)*(-9/16);(p(1)^2-1)*(p(1)-
1/3)*(27/16);(p(1)^2-1)*(p(1)+1/3)*(-27/16);(p(1)^2-
1/9)*(p(1)+1)*(9/16)];
dNdp=[(-9/16)*(3*p(1)^2-2*p(1)-1/9);(27/16)*(3*p(1)^2-(2/3)*p(1)-1);(-
27/16)*(3*p(1)^2+(2/3)*p(1)-1);(9/16)*(3*p(1)^2+2*p(1)-1/9)];
end
```

```
%% Gaussian Quadrature
function [qpts] = quadrature(n)
%   QUADRATURE
%     quadrature(n) returns a quadrature table for a rule with n
%     integration points.  The first row of the table gives the
quadrature
%     point location, and the second gives the quadrature weights.

    u = 1:n-1;
    u = u./sqrt(4*u.^2 - 1);

    A = zeros(n);
    A(2:n+1:n*(n-1)) = u;
    A(n+1:n+1:n^2-1) = u;

    [v, x] = eig(A);
    [x, k] = sort(diag(x));
    qpts = [x'; 2*v(1,k).^2];
end
```

Outputs

Plot for Temperature field for the linear element

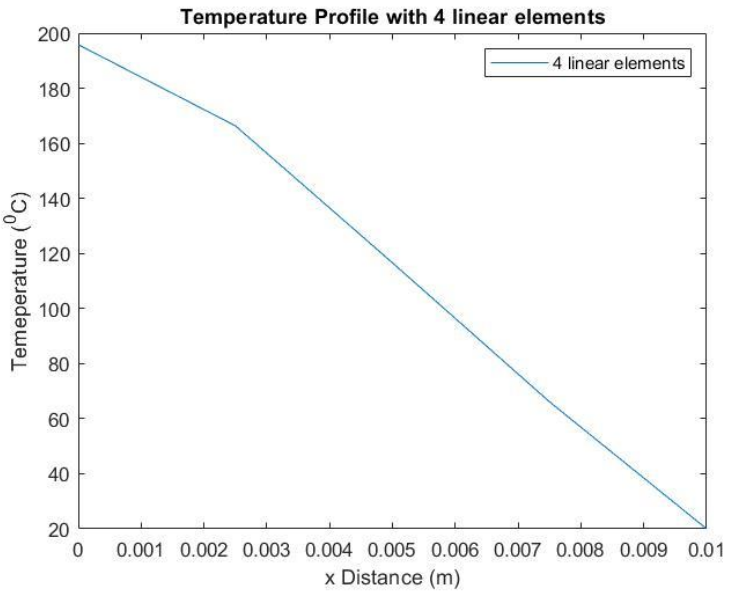


Figure 5: Temperature Field FEM 4 Linear Element

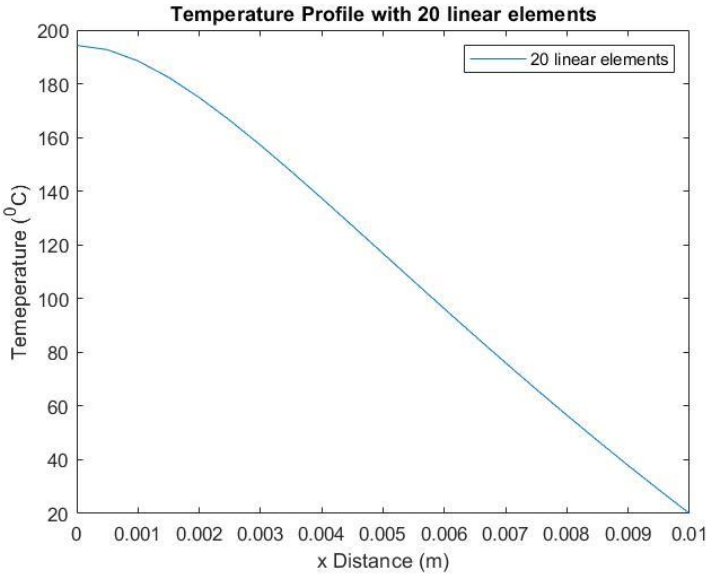


Figure 6: Temperature Field FEM 20 Linear Element

$$T_{max,4\ Linear\ element} = 195.8089℃\ and\ T_{max,20\ Linear\ element} = 194.2923℃$$

With the increase in the number of element smoothness in the temperature field plot increase and here, C_0 continuity is present. The maximum temperature is present at x=0 as per the analytical equation relation. Also, eL2 norm error decreases with an increase in the number of elements.

$$e\_L2_{\ 4\ Linear\ element} = 0.0130\ and\ e\_L2_{\ 20\ Linear\ element} = 6.0482e-04$$

Plot for Heat Flux field for the linear element

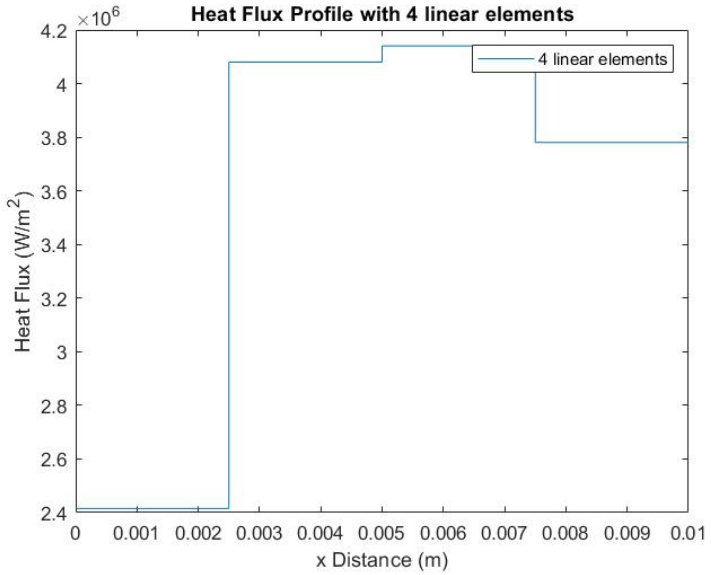


Figure 7: Heat Flux Field with 4 Linear Elements

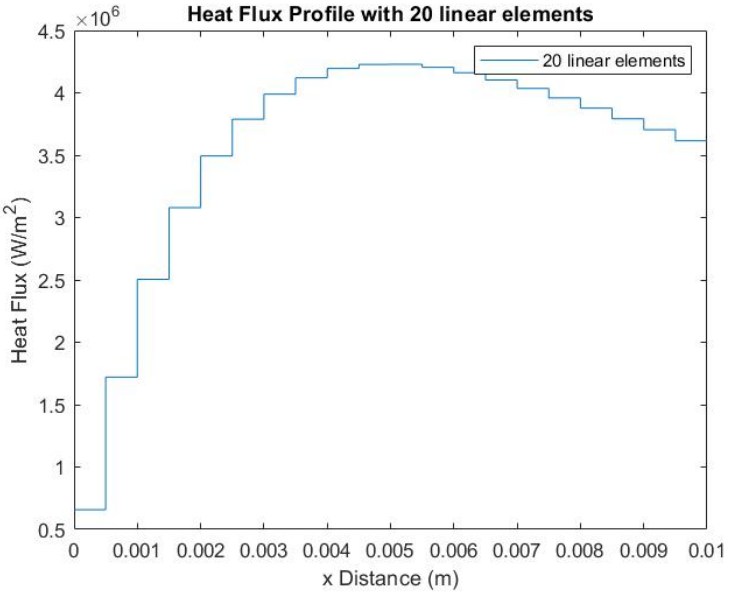


Figure 8: Heat Flux Field with 20 Linear Elements

Here also, With the increase in the number of element smoothness in the heat flux field increase, and here, C_1 continuity is present, and the elements are linear; therefore, the heat flux approximation in the element is constant. Also, energy error norm error decreases with an increase in the number of elements.

$$e\_en_{\,4\ Linear\ element} = 0.2678\ and\ e\_en_{\,20\ Linear\ element} = 0.0553$$

Plot for Temperature Field for the Quadratic Element

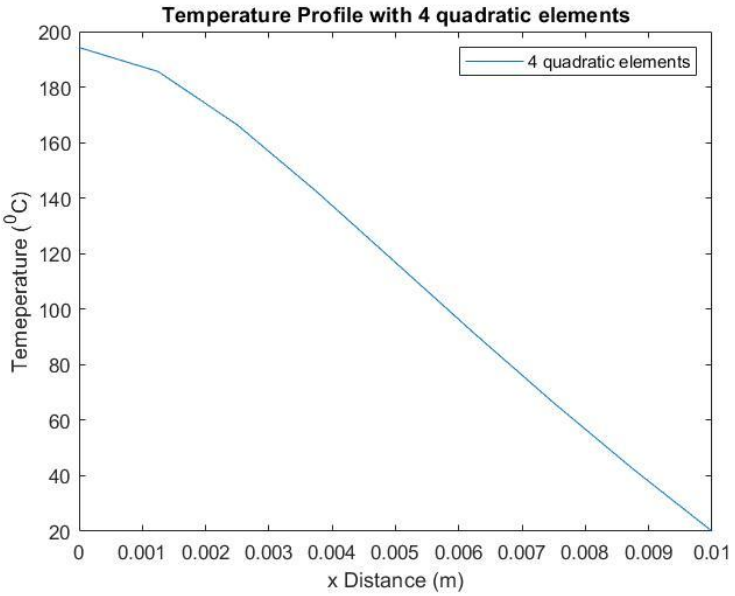


Figure 9: Temperature Field FEM 4 Quadratic Element

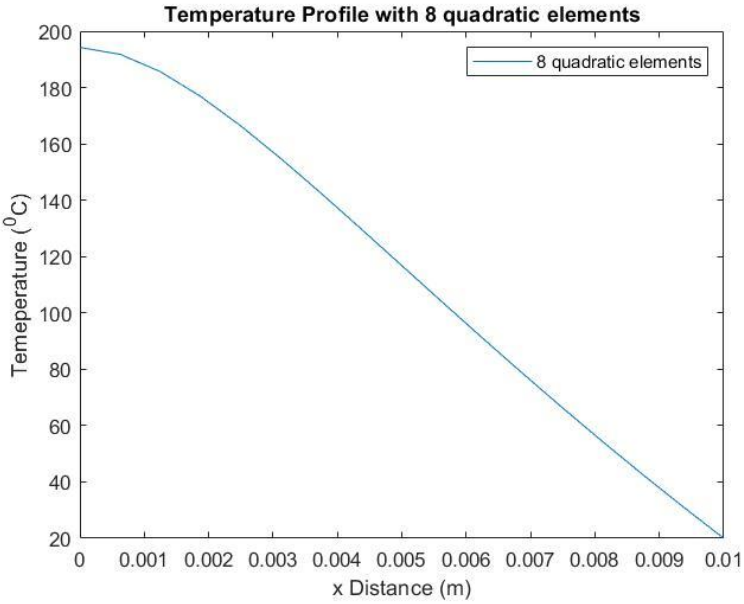


Figure 10: Temperature Field FEM 8 Quadratic Element

$$T_{max,4\ Quadratic\ element} = 194.2558℃\ and\ T_{max,8\ Quadratic\ element} = 194.2252℃$$

Taking quadratic elements, the temperature field within the element is more accurate, and here, C_0 continuity is present. The quadratic element fits the temperature field function in a better way and increases smoothness. Also, L2 norm error decreases with an increase in the number of elements.

$$e\_L2_{\ 4\ Quadratic\ element} = 0.0015 and\ e\_L2_{\ 8\ Quadratic\ element} = 2.1056e-04$$

Plot for Heat Flux field for the quadratic element

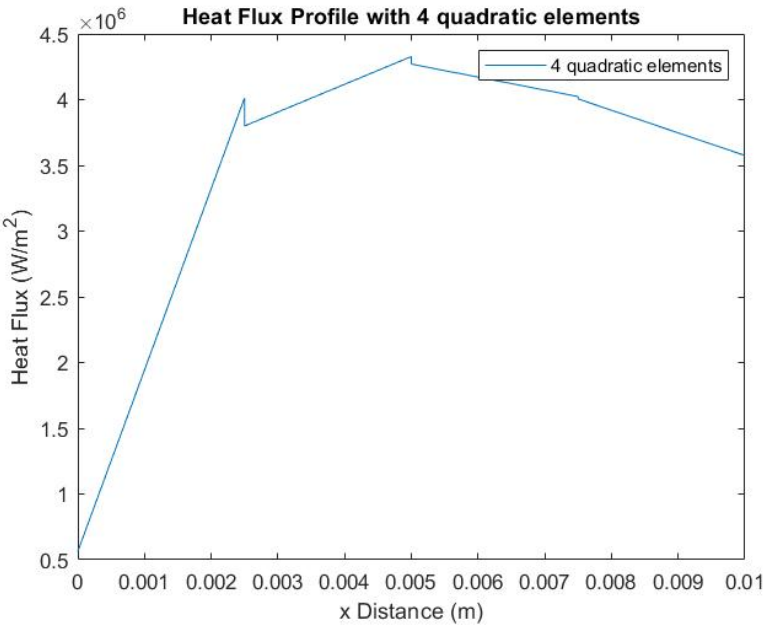


Figure 11: Heat Flux Field with 4 Quadratic Elements

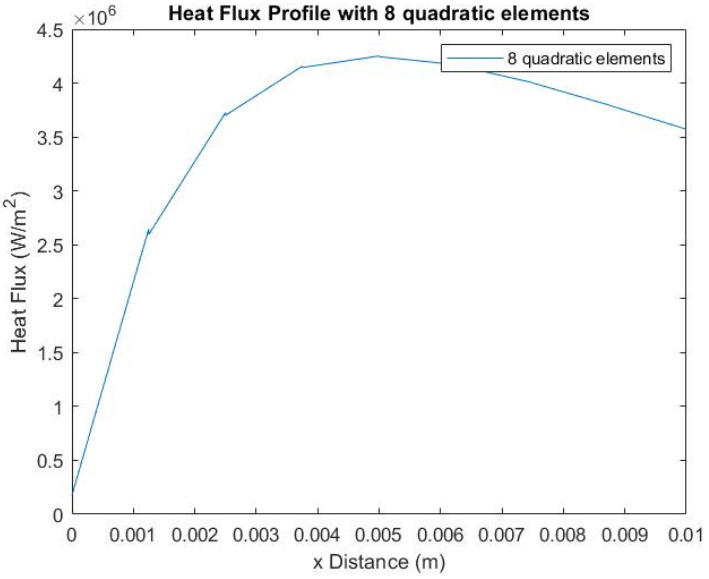


Figure 12: Heat Flux Field with 8 Quadratic Elements

Here also, With the increase in the number of element smoothness in the heat flux field increase, and here, C_1 continuity is present, and the elements are quadratic. Therefore, the heat flux is a linear approximation in an element. Also, energy error norm error decreases with an increase in the number of elements.

$$e\_en_{\,4\ Quadratic\ element} = 0.0570 and\ e\_en_{\,8\ Quadratic\ element} = 0.0151$$

Plot for Temperature Field for the Cubic Element

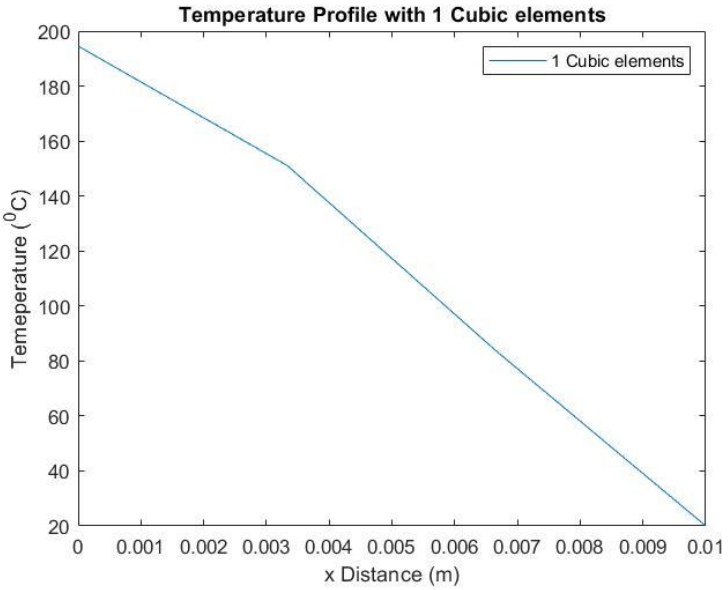


Figure 13: Temperature Field FEM 1 Cubic Element

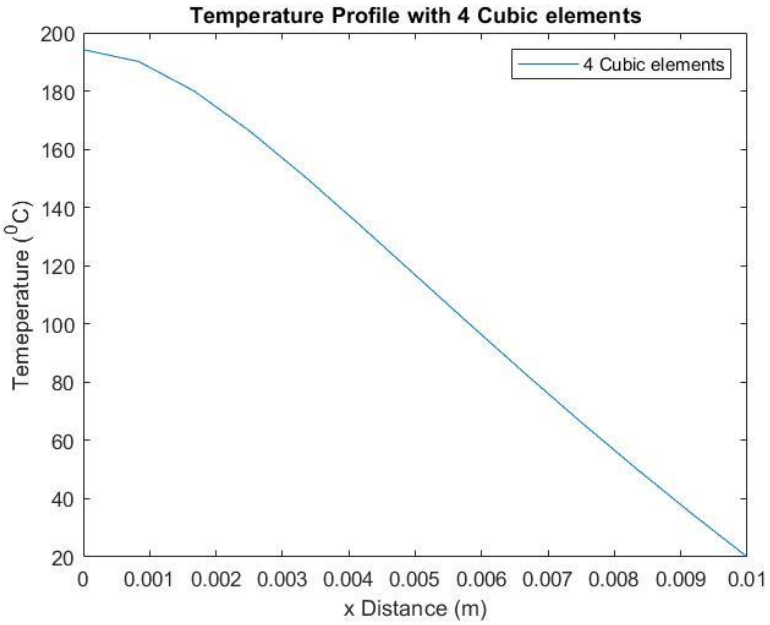


Figure 14: Temperature Field FEM 4 Cubic Element

$$T_{max,1\ Cubic\ element} = 194.5317℃\ and\ T_{max,4\ Cubic\ element} = 194.2233℃$$

Taking the Cubic element, the temperature field within the element is a cubic function, and here, C_0 continuity is present. Thus, the approximation temperature field has a close curve fit to the exact solution. As can be seen from L2 norm error decreases with an increase in the number of elements.

$$e\_L2_{\ 1\ Cubic\ element} = 0.0110 and\ e\_L2_{\ 4\ Cubic\ element} = 1.2346e-04$$

Plot for Heat Flux field for the Cubic element

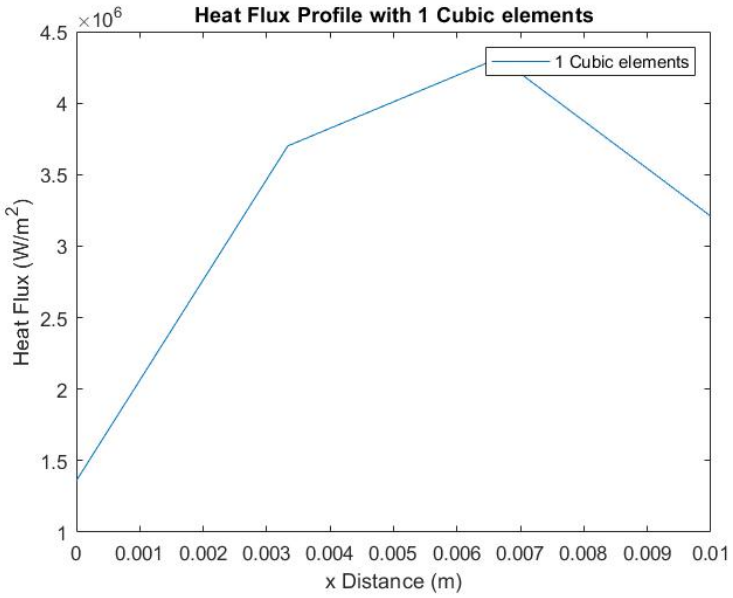


Figure 15: Heat Flux Field with 1 Cubic Element

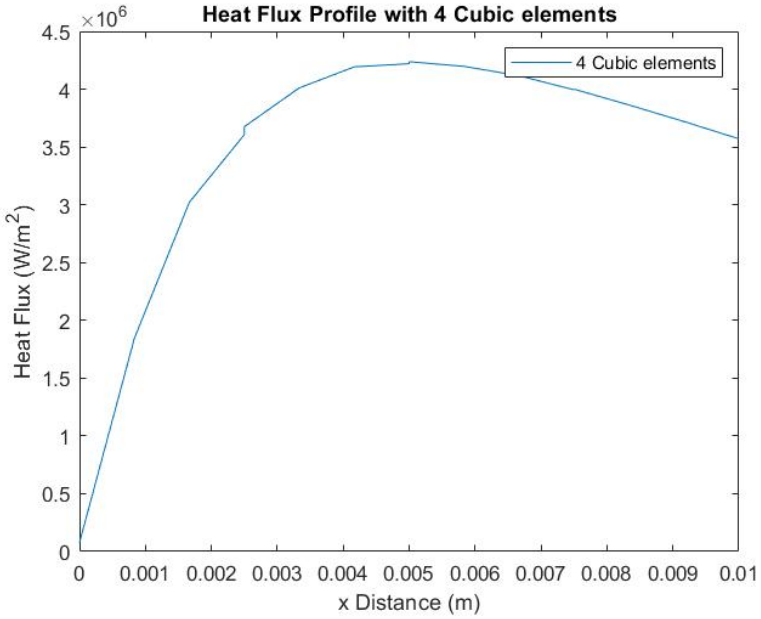


Figure 16: Heat Flux Field with 4 Cubic Elements

Here also, With the increase in the number of element smoothness in the heat flux field increase, and here, C_1 continuity is present, and the elements are cubic; therefore, the heat flux is a quadratic approximation of exact heat flux function in an element. Also, energy error norm error decreases with an increase in the number of elements.

$$e\_en_{\,1\ cubic\ element} = 0.2152\ and\ e\_en_{\,4\ Cubic\ element} = 0.0069$$

Error Norm

Code

```
function [fe,K,T,I_max,e_L2] =
tdhanote_Project01_FEM_and_Analytical_for_Error_Norm(nn_per_element,k,r0,
r1,L,rho,no_qpts)
%% input defination

% nn_per_element = Number of nodes per element
% number_of_elements = Number of elements
% k = Thermal conductivity
% r0 = Radius at the ends of bar
% r1 = radius at the center of bar
% roh = Electrical Resistivity
% no_qpts = Number of quadrature Points

%% Inpts
if nargin==0
nn_per_element=3;
k=205;
r0=0.002;
r1=0.001;
L=0.01;
I=1000;
rho=2.82e-8;
no_qpts=10;
end
count=1;
for number_of_elements=[4,8,16,32,64,128,256,512]
%% Heat Generation as a function of x

s=@(x)(((I^2)*rho)/(pi()*(r1 + (r0-r1)*(x/L))^2));
%% Arear as a function of x

A=@(x)(pi()*(r1 + (r0-r1)*(x/L))^2);

%% Quadture points selection

qpts=quadrature(no_qpts);

%% Total nodes and connectivity matrix determination Corresponding shape
function
%Linear
if nn_per_element==2
    nodes_total=number_of_elements+1;
    conn=[1:number_of_elements;2:number_of_elements+1];
    shape=@shape2;
% Quadratic
elseif nn_per_element==3
     nodes_total=2*number_of_elements+1;
     conn=[1:2:2*number_of_elements-
1;2:2:2*number_of_elements;3:2:2*number_of_elements+1];
     shape=@shape3;
%Cubic
else
 conn=[1:3:3*number_of_elements-2;2:3:3*number_of_elements-
1;3:3:3*number_of_elements;4:3:3*number_of_elements+1];
```

```
 nodes_total=3*number_of_elements+1;
 shape=@shape4;
end
%% Coordiantes of each Nodes

x=linspace(0, L,nodes_total);

%% Stiffness matrix  K Calculation

K=zeros(nodes_total);
for c=conn
    xe=x(:,c);
    Ke=zeros(length(c));
    for q=qpts
        [N,dNdp]=shape(q(1));
        J=xe*dNdp;
        B=dNdp/J;
        xq=xe*N;
        Ke=Ke+B*k*A(xq)*B'*J*q(2);
    end
    sctr=c;
    K(sctr,sctr)=K(sctr,sctr)+Ke;
end

%% Heat Flux Matrix calculations
fe=zeros(nodes_total,1);

for c=conn
    xe=x(:,c);
        for i=1:no_qpts
    [N,DNDp]=shape(qpts(1,i));
    J=xe*DNDp;
    w=qpts(2,i);
    xp=xe*N;
    F=s(xp)*N*J*w;
    fe(c)=fe(c)+F;
        end

end
%% Temperature Calculation at nodes

K(nodes_total, :)= 0;
K(nodes_total,nodes_total)=1;
fe(nodes_total,1)=20;
T=K\fe;

DTDX=[];
x1=[];
Heat_flux(1)=0;
%% Heat Flux
for c=conn
    xe=x(:,c);
    for p=linspace(-1,1,nn_per_element)
        [N,dNdp]=shape(p);
        J=xe*dNdp;
        B=dNdp/J;
        DTDX(end+1)=-B'*T(c);
```

```
        x1(end+1)=xe*N;
    end
end
Heat_flux(1:(size(DTDX,2)))=DTDX*k;


%% Analytical Solution

syms r0a r1a ka Ia La rhoa xa ra c1 c2 C1 C2
ra=r1a +(r0a-r1a)*(xa/La);
Aa=pi()*ra^2;
pa=int(1/Aa,xa);
HF(r0a,r1a,ka,Ia,rhoa,La,xa,c1)=-(pa*Ia^2*rhoa/Aa)-(c1/Aa);
dTdxa(r0a,r1a,ka,Ia,rhoa,La,xa,c1)=-((pa*Ia^2*rhoa)/(ka*Aa))-c1/(ka*Aa);
Ta(r0a,r1a,ka,Ia,rhoa,La,xa,c1,c2)=int(dTdxa,xa)+c2;
x_T=L;
x_dTdx=0;
c1=solve(dTdxa(r0,r1,k,I,rho,L,x_dTdx,c1)==0,c1);
c2=solve(Ta(r0,r1,k,I,rho,L,x_T,c1,c2)==20,c2);
x_a=linspace(0,0.01,11);
Temp_analytical=Ta(r0,r1,k,I,rho,L,x_a,c1,c2);
Heat_flux_analytical=-HF(r0,r1,k,I,rho,L,x_a,c1);

%% Convergence
Fun_num=0;
Fun_den=0;
for c=conn
    xe=x(c);
for q=1:size(qpts,2)
    [N,dNdp]=shape(qpts(1,q));
    xee=xe*N;
    T1=T(c)'*N;
    J=xe*dNdp;
    Temp_Exact=Ta(r0,r1,k,I,rho,L,xee,c1,c2);
    fun_num=(Temp_Exact-T1)^2;
    fun_den=Temp_Exact^2;
    Fun_num=Fun_num+fun_num*J*qpts(2,q);
    Fun_den=Fun_den+fun_den*J*qpts(2,q);
end
end
e_L2(count)=sqrt(abs(Fun_num/Fun_den));
LE(count)=L/number_of_elements;
Fun_s_num=0;
Fun_s_den=0;
 for c=conn
     xe=x(:,c);
     for p=linspace(-1,1,nn_per_element)
         [N,dNdp]=shape(p);
         XX=xe*N;
         J=xe*dNdp;
         B=dNdp/J;
         Heat_Exact=-HF(r0,r1,k,I,rho,L,XX,c1);
         heat_flux_FEM=-B'*T(c)*k;
         fun_s_num=(Heat_Exact-heat_flux_FEM)^2;
         fun_s_den=Heat_Exact^2;
         Fun_s_num=Fun_s_num+(fun_s_num*J*qpts(2,q))/2;
         Fun_s_den=Fun_s_den+(fun_s_den*J*qpts(2,q))/2;
```

```
    end
 end
  e_en(count)=sqrt(abs(Fun_s_num/Fun_s_den));
count=count+1;
end
figure('name',"L2 temperature error norm vs element size")
plot(log10(LE),log10(e_L2),'o')
title('log(eL2)vs log(LE)')
xlabel('log(LE)--log of element length')
ylabel('log(eL2)--log of L2 Temperature error norm')
legend('Quadratic Element')
figure('name',"flux error norm vs element size")
plot(log10(LE),log10(e_en),'o')
title('log(een)vs log(LE)')
xlabel('log(LE)--log of element length')
ylabel('log(een)--log of flux error norm')
legend('Quadratic Element')

end
%% Shape functions for different elements
function [N,dNdp] = shape2(p)
N=0.5*[1-p(1);1+p(1)];
dNdp=[-0.5;0.5];
end
function [N,dNdp] = shape3(p)
N=[p(1)*(1-p(1))*-0.5;1-p(1)^2;0.5*p(1)*(p(1)+1)];
dNdp = [(p(1)-0.5);(-2*p(1));(p(1)+0.5)];
end
function [N,dNdp] = shape4(p)
N=[(p(1)^2-1/9)*(p(1)-1)*(-9/16);(p(1)^2-1)*(p(1)-1/3)*(27/16);(p(1)^2-
1)*(p(1)+1/3)*(-27/16);(p(1)^2-1/9)*(p(1)+1)*(9/16)];
dNdp=[(-9/16)*(3*p(1)^2-2*p(1)-1/9);(27/16)*(3*p(1)^2-(2/3)*p(1)-1);(-
27/16)*(3*p(1)^2+(2/3)*p(1)-1);(9/16)*(3*p(1)^2+2*p(1)-1/9)];
end
%% Gaussian Quadrature
function [qpts] = quadrature(n)
%   QUADRATURE
%     quadrature(n) returns a quadrature table for a rule with n
%     integration points.  The first row of the table gives the quadrature
%     point location, and the second gives the quadrature weights.

    u = 1:n-1;
    u = u./sqrt(4*u.^2 - 1);

    A = zeros(n);
    A(2:n+1:n*(n-1)) = u;
    A(n+1:n+1:n^2-1) = u;

    [v, x] = eig(A);
    [x, k] = sort(diag(x));
    qpts = [x'; 2*v(1,k).^2];
end
```

The plot of L2 temperature error norm vs. element size and rate of convergence for linear elements.

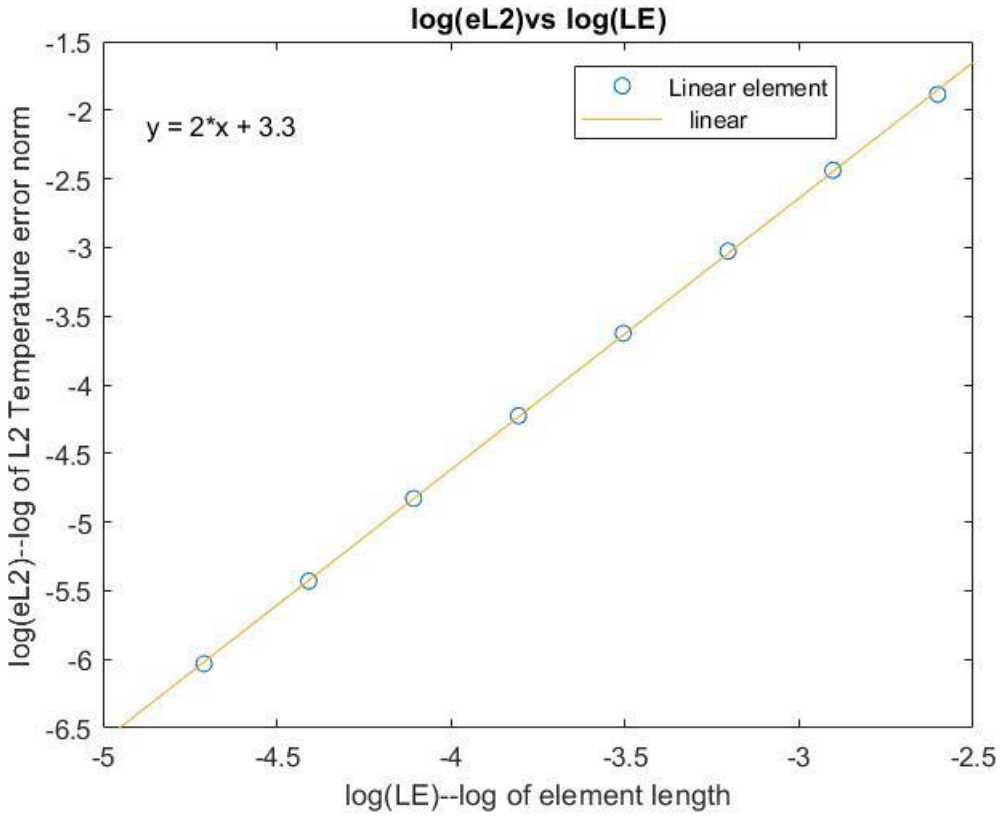


Figure 17: L2 Temperature Error Norm vs. Element Size for Linear Element

$$L2\ Error\ Norm\ Function, \log(e_{L2}) = C + \alpha \log(h)$$

$$e_{L2} - Error\ Norm\ C - Arbitrary\ Constant\ \alpha - Rate\ of\ Convergence\ h - Element\ Size$$

$$L2\ Error\ Norm\ Equatiion\ From\ Plot, \log(e_{L2}) = 3.3 + 2\log(h)$$

Comparing both the Equation, we obtain the convergence Rate $\alpha = 2$ As per the general Mathematical Literature Theory for Linear element, the rate of convergence is equal to 2, and our result is accurately matching.

The plot of Flux error norm vs. element size on and rate of convergence using linear elements.

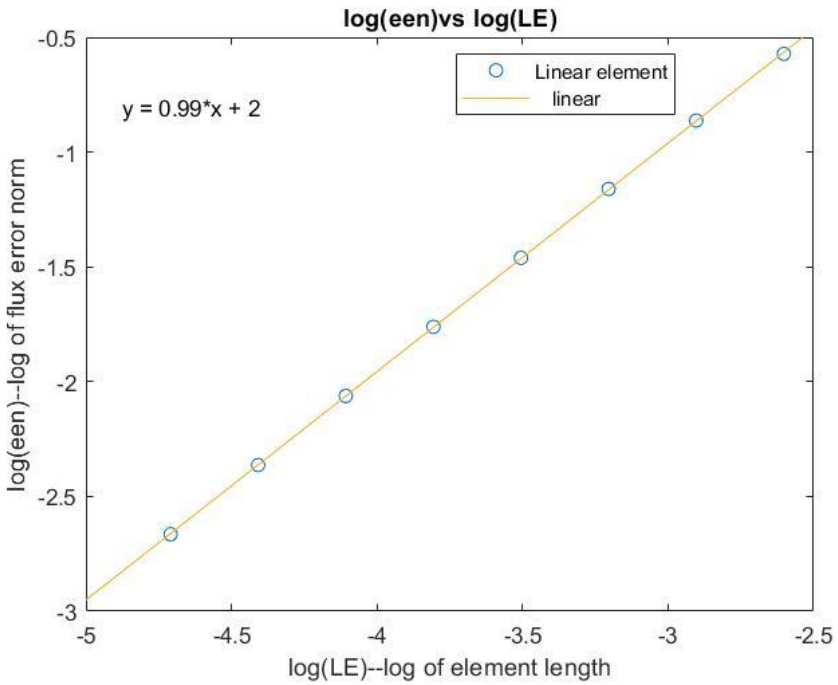


Figure 18: Flux Error Norm vs. Element Size for Linear Element

$$Energy\ Error\ Norm\ Function, \log(e_{en}) = C + \alpha \log(h)$$

$$e_{en} - Energy\ Error\ Norm\ C - Arbitrary\ Constant\ \alpha - Rate\ of\ Convergence\ h - Element\ Size$$

$$L2\ Error\ Norm\ Equatiion\ From\ Plot, \log(e_{L2}) = 3.3 + 0.99 \log(h)$$

Comparing both the Equation, we obtain the convergence Rate $\alpha = 0.99$. *As per the general Mathematical Literature Theory for Linear element, the rate of convergence for energy error norm is one less than the L2 Error norm, as Energy Error norm being it derivative and therefore for linear element Theoretically* $\alpha = 1$. *So here we can see the rate of convergence is from plot equation is approximately equal to expected theoretical Value.*

The plot of L2 temperature error norm vs. element size on and rate of convergence using quadratic elements.

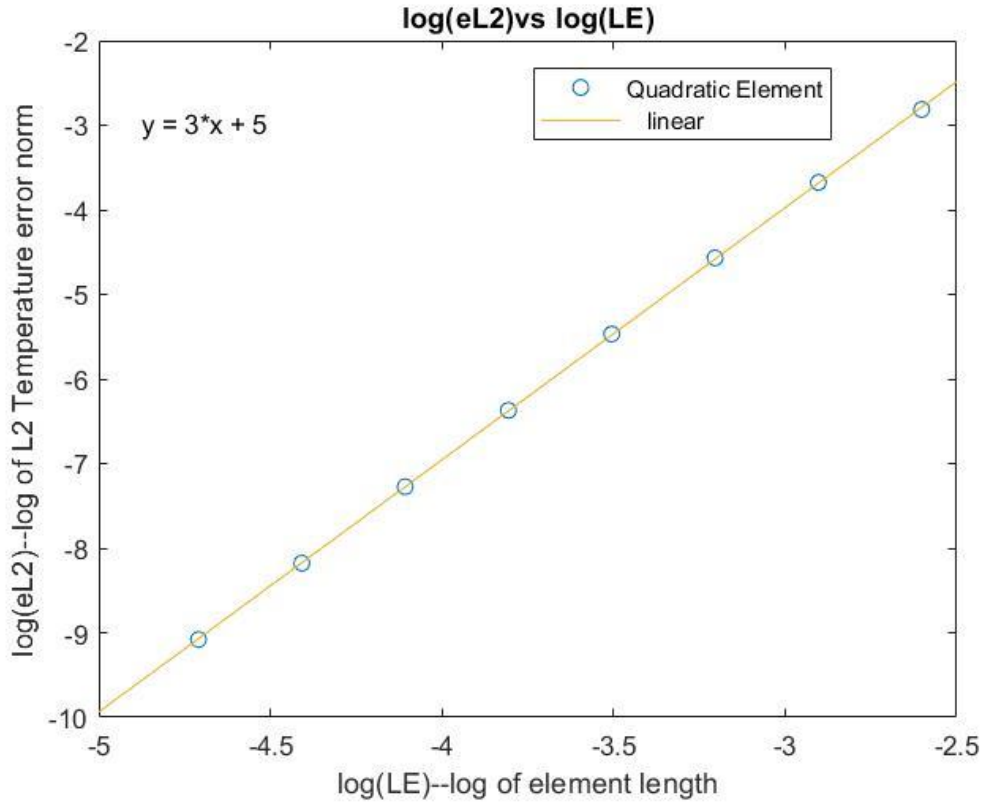


Figure 19: L2 Temperature Error Norm vs. Element size for Quadratic Element

$$2\ Error\ Norm\ Function, \log(e_{L2}) = C + \alpha \log(h)$$

$$e_{L2} - Error\ Norm\ C - Arbitrary\ Constant\ \alpha - Rate\ of\ Convergence\ h - Element\ Size$$

$$L2\ Error\ Norm\ Equatiion\ From\ Plot, \log(e_{L2}) = 5 + 3\log(h)$$

Comparing both the Equation we obtain the convergence Rate $\alpha = 3$ As per the general Mathematical Literature Theory for quadratic element, the rate of convergence is equal to 2, and our result is accurately matching.

The plot of Flux error norm vs. element size on and rate of convergence using quadratic elements.

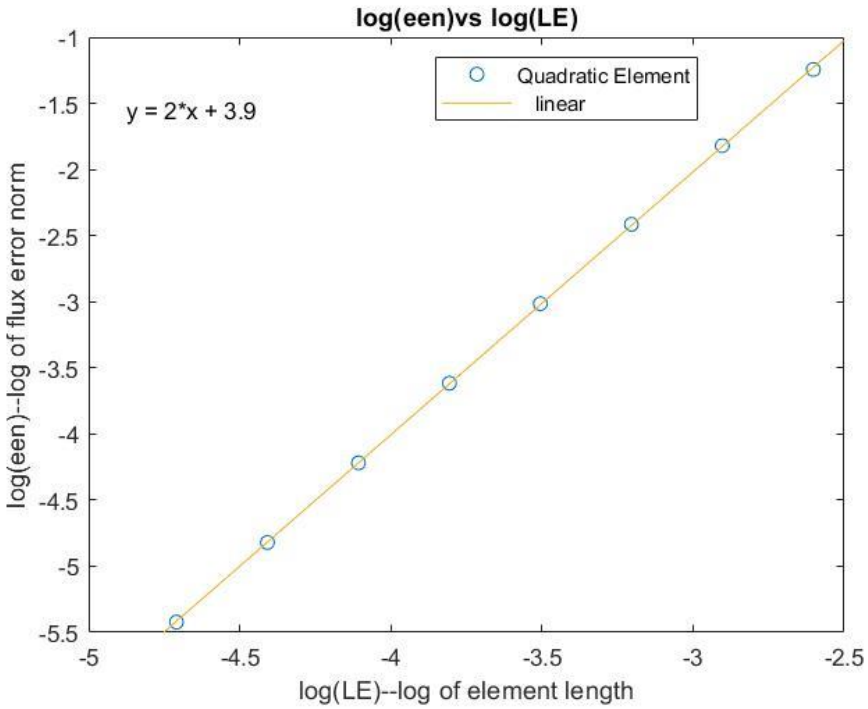


Figure 20: Flux Error Norm vs. Element Size for Quadratic Element

$$Energy\ Error\ Norm\ Function, \log(e_{en}) = C + \alpha \log(h)$$

$$e_{en} - Energy\ Error\ Norm\ C - Arbitrary\ Constant\ \alpha - Rate\ of\ Convergence\ h - Element\ Size$$

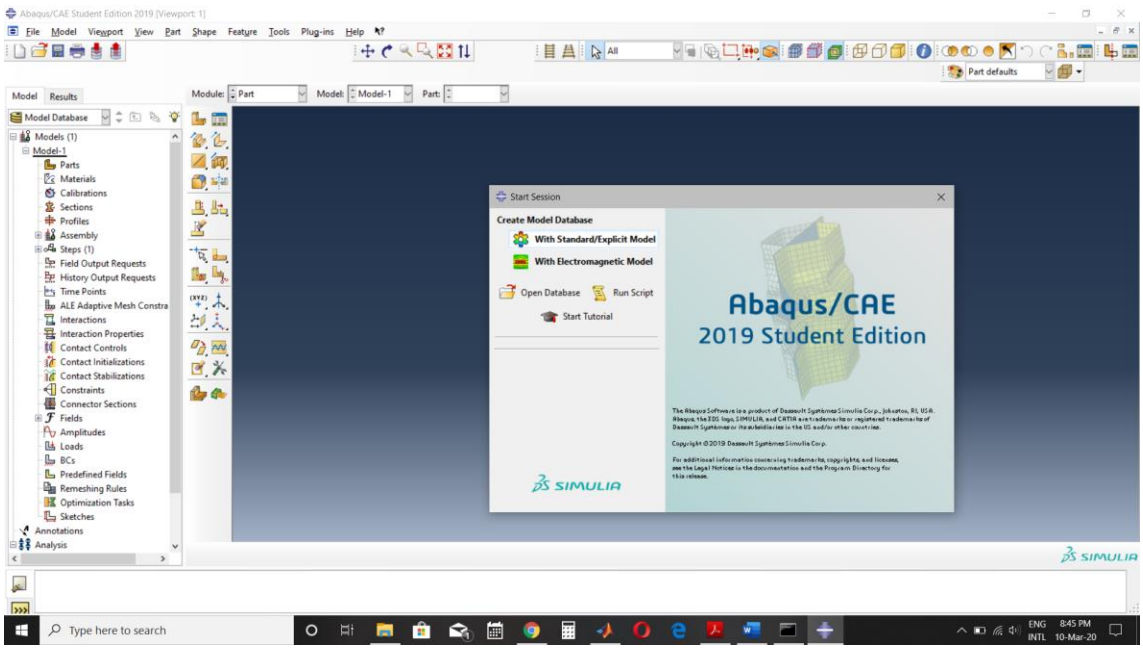
$$L2\ Error\ Norm\ Equatiion\ From\ Plot, \log(e_{L2}) = 3.9 + 2\log(h)$$

Comparing both the Equation, we obtain the convergence Rate $\alpha = 2$ *As per the general Mathematical Literature Theory for Quadratic element, the rate of convergence for energy error norm is* $\alpha = 1$. *So, we can see that the rate of convergence from the plot equation is equal to the expected theoretical value.*

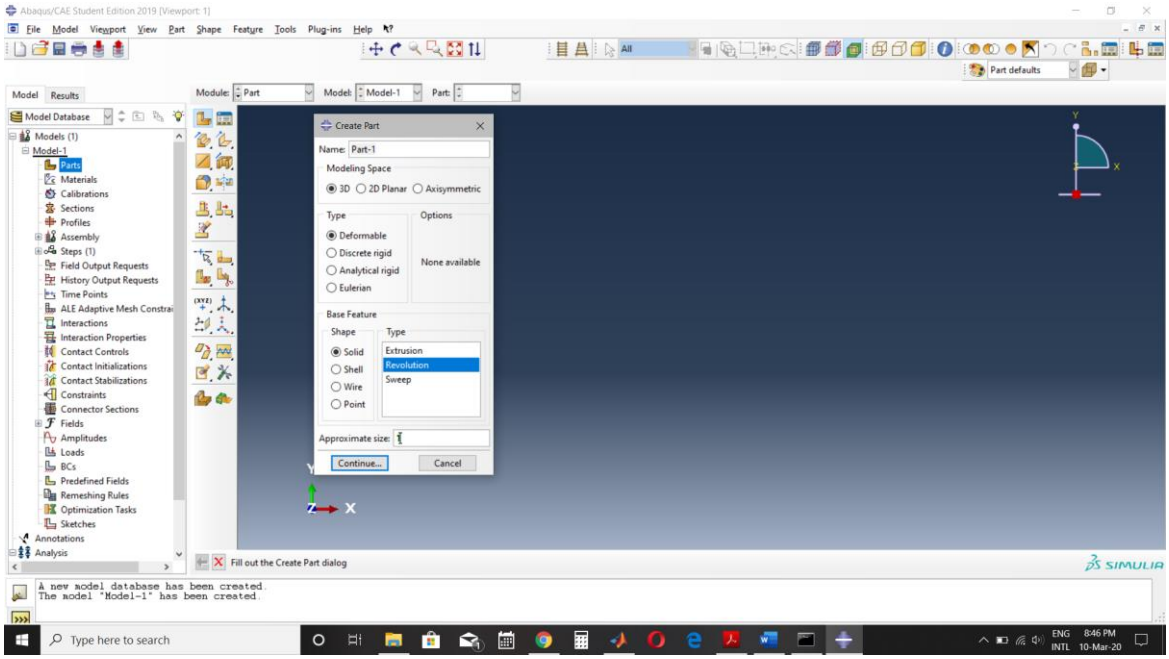
Abaqus Solution

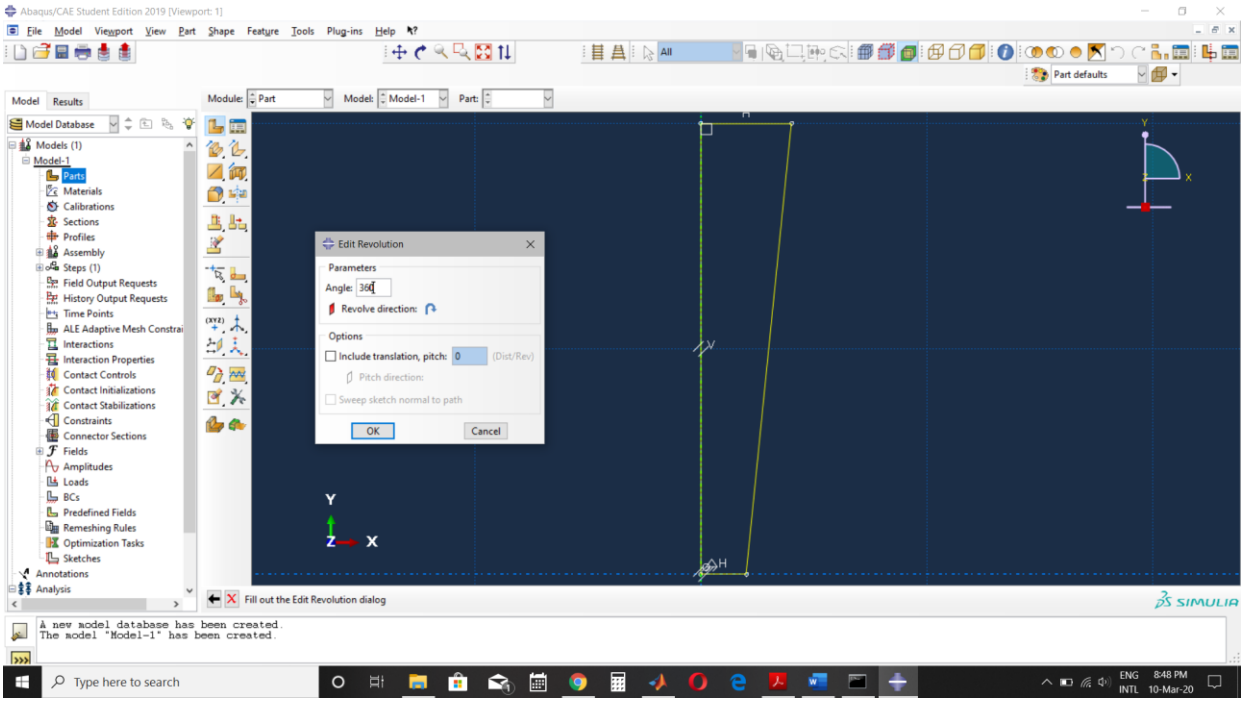
Procedure

Step 1: Select Model – Standard/Explicit Model

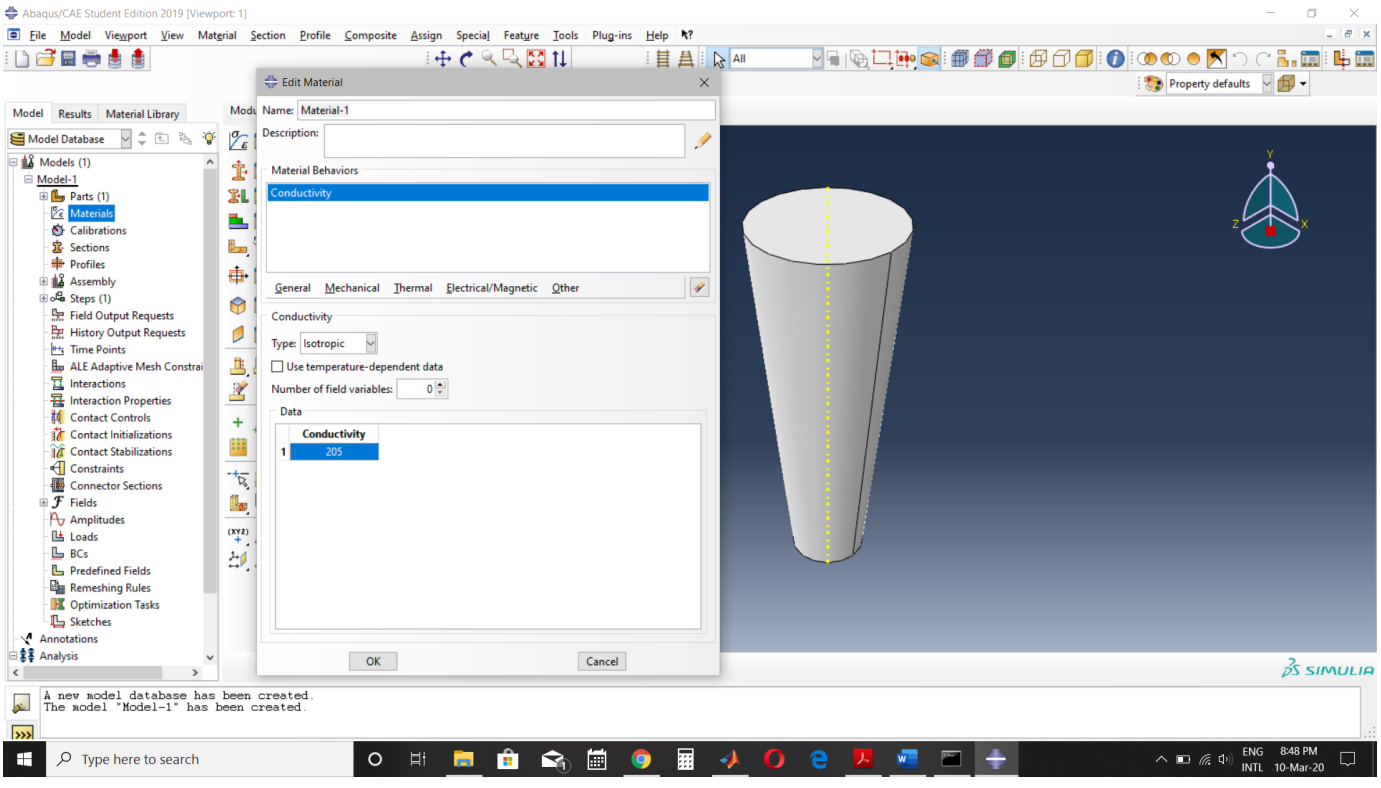


Step 2: Create Part

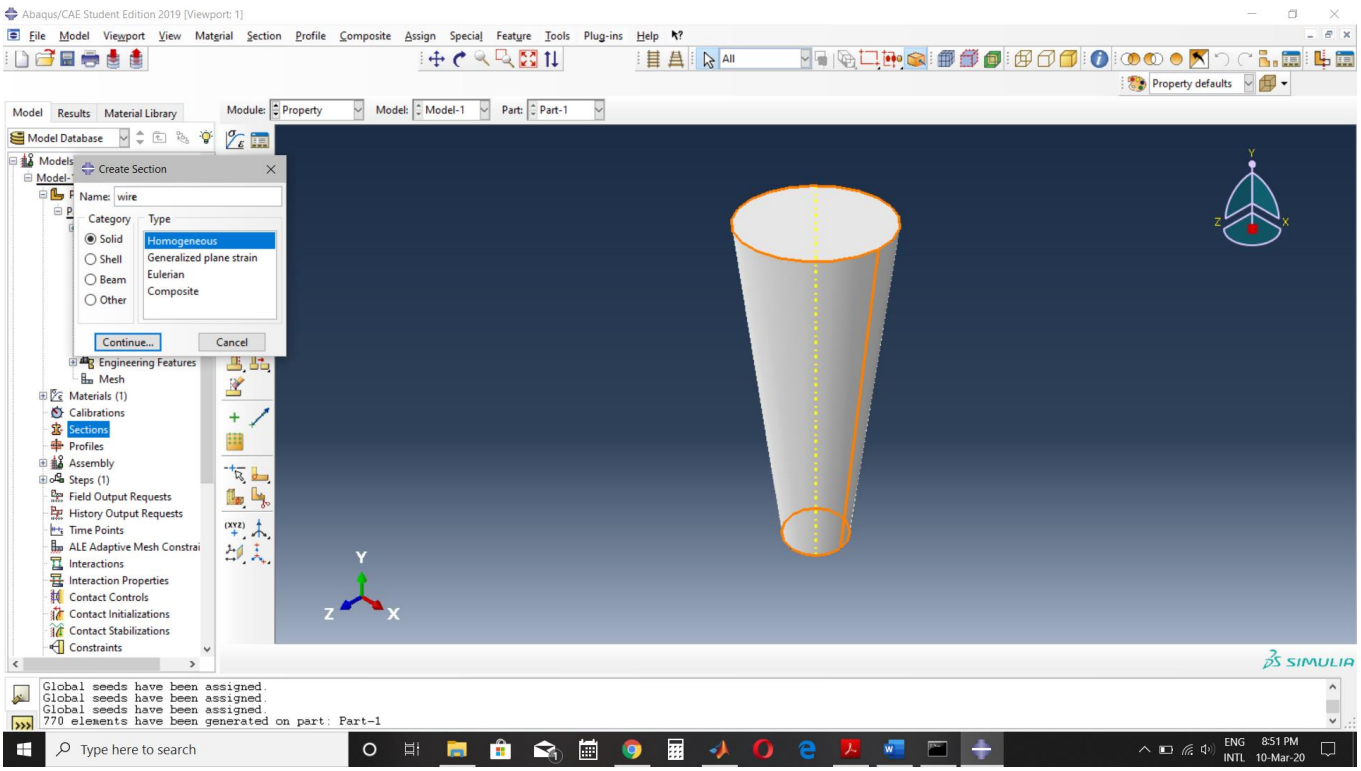




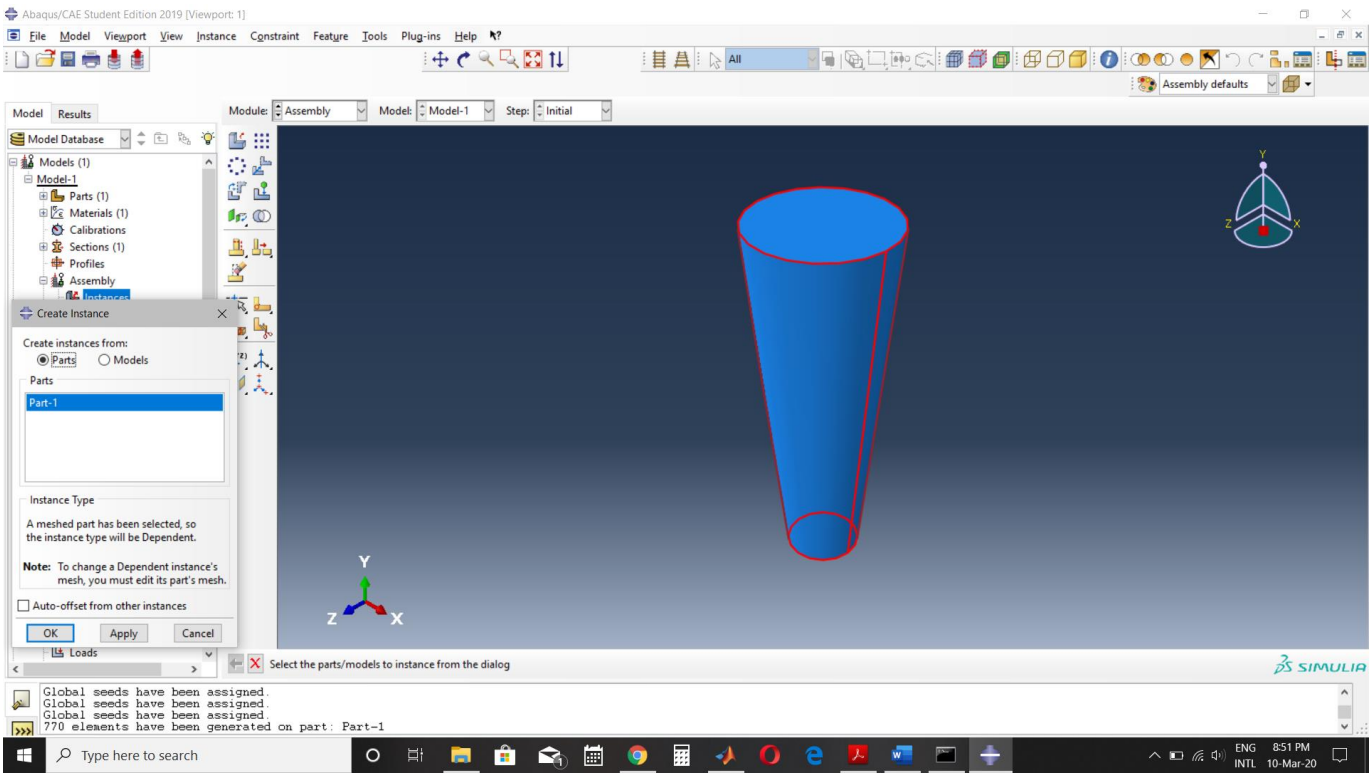
Step 3: Create Material



Step 4: Create Section- To assign material

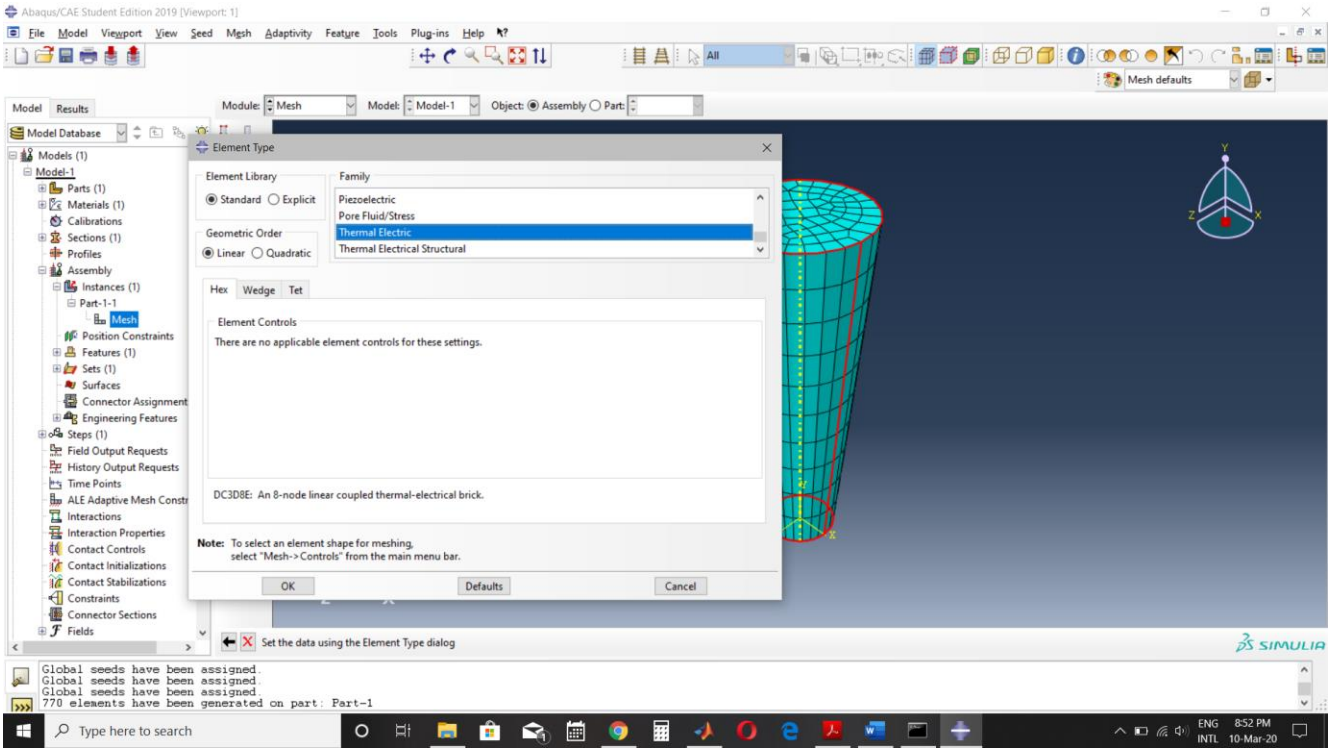


Step 5: Create Instance

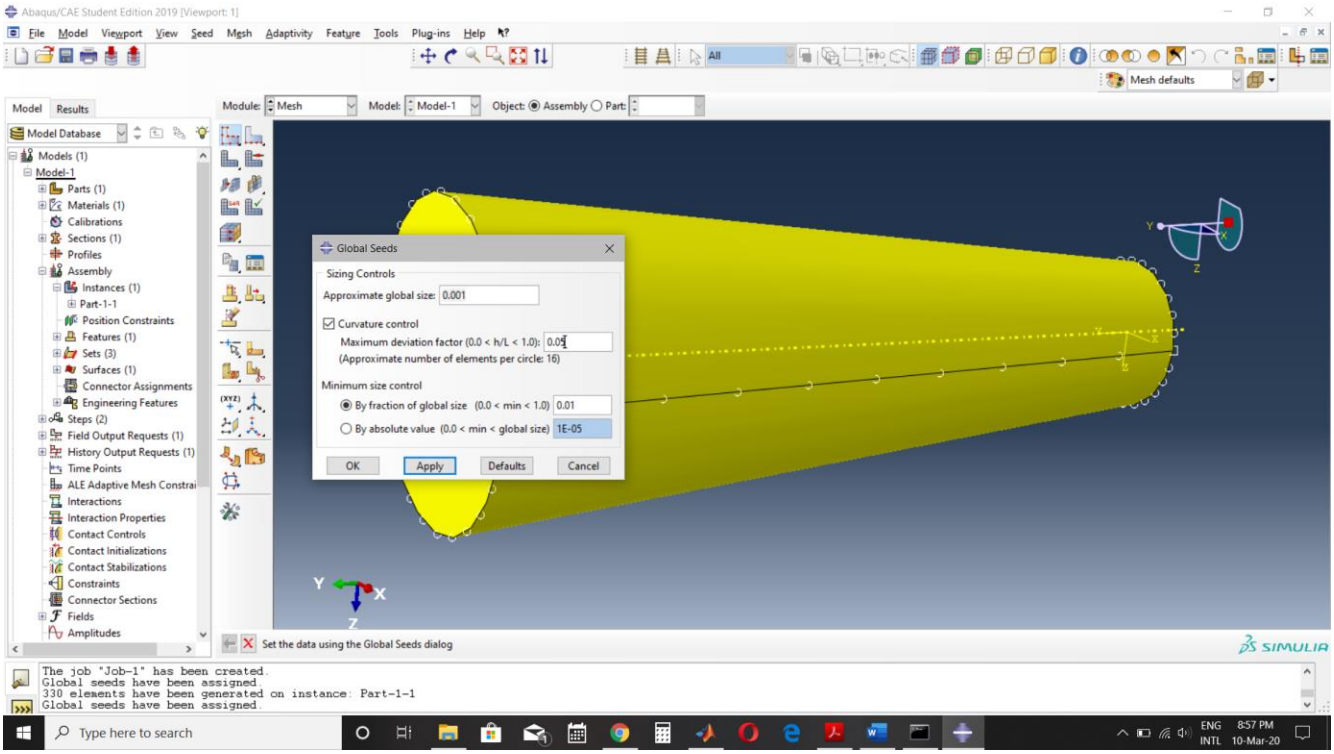


Step 6: Meshing

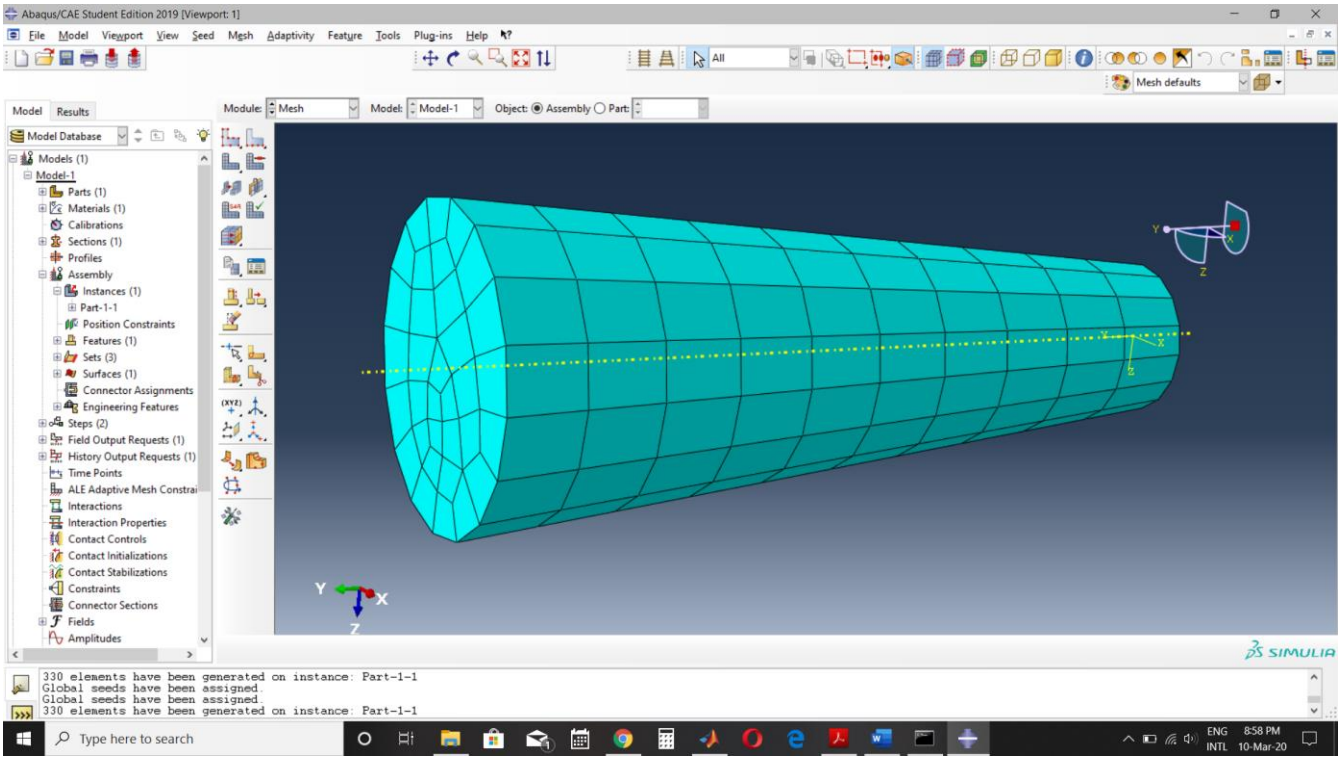
Step 6.1: Assign Element Type and Family



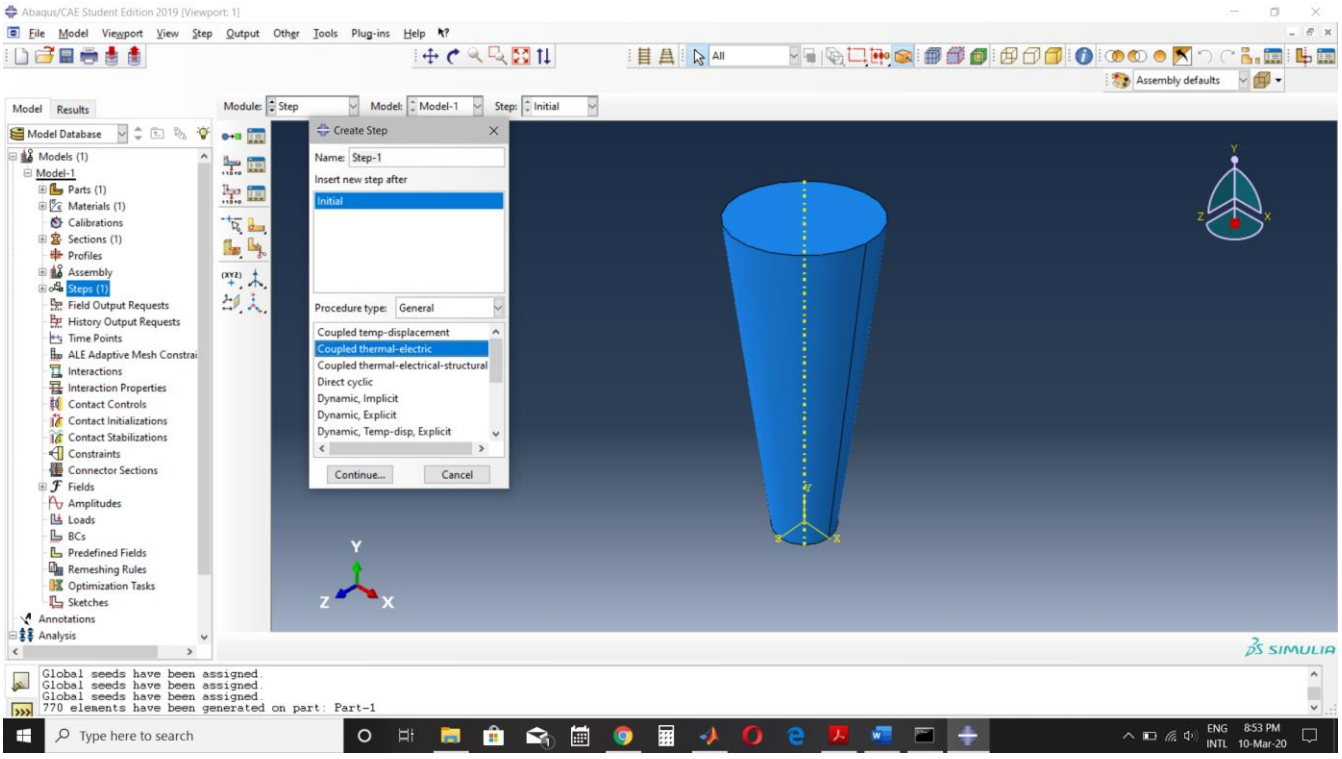
Step 6.2: Global Seed – To define the Number of Elements



Step 6.3: Create Mesh

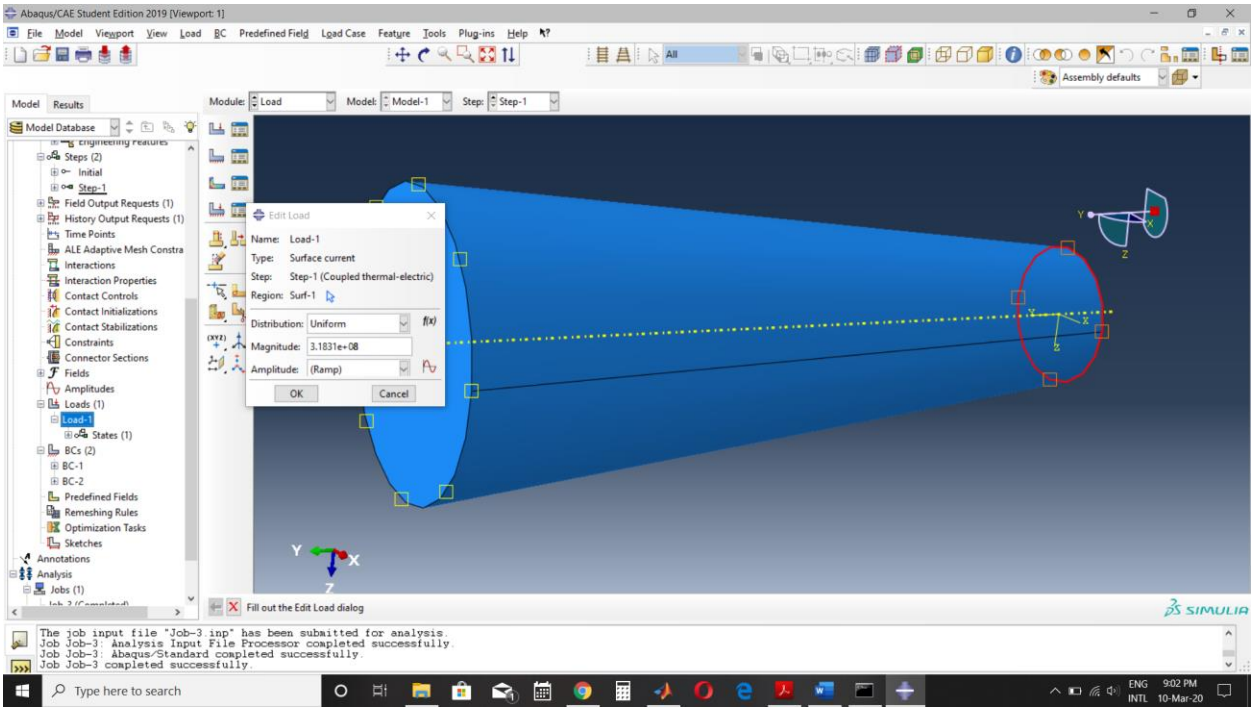


Step 7: Create Step – Procedure Type – Coupled Thermal Electric



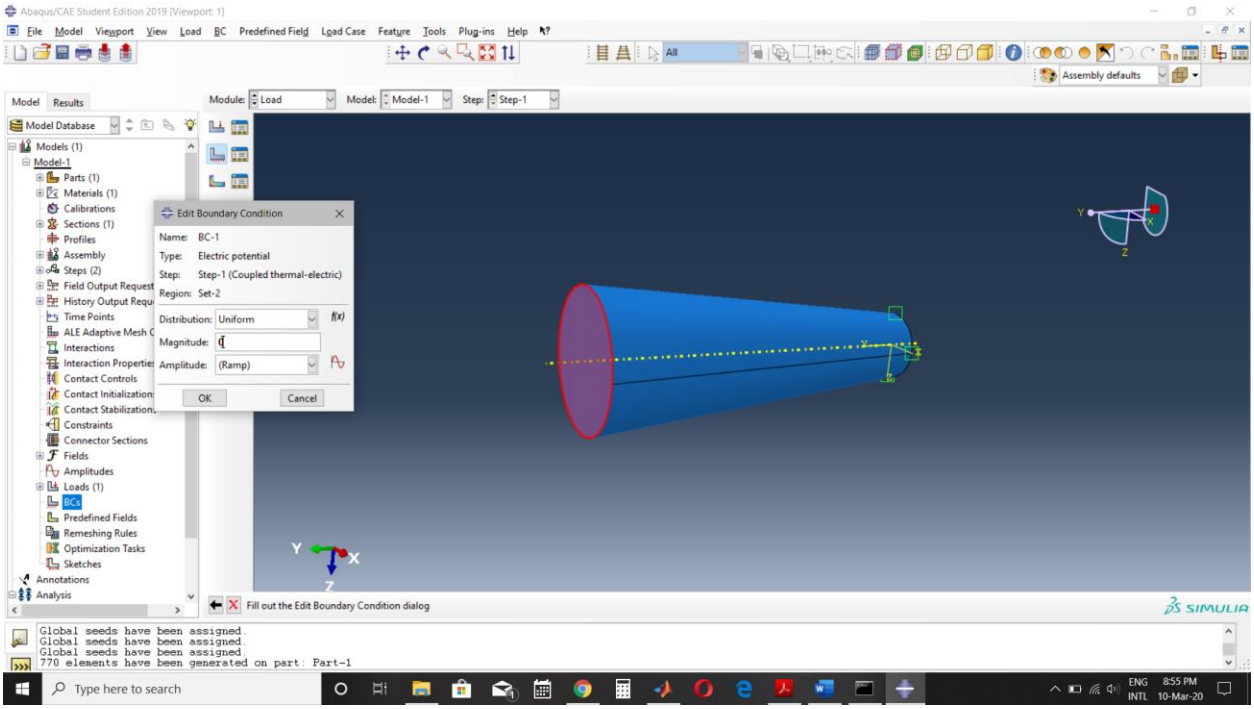
Step 8: Define Load and Boundary Condition

Load 1 Surface Current at x=0



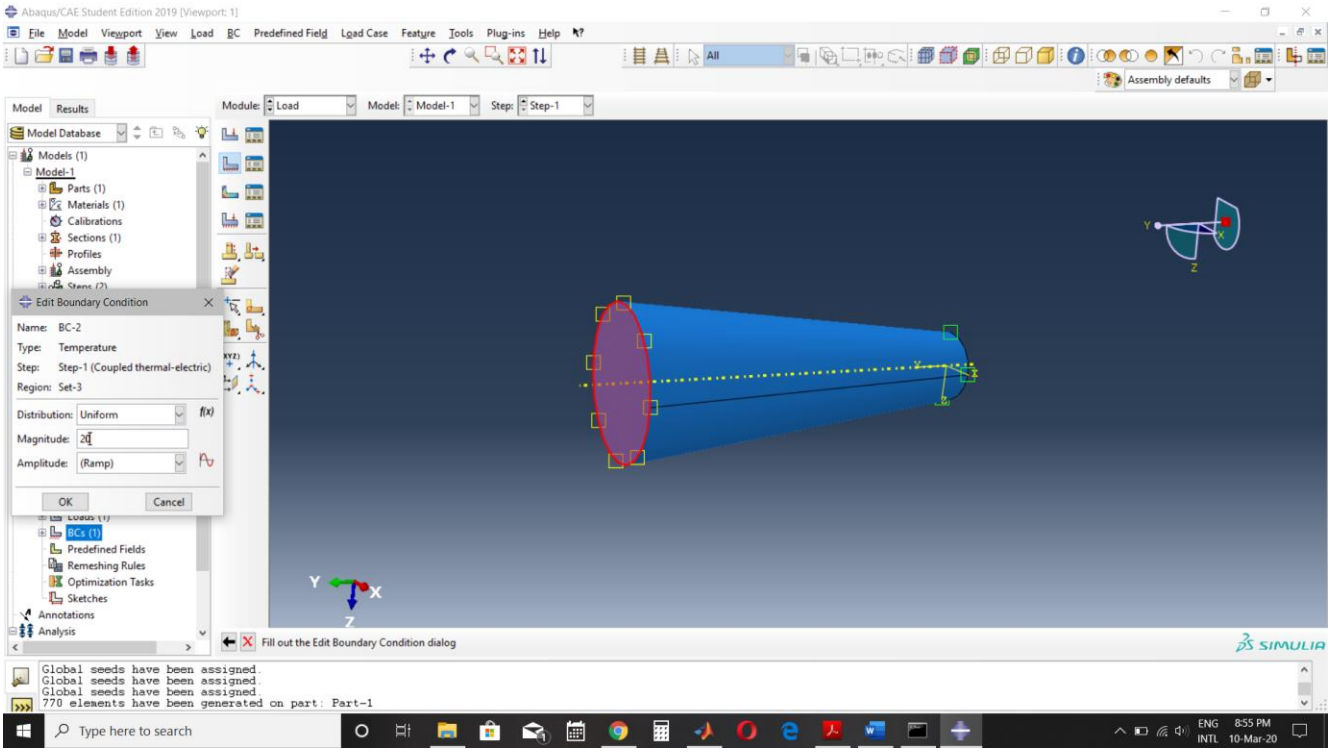
Boundary Condition 1

Electric Potential Zero at x=L as to define the current direction from x=0 towards x=L.

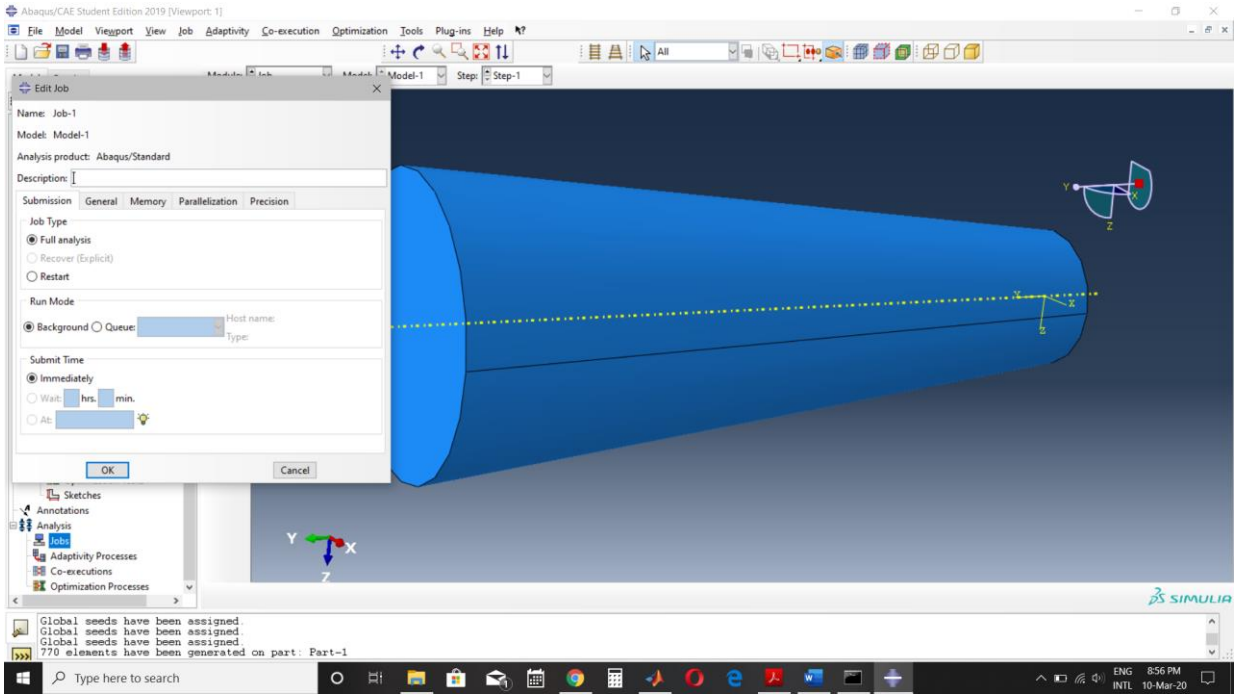


Boundary Condition 2

Temperature T=20^{0}C at x=L (defined in problem statement)

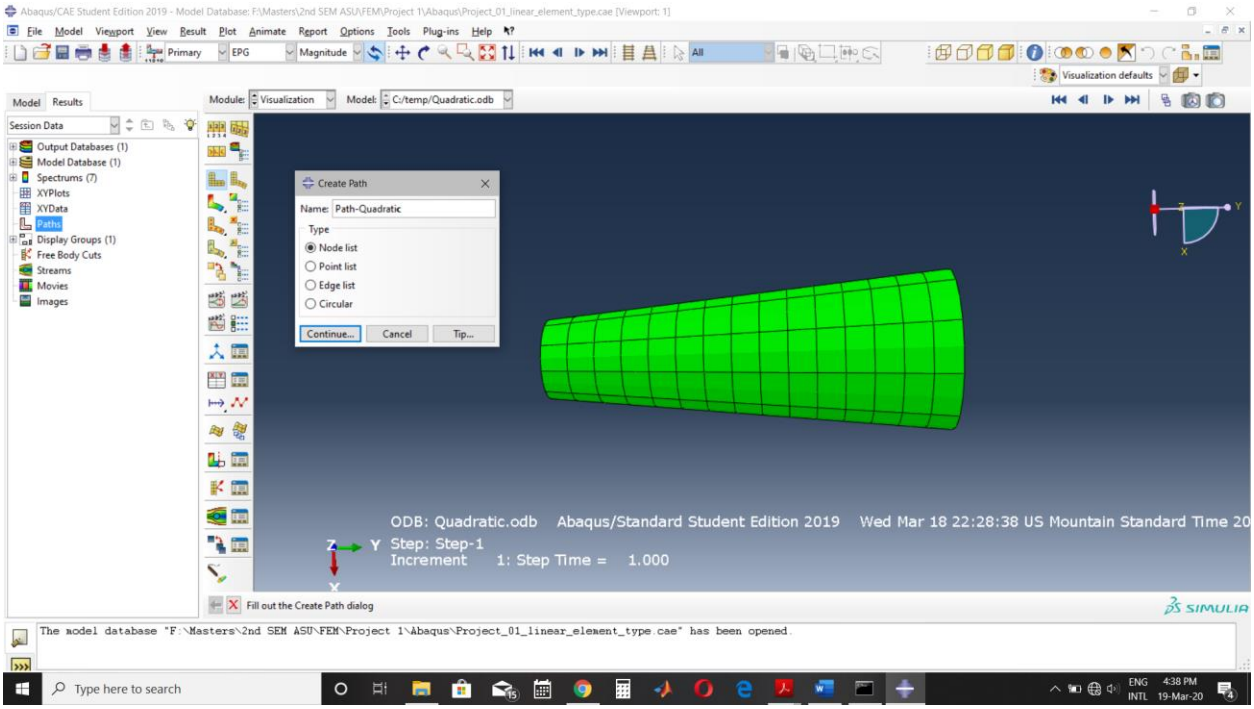


Step 9: Creating a Job and Submitting it

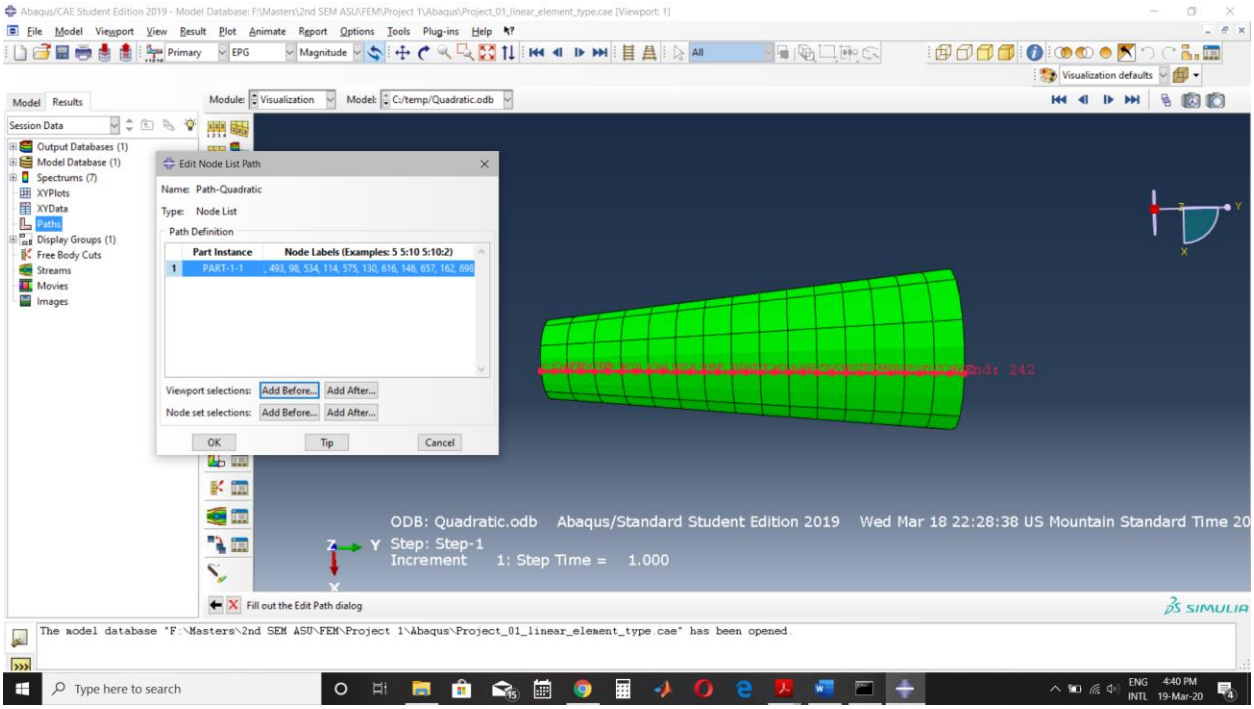


Step 10: Post Processing

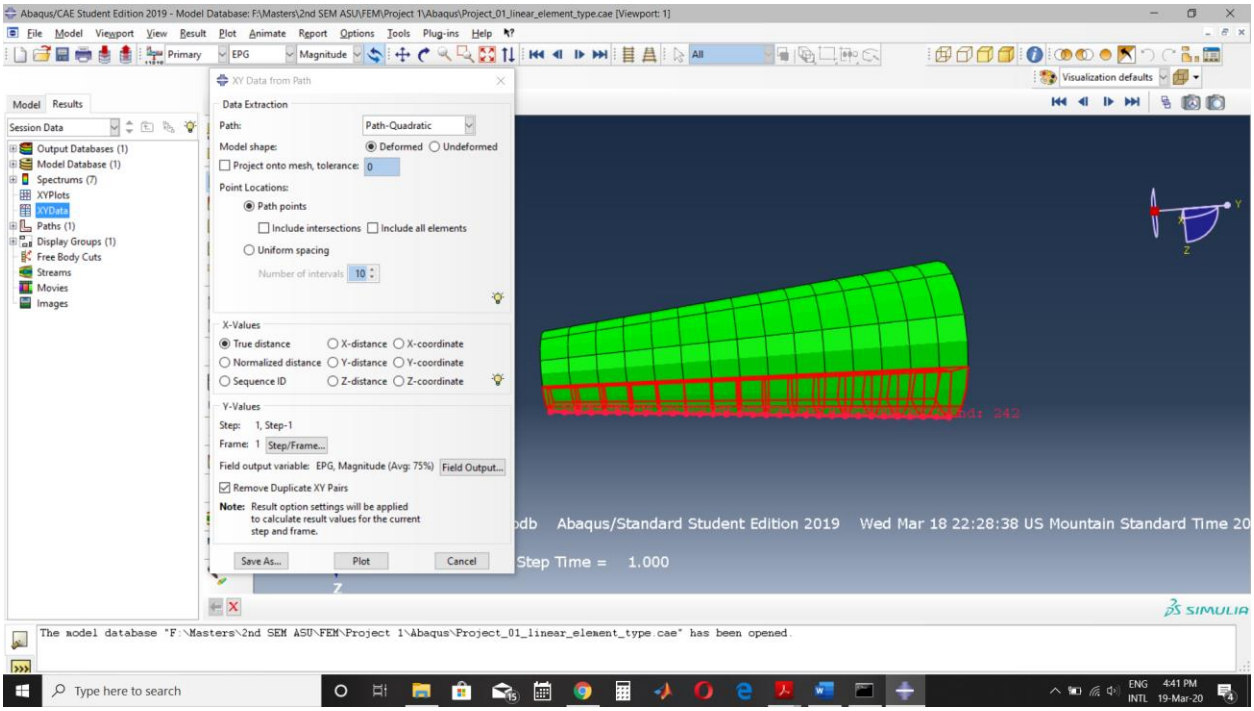
Step 10.1: Create Path



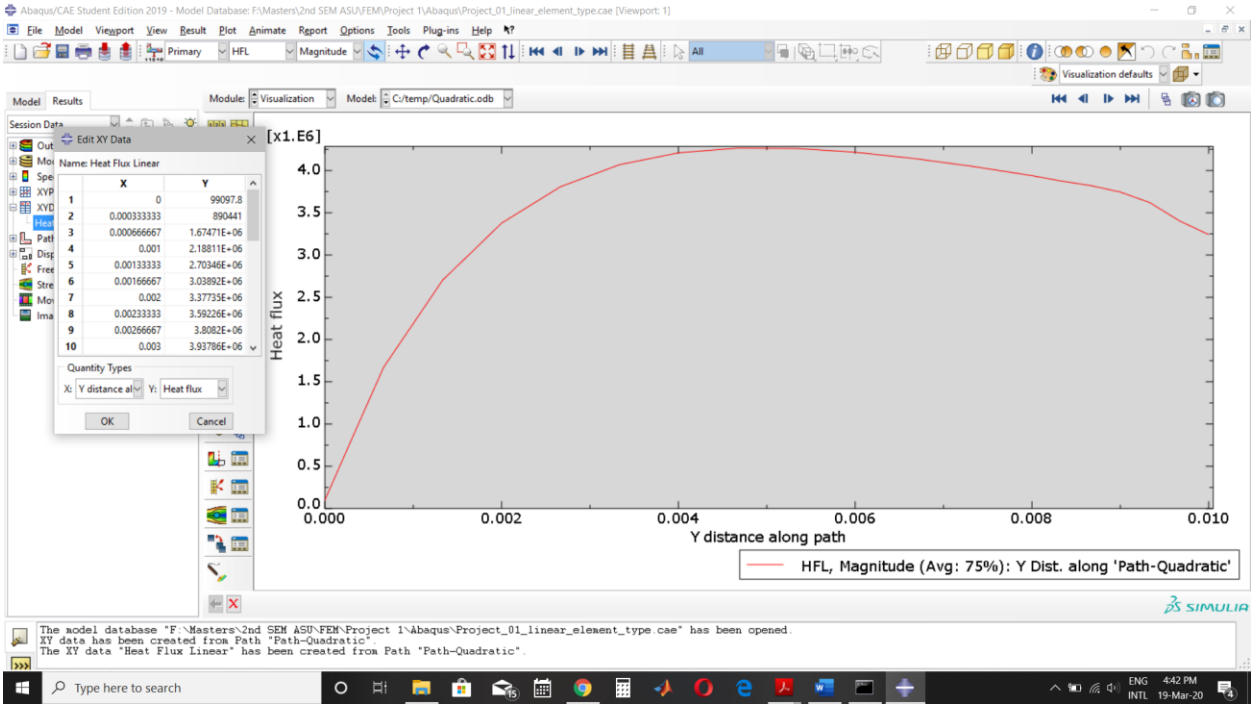
Step 10.2: Adding Node List



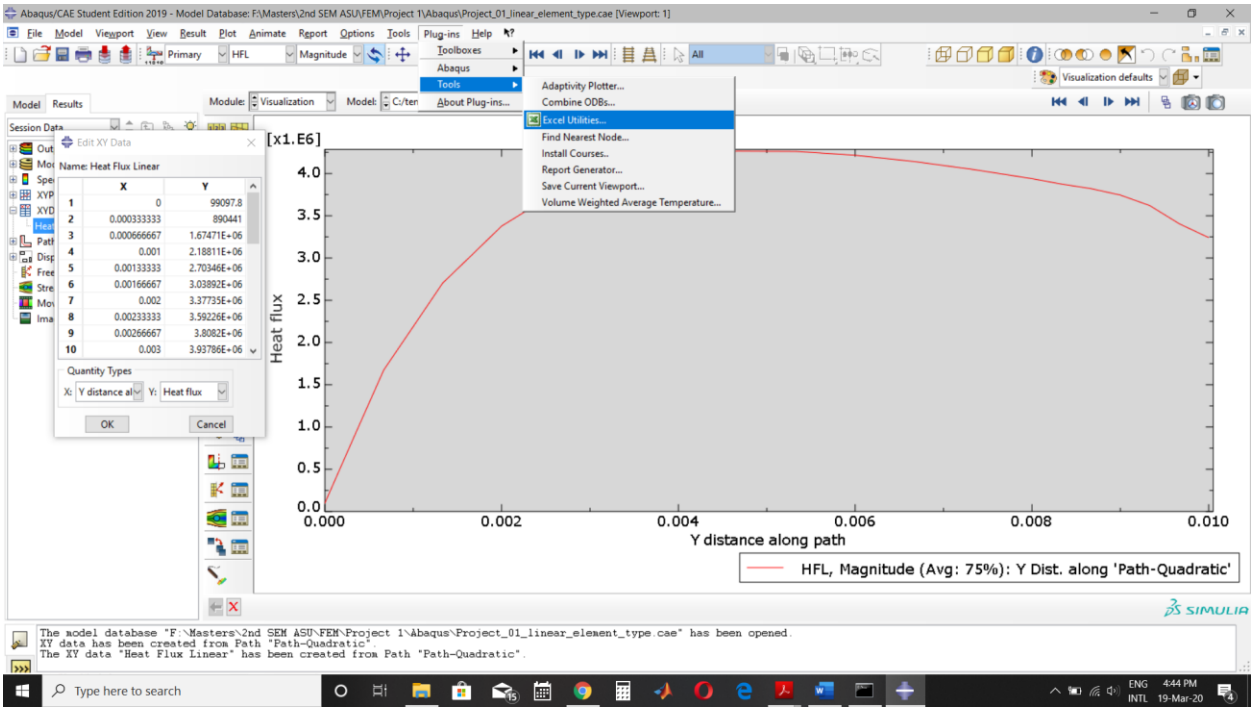
Step 10.3 Create XY Data



Step 10.4 Plot and Save Data



Step 11: Save Data in excel using Plug-in tools



Plot temperature and heat flux fields for linear elements

Temperature(^{0}C) and Distance (m)

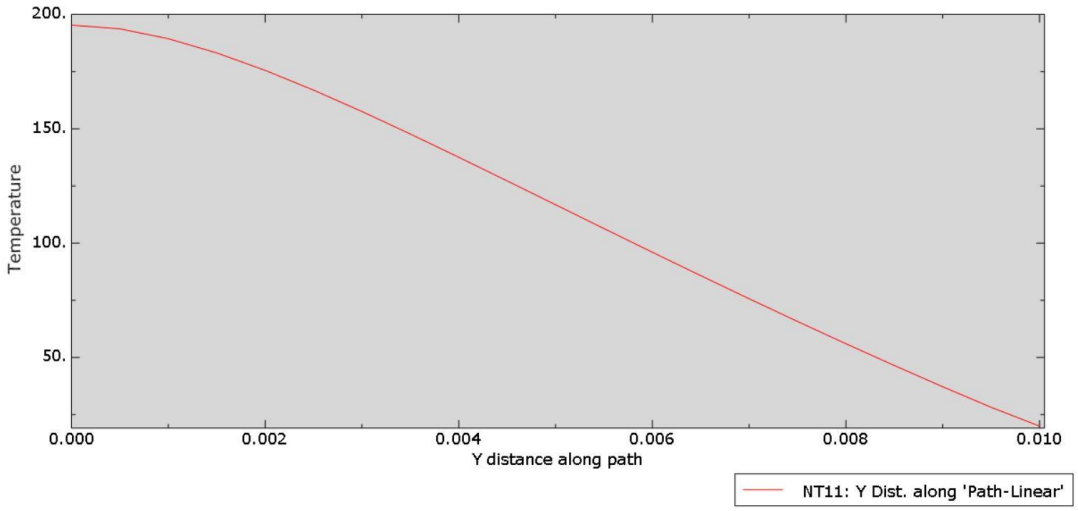


Figure 21: Temperature Field Abaqus Solution Linear Element

Data Table of Linear Element Temperature vs. Distance from Abaqus

Distance (m)	Temperature (^{0}C)
0	195.3227
0.0005	193.6992
0.001	189.3965
0.0015	183.1787
0.002	175.5553
0.0025	166.9175
0.003	157.5625
0.0035	147.7178
0.004	137.5581
0.0045	127.2176
0.005	116.7996
0.0055	106.3835
0.006	96.02995
0.0065	85.78525
0.007	75.68442
0.0075	65.75407
0.008	56.01518
0.0085	46.48763
0.009	37.20089
0.0095	28.21306
0.01	20

Heat Flux (W/m^2) and Distance (m)

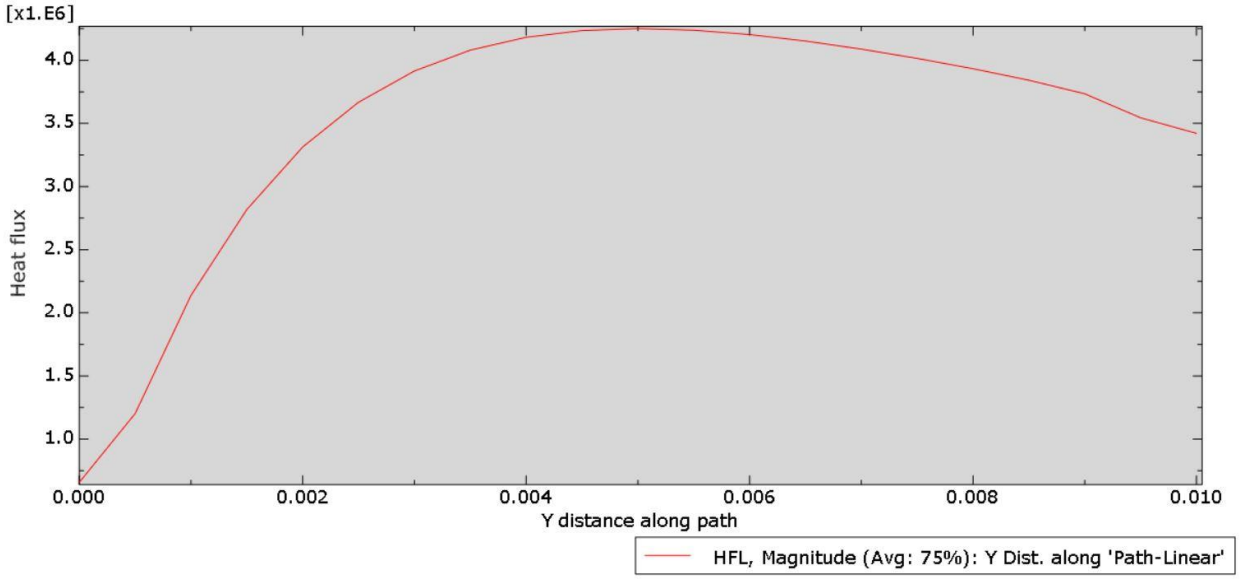


Figure 22: Heat Flux Field Abaqus Solution Linear Element

Data Table of Linear Element Heat Flux vs. Distance from Abaqus

Distance (m)	Heat Flux (W/m^2)
0	656675.5
0.0005	1201507
0.001	2137933
0.0015	2817713
0.002	3312929
0.0025	3667496
0.003	3914773
0.0035	4079800
0.004	4181640
0.0045	4234883
0.005	4250768
0.0055	4237997
0.006	4203333
0.0065	4152031
0.007	4088130
0.0075	4014545
0.008	3932819
0.0085	3841683
0.009	3734352
0.0095	3544114
0.01	3421090

Plot temperature and heat flux fields for quadratic elements

Temperature(^{0}C) and Distance (m)

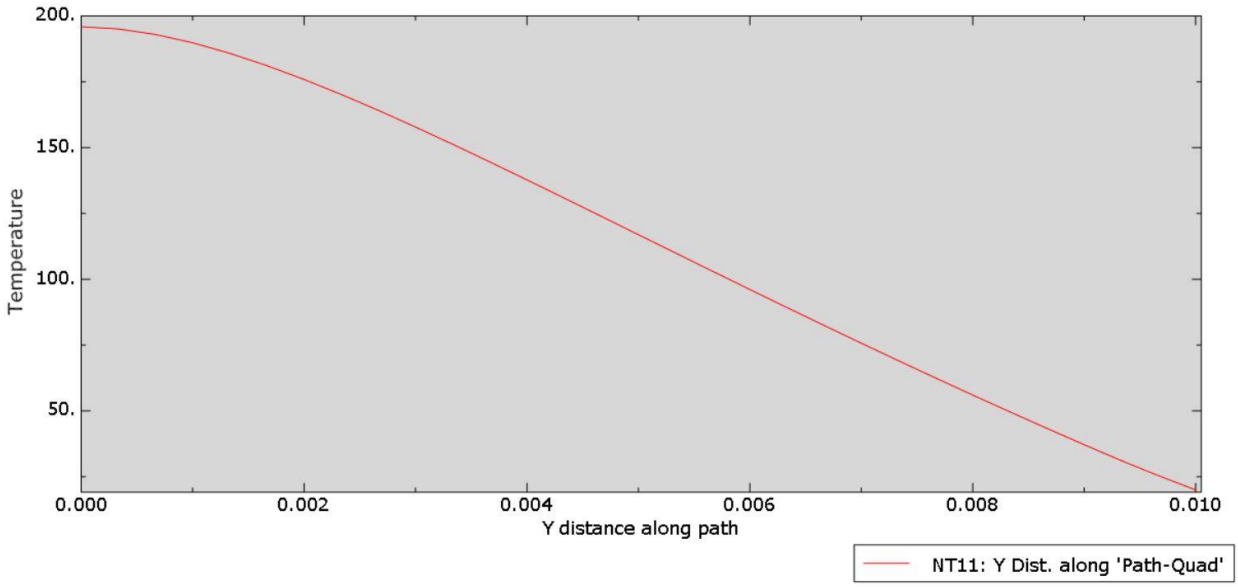


Figure 23: Temperature Field Abaqus Solution Quadratic Element

Data Table of Quadratic Element Temperature vs Distance from Abaqus

Distance (m)	Temperature(^{0}C)	Distance (m)	Temperature(^{0}C)
0	195.8694	0.007	75.72885
0.000333	195.0627	0.007333	69.07631
0.000667	192.9594	0.007667	62.50652
0.001	189.8108	0.008	56.02571
0.001333	185.8082	0.008333	49.64165
0.001667	181.1222	0.008667	43.36214
0.002	175.8763	0.009	37.17207
0.002333	170.1852	0.009333	31.12393
0.002667	164.136	0.009667	25.41624
0.003	157.8096	0.01	20
0.003333	151.2662		
0.003667	144.5628		
0.004	137.7415		
0.004333	130.8428		
0.004667	123.8961		
0.005	116.93		
0.005333	109.9652		
0.005667	103.0218		
0.006	96.11397		
0.006333	89.25604		
0.006667	82.45769		

Heat Flux (W/m^2) and Distance (m)

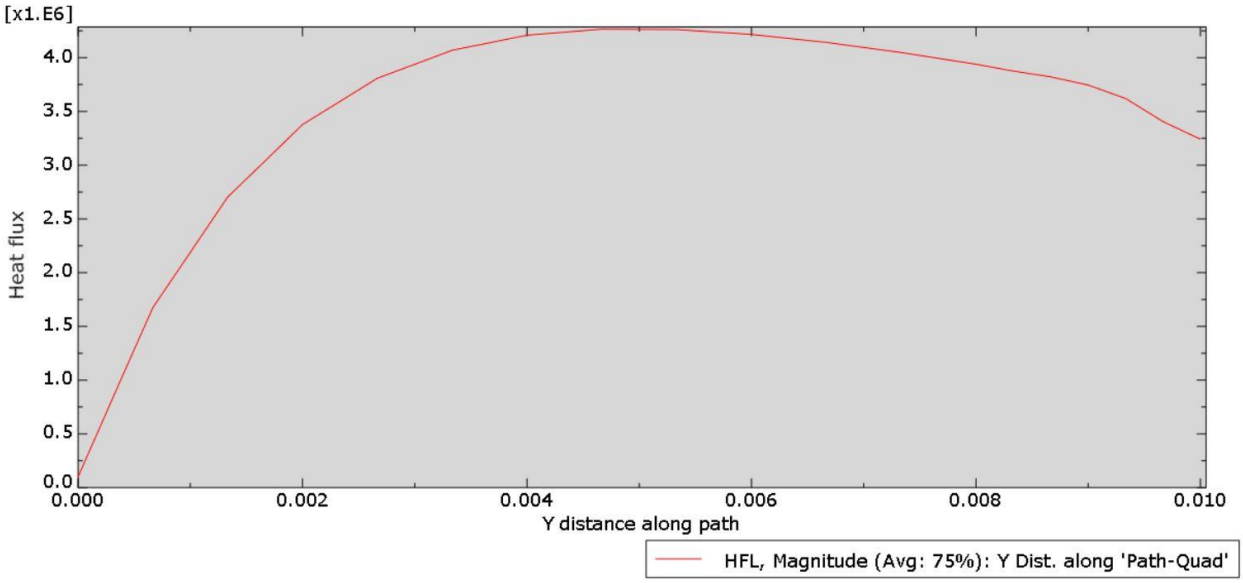


Figure 24: Heat Flux Field Abaqus Solution Quadratic Element

Data Table of Quadratic Element Heat Flux vs. Distance from Abaqus

Distance (m)	Heat Flux (W/m^2)	Distance (m)	Heat Flux (W/m^2)
0	99097.76	0.005667	4238149
0.000333	890441.4	0.006	4215635
0.000667	1674714	0.006333	4178506
0.001	2188106	0.006667	4141547
0.001333	2703464	0.007	4094395
0.001667	3038924	0.007333	4047651
0.002	3377345	0.007667	3993179
0.002333	3592263	0.008	3938730
0.002667	3808199	0.008333	3874746
0.003	3937858	0.008667	3821064
0.003333	4068356	0.009	3744635
0.003667	4138238	0.009333	3620101
0.004	4208660	0.009667	3405143
0.004333	4236380	0.01	3241133
0.004667	4264484		
0.005	4262540		
0.005333	4260864		

From the above plot from Abaqus, we can see that the Quadratic element has better approximation then Linear elements.

Plot temperature fields for analytical, ABAQUS, and MATLAB solutions all on the same plot.

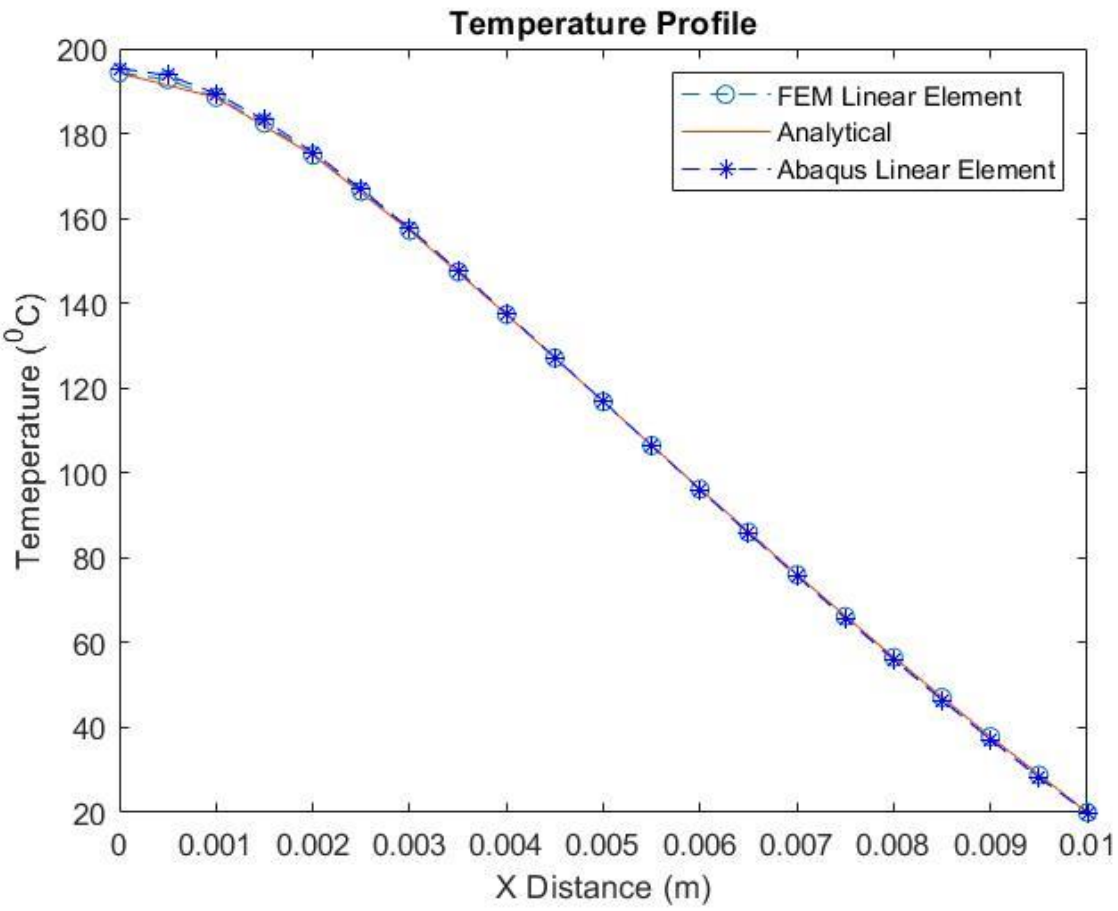


Figure 25: Comparison of Temperature vs. Distance FEM Abaqus and Analytical Solution with Linear Elements

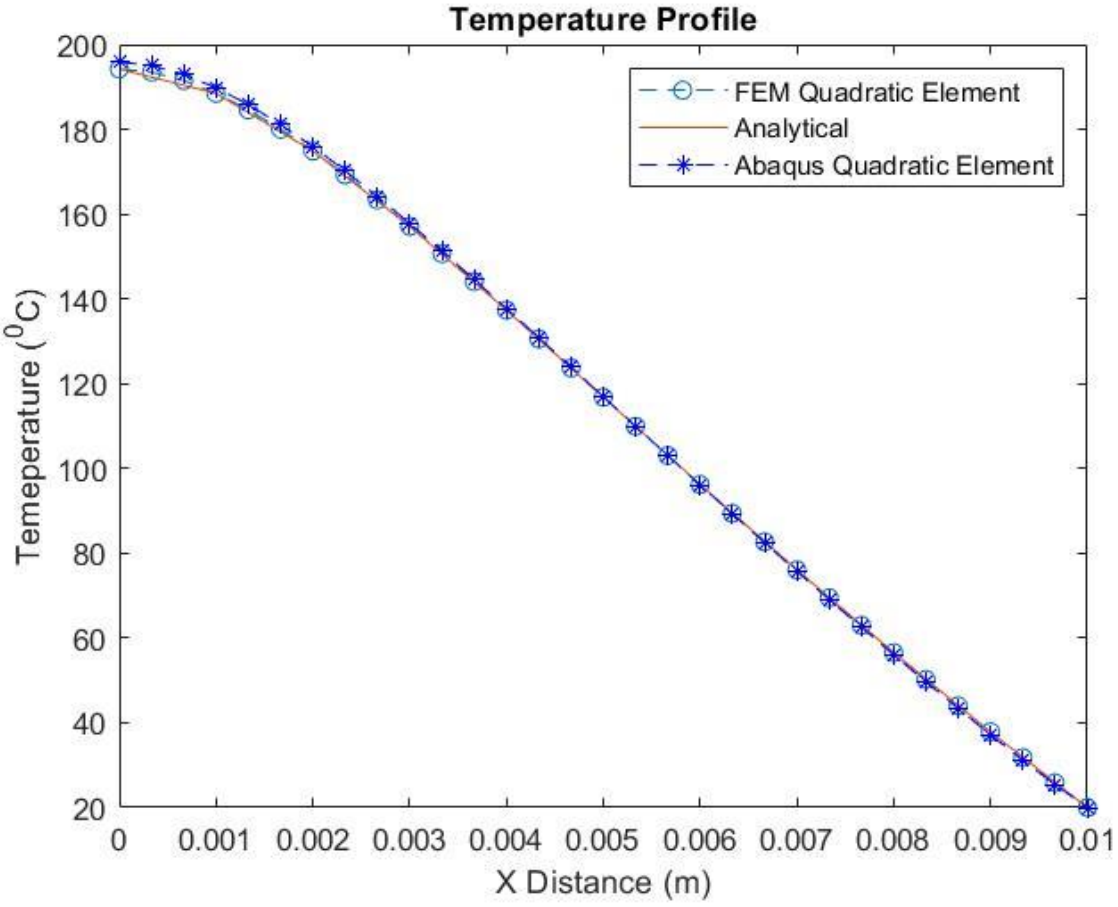


Figure 26: Comparison of Temperature vs. Distance FEM Abaqus and Analytical Solution with Quadratic Elements

The temperature field from all three-solution method are approximately equal, and we can observe in case of quadratic element fit is better than the linear element.

Plot heat flux fields for analytical, ABAQUS and MATLAB solutions all on the same plot.

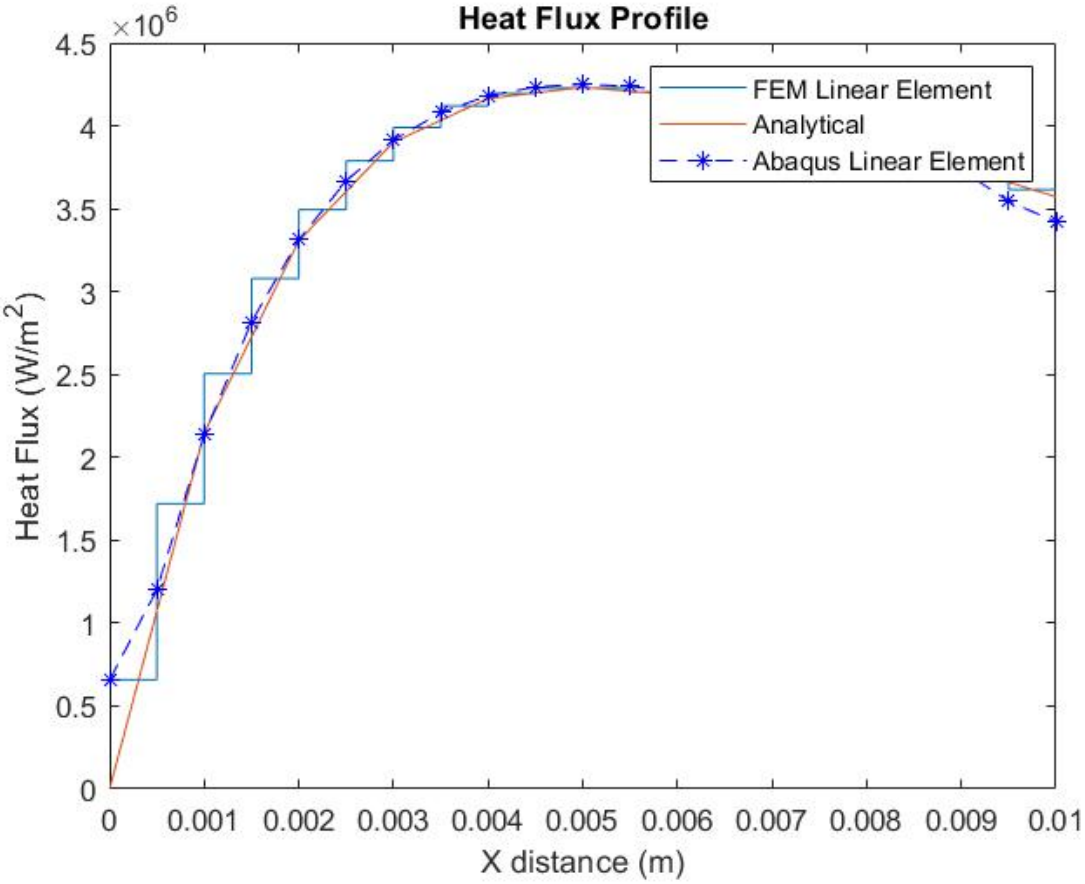


Figure 27: Comparison of Heat Flux vs. Distance FEM Abaqus and Analytical Solution with Linear Elements

In the case of the linear element, we can see that the FEM plot is a step type function as the variable function (Temperature) is a linear approximation. Thus, its derivative (Heat Flux) within the element is constant, and we can see that the Abaqus result is closer to the analytical solution.

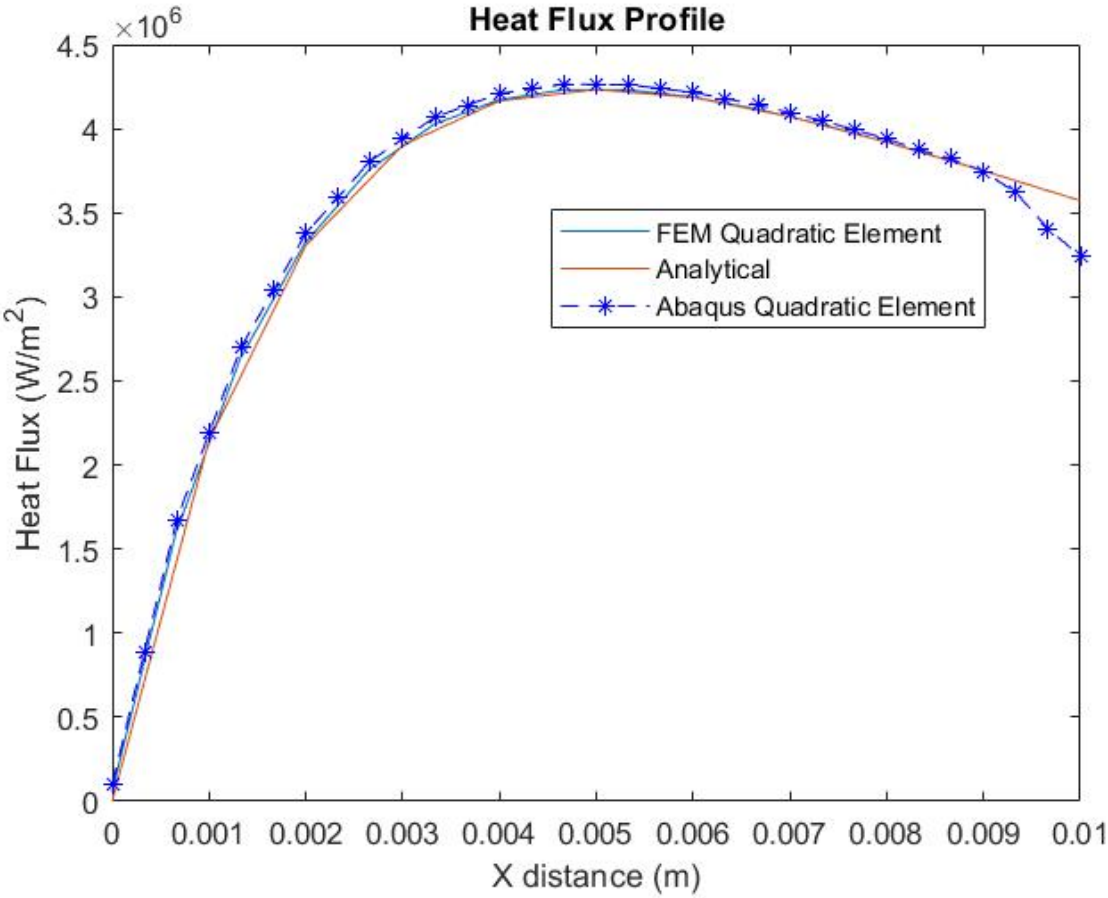


Figure 28: Comparison of Heat Flux vs. Distance FEM Abaqus and Analytical Solution with Quadratic Elements

In the case of the quadratic element, we can see that the FEM plot is a linear function as the variable function (Temperature) is a quadratic approximation in the element. Thus, its derivative (Heat Flux) within the element is a linear function, and we can see that now both the FEM and Abaqus results are closer to the analytical solution.

Steady-State Temperature in Rob with Temperature-Dependent Electrical Resistivity (Extra Project)

Approach to solve the problem

$$Electrical\ Resitivity\ as\ a\ function\ of\ Temperature, \rho(T) = 2.60e^{-8} + 1.1e^{-10}T\ \Omega m$$

The approach to solving this problem is through an iterative process in the Finite Element Method. Here we can iterate by following steps:

1. Consider the initial temperature of rob as 20^0C, and we can get initial electrical resistivity at each node.
2. Now the heat source is also dependent on Electrical Resistivity (s(x, rho)).
3. As we know, the equation for 'f' now become f=s(x, rho)*N*J*w. We must use the value of the electrical resistivity of that point in the element. Therefore, we formed a matrix that contains the value of electrical resistivity at each node.
4. From the first iteration, we get a temperature value at each node, and we find the error with the initial temperature value at each node.
5. At this same time, we find the electrical resistivity value with this temperature using the above equation at each node.
6. Now we find an error in temperature in each node.
7. And now we re-iterate with new electrical resistivity and initial temperature (Determined in the previous iteration)
8. Now we find the new temperature at nodes using new electrical resistivity values.
9. Repeat the process 6-8 until the required error criteria are reached on each node.

$$Convergence\ Criteria\ (at\ each\ node) = \frac{New\ Temprature - Old\ Temperature}{New\ Temperature} \leq 0.00001$$

MATLAB Code

```
function [fe,K,T,I_max] =
tdhanote_Project01_FEM_Variable_Electical_Resistivity(nn_per_element,number_o
f_elements,k,r0, r1,L,T0,no_qpts)
%% input definition

% nn_per_element = Number of nodes per element
% number_of_elements = Number of elements
% k = Thermal conductivity
% r0 = Radius at the ends of bar
% r1 = radius at the center of bar
% roh_0 = Electrical Resistivity is constant or vaariable
% no_qpts = Number of quadrature Points

%% Inpts
if nargin==0
nn_per_element=4;
number_of_elements=1;
k=205;
r0=0.002;
```

```
r1=0.001;
L=0.01;
I=1000;
no_qpts=10;
T0=20;
end

%% Heat Generation as a function of x

s=@(x,rho)(((I^2)*rho)/(pi()*(r1 + (r0-r1)*(x/L))^2));

%% Arear as a function of x

A=@(x)(pi()*(r1 + (r0-r1)*(x/L))^2);

%% Quadture points selection

qpts=quadrature(no_qpts);

%% Total nodes and connectivity matrix determination and Corresponding shape
function

if nn_per_element==2
    nodes_total=number_of_elements+1;
    conn=[1:number_of_elements;2:number_of_elements+1];
    shape=@shape2;
elseif nn_per_element==3
     nodes_total=2*number_of_elements+1;
     conn=[1:2:2*number_of_elements-
1;2:2:2*number_of_elements;3:2:2*number_of_elements+1];
     shape=@shape3;
else
 conn=[1:3:3*number_of_elements-2;2:3:3*number_of_elements-
1;3:3:3*number_of_elements;4:3:3*number_of_elements+1];
 nodes_total=3*number_of_elements+1;
 shape=@shape4;
end
%% Electrical Resistivity
rho_fun=@(T_in)(2.6e-8+1.1e-10*T_in);
%% Coordiantes of each Nodes
x=linspace(0, L,nodes_total);

%% Stiffness matrix  K Calculation

K=zeros(nodes_total);
for c=conn
    xe=x(:,c);
    Ke=zeros(length(c));
    for q=qpts
        [N,dNdp]=shape(q(1));
        J=xe*dNdp;
        B=dNdp/J;
        xq=xe*N;
        Ke=Ke+B*k*A(xq)*B'*J*q(2);
    end
    sctr=c;
    K(sctr,sctr)=K(sctr,sctr)+Ke;
```

```
end

%% Heat Flux Matrix calculations

T1=ones(nodes_total,1)*T0;
T_0=ones(nodes_total,1)*T0;
Rho=rho_fun(T_0);
dT=100;
count=0;
dTT=[];
while max(dT) >= 0.00001
    fe=zeros(nodes_total,1);
for c=conn
    xe=x(:,c);
    RHO=Rho(c);
        for i=1:no_qpts
    [N,DNDp]=shape(qpts(1,i));
    J=xe*DNDp;
    w=qpts(2,i);
    xp=xe*N;
    rho=RHO'*N;
    F=s(xp,rho)*N*J*w;
    fe(c)=fe(c)+F;
        end
end
%% Temperature Calculation at nodes
K(nodes_total, :)= 0;
K(nodes_total,nodes_total)=1;
fe(nodes_total,1)=T0;
T=K\fe;
Rho=rho_fun(T);
dT=abs(T-T1);
T1=T;
%% convergence of Temperature for Electrical resistivity change
dTT(:,count+1)=(dT./T);
count=count+1;
end

%% Plots
figure('name',"Temprature Profile");
plot(x,T,'o--')
xlim([0 L])
title('Temperature Profile')
xlabel('X Distance (m)')
ylabel('Temeperature(^0C)')
legend('FEM')
hold off
figure('name',"Convergence plot Temperature each Node");
plot(1:count,dTT,'o--')
title('Convergence plot Temperature each Node')
xlabel('Number of Iteration')
ylabel('Error')
legend('Node 1','Node 2','Node 3','Node 4')
end

%% Shape functions for different elements
function [N,dNdp] = shape2(p)
```

```
N=0.5*[1-p(1);1+p(1)];
dNdp=[-0.5;0.5];
end
function [N,dNdp] = shape3(p)
N=[p(1)*(1-p(1))*-0.5;1-p(1)^2;0.5*p(1)*(p(1)+1)];
dNdp = [(p(1)-0.5);(-2*p(1));(p(1)+0.5)];
end
function [N,dNdp] = shape4(p)
N=[(p(1)^2-1/9)*(p(1)-1)*(-9/16);(p(1)^2-1)*(p(1)-1/3)*(27/16);(p(1)^2-
1)*(p(1)+1/3)*(-27/16);(p(1)^2-1/9)*(p(1)+1)*(9/16)];
dNdp=[(-9/16)*(3*p(1)^2-2*p(1)-1/9);(27/16)*(3*p(1)^2-(2/3)*p(1)-1);(-
27/16)*(3*p(1)^2+(2/3)*p(1)-1);(9/16)*(3*p(1)^2+2*p(1)-1/9)];
end
%% Gaussian Quadrature
function [qpts] = quadrature(n)
%   QUADRATURE
%     quadrature(n) returns a quadrature table for a rule with n
%     integration points.  The first row of the table gives the quadrature
%     point location, and the second gives the quadrature weights.

    u = 1:n-1;
    u = u./sqrt(4*u.^2 - 1);

    A = zeros(n);
    A(2:n+1:n*(n-1)) = u;
    A(n+1:n+1:n^2-1) = u;

    [v, x] = eig(A);
    [x, k] = sort(diag(x));
    qpts = [x'; 2*v(1,k).^2];
end
```

Convergence Plot

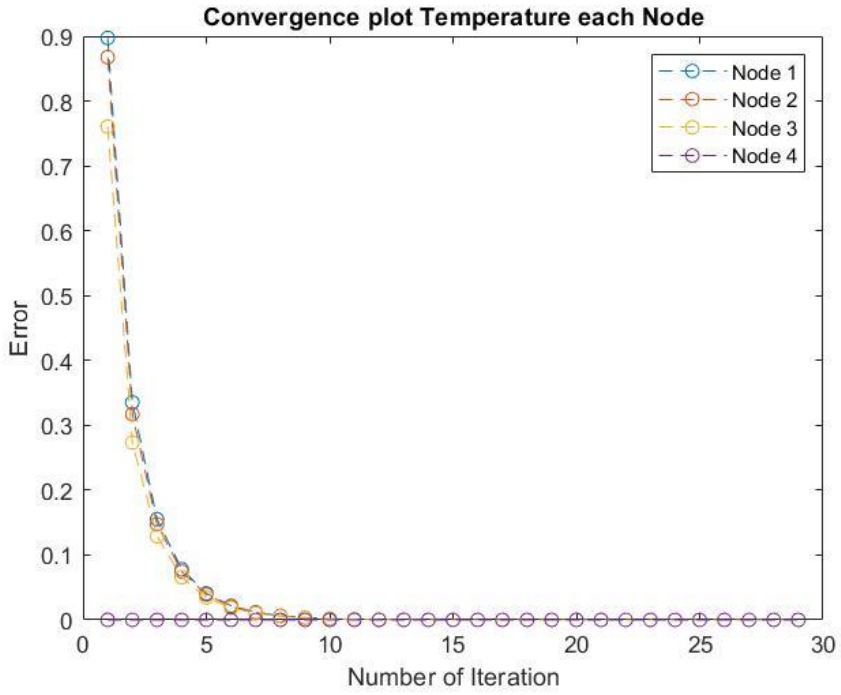


Figure 29: Convergence Plot Temperature Error vs. Number of Iteration

Temperature Plot Variable Electrical Resistivity

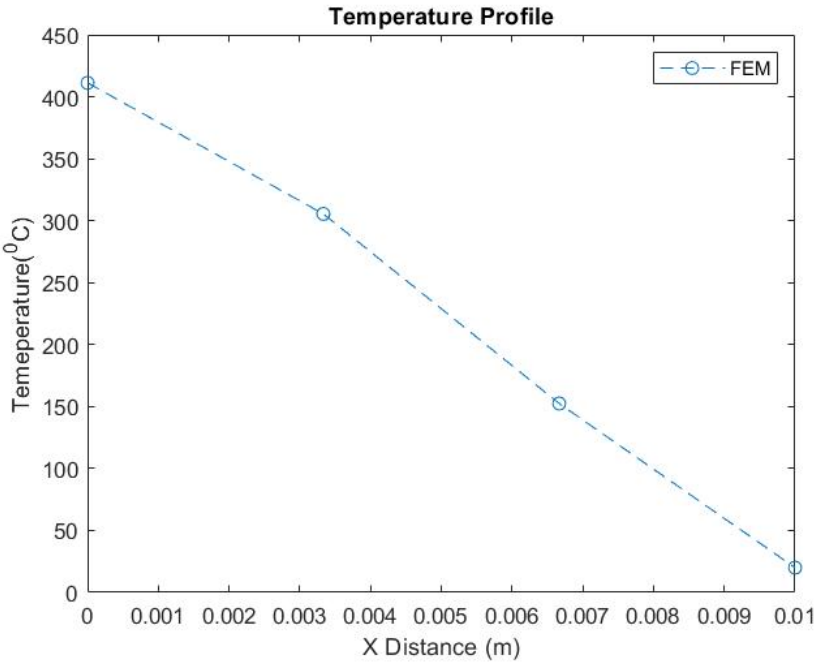


Figure 30: Temperature Field When Electrical Resistivity is Variable