MAE 503

FINITE ELEMENTS IN ENGINEERING

Project 2

Rohit Iyer (riyer9)

Harsh Verma (hpverma)

Ujjawal Jha(ujha2)

Maulik Kamdar(mkamdar1)

Contents

Problem Summary:

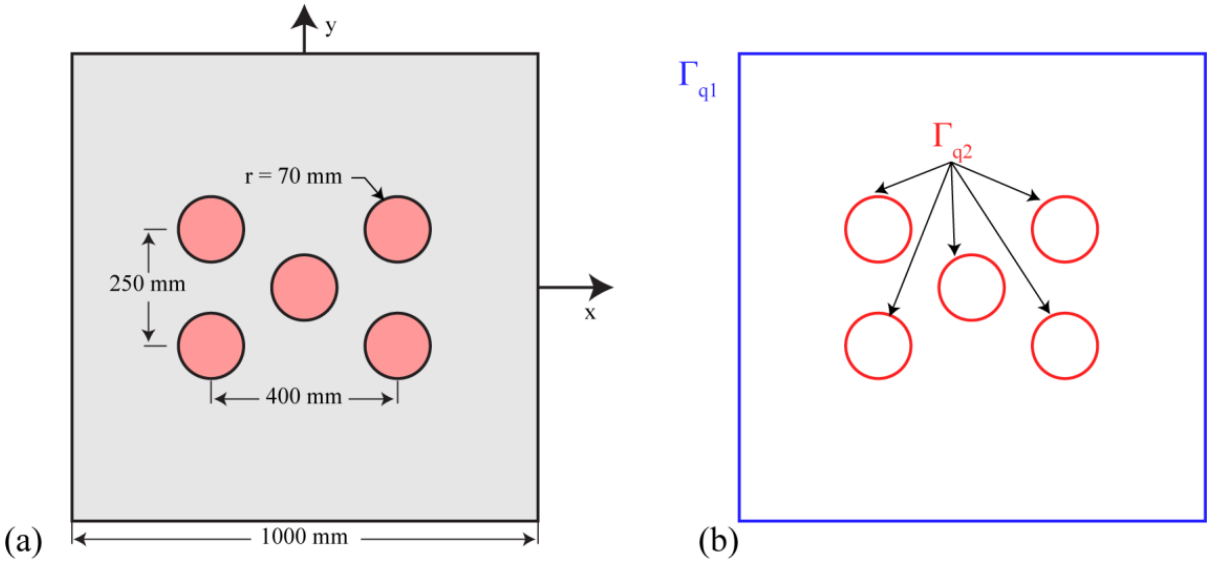


Fig 1: Given 2D geometry

The figure above shows a cross sectional view of five nuclear rods in a ceramic casing. The decay of the rods causes heat flux into the ceramic casing which is then convected out of the casing from the outer wall. Considering the required given data, we are calculating the temperature of ceramic housing caused due to the decay of rods and the heat flux fields associated with it using Abaqus and then verifying the results with the help of MATLAB. During the process, we have used processes like Richardson method, manufactured solutions and at last calculated rate of convergence on various element sizes and nodes on MATLAB. To simulate the physical interactions on the ceramic causing, we have considered a finite element analysis (Abaqus) which uses a geometrical mesh made up of nodes and elements. The ultimate goal of this project was to capture the temperature and heat flux distribution in the given geometry using Abaqus and finite elements methods which was later checked with manufactured solutions to ensure convergence and reasonable error norm values.

Strong Form equations:

- Strong form equations :-

for a given boundary Γ_i and domain Ω_i

for heat

$$\oint_{\Gamma_i} \bar{q} \cdot \bar{n} \, d\Gamma \quad - (1)$$

for the source term for heat generation (S) :-

$$\int_{\Omega_i} S \, d\Omega \quad - (2)$$

Thus the rate of change of energy is :-

$$\dot{Q} = \int_{\Omega_i} S \, d\Omega - \int_{\Gamma_i} \bar{q} \cdot \bar{n} \, d\Gamma \quad \text{- from (1) \& (2)}$$

q

By the divergence theorem

$$\dot{Q} = \int_{\Omega_i} S \, d\Omega - \int_{\Omega_i} \nabla \cdot \bar{q} \, d\Omega$$

$$= \int_{\Omega_i} S - \nabla \cdot \bar{q} \, d\Omega$$

Since our problem is steady state $\dot{Q} = 0$

$$\therefore \int_{\Omega_i} S - \nabla \cdot \bar{q} \, d\Omega = 0$$

This gives

$$S - \nabla \cdot \bar{q} = 0$$

$$\therefore \boxed{\nabla \cdot \bar{q} - S = 0} \quad -(*) \text{ (Energy balancing)}$$

Now for heat flux we use fourriers Law.

$$\boxed{\bar{q} = -K \nabla T}$$

(K: conductivity)

Substituting this in (*) we get.

$$\nabla \cdot (-K \nabla T) - S = 0$$

$$\therefore \boxed{\nabla (K \nabla T) + S = 0} \quad -(\text{combined})$$

General equation; since our conductivity is const.

$$\boxed{K \nabla^2 T + S = 0}$$

These two equations are the strong form for heat conduction or 2D heat transfer.

Also, we apply the boundary conditions.

For Natural Boundary condition we have:

$$\boxed{q_n = \bar{q} \cdot \bar{n} = \bar{q}(x, y)} \quad -(\text{for } \Gamma_q)$$

For Essential B.C.,

$$\boxed{T = \bar{T}(x, y)} \quad -(\text{on } \Gamma_T)$$

System of Units:

Length	m
Power	Watts
Temperature	Kelvin
Radius of rods	m
Heat Flux	20,000 W/m^2
Thermal Conductivity	17 W/m-K
Convective Heat Transfer Coefficient	10 W/m^2-K

Table 1: System of units

Weak form of the problem:

— Weak form :-

- for the weak form we multiply strong form with a test function(w) and integrate over the domain.

$$\therefore \int_{\Omega} w\left[\bar{\nabla}\cdot(-k\bar{\nabla}T) - s\right] d\Omega = 0$$

Using integration by parts;

$$-\int_{\Omega} \bar{\nabla}\cdot(wk\bar{\nabla}T)_{d\Omega} - \int_{\Omega} \bar{\nabla}_{,}w\cdot k\bar{\nabla}T\, d\Omega$$

$$+\int_{\Omega} ws\, d\Omega = 0.$$

Rearranging

$$\therefore \int_{\Omega} \nabla w\cdot k\nabla T\, d\Omega = -\int_{\Omega} \nabla\cdot(wk\nabla T)\, d\Omega$$

$$+\int_{\Omega} ws\, d\Omega$$

Now solving 1st term in RHS. with divergence theorem;

$$\therefore \int_{\Omega} \nabla w\cdot k\nabla T\, d\Omega = \oint_{\Gamma} wk\nabla T\, d\Gamma$$

$$+\int_{\Omega} ws\, d\Omega.$$

For temperature boundary (Γ_T) we have $w = 0$

This gives for fourier's law

$$-\bar{q} = K \nabla T \cdot \bar{n} \quad - \text{ for } \Gamma_q$$

$$\therefore \int_\Omega \nabla w \cdot K \nabla T \, d\Omega = -\int_{\Gamma_q} w \bar{q} \, d\Gamma + \int_\Omega w s \, d\Omega$$

This is the weak form equation.

Applying BC.

$$T = \bar{T} \quad \text{on } \Gamma_T$$

Now for discretization of test & trial function:-

$$T(x,y) \approx T^h(x,y) = d_I^e N_I^e$$

(h ← element size; d_I^e ← nodal values of temp.; N_I^e ← shape fn.)

$$w(x,y) \approx w^h(x,y) = w_I^e N_I^e$$

$$\therefore \underset{\sim}{d}^e = \underset{\sim}{L}^e \underset{\sim}{d} \leftarrow \text{global matrix.}$$

($\underset{\sim}{L}^e$ ← gather matrix)

Also $\underset{\sim}{w}^e = \underset{\sim}{L}^e \underset{\sim}{w}$

The gradients are as follows.

$$\nabla T \approx \nabla T^h = d_I^e \nabla N_I^e$$

$$\nabla w \approx \nabla w^h = w_I^e \nabla N_I^e$$

Now we substitute these into the weak form.

$$\therefore \underset{\sim}{w}^T \sum_e \left[(\underset{\sim}{L}^e)^T \int_\Omega (\underset{\sim}{B}^e)^T \underset{\sim}{D}^e \underset{\sim}{B}^e \, d\Omega \, (\underset{\sim}{L}^e) \right] \underset{\sim}{d}$$

This K for our case (diffusivity tensor)

$$= -w^T \sum_e (\underset{\sim}{L}^e)^T \int_{\Gamma_q} (\underset{\sim}{N}^e)^T \bar{q} \, d\Gamma + w^T \sum_e (\underset{\sim}{L}^e)^T \int_\Omega (N^e)^T S \, d\Omega$$

$$\therefore \underset{\sim}{w}^T \sum_e \left[(\underset{\sim}{L}^e)^T \int_\Omega (\underset{\sim}{B}^e)^T \underset{\sim}{D}^e \underset{\sim}{B}^e \, d\Omega \, (\underset{\sim}{L}^e) \right] \underset{\sim}{d} + w^T \sum_e (\underset{\sim}{L}^e)^T \int_{\Gamma_q} (\underset{\sim}{N}^e)^T \bar{q} \, d\Gamma - w^T \sum_e (\underset{\sim}{L}^e)^T \int_\Omega (N^e)^T S \, d\Omega$$

K^e $\qquad$ f^e

the elemental K and f values are

$$K^e = \int_{\Omega^e} (\underset{\sim}{B}^e)^T \underset{\sim}{D}^e \underset{\sim}{B}^e \, d\Omega$$

$$f^e = -\int_{\Gamma_q} (\underset{\sim}{N}^e)^T \bar{q} \, d\Gamma + \int_{\Omega^e} (N^e)^T S \, d\Omega$$

∴ Global K and f are

$$K = \sum_e (\underset{\sim}{L}^e)^T \underset{\sim}{k}^e \underset{\sim}{L}^e$$

$$f = \sum_e (\underset{\sim}{L}^e)^T f^e$$

∴ Taking w^T common;

$$w^T [Kd - f] = 0$$

$$\therefore Kd = f$$

$$\boxed{d = f \backslash K}$$

ABAQUS:

- We start with a shell type 2D planar geometry simplified using symmetry.
- After creating the geometry, we add a material property to the casing with the given thermal conductivity value of 0.017W/mm-K
- Then we create the assembly as an independent part
- The next option is in the step feature, we add a step after the already existing initial step.
- In the initial step we added a pre-defined field of prescribed temperature at the outer wall at 0 °C.
- In the Step 1 part, the convection interaction is applied on the outer walls of the casing of 10^{-5} W/mm^{2}-K, also we apply the heat source load of 0.02 W/mm^{2} at the boundary of the nuclear rods.
- Now for the mesh size we are using Tri3 element type and varying the element size in the geometry to get closer to the analytical solution.
- The element size is kept finer at the fuel rod boundary than the casing for a better result.
- The Tri3 element type helped in getting more nodes for the solution being as finer of a mesh as possible as compared to quad element type.
- After tuning the different sizes on the Tri3 element the best result was found at global size of 25 and on the fuel rod boundary a size of 9. This gave better heat flux results than the prior one.
- For a better perspective plots of QUAD and Tri3 element type meshes have been attached below.
- For assumptions, we have assumed that since geometry is symmetric about X and Y and that the material is perfectly isotropic in nature thus enabling us to simplify it using symmetry to save computational power.
- The convective and heat flux boundary conditions are constant in nature and not subject to change with time or position.

Comparison between Abaqus results & MATLAB results are given below:

We start by comparing our results of tr3 mesh of 20 mm size between MATLAB and Abaqus

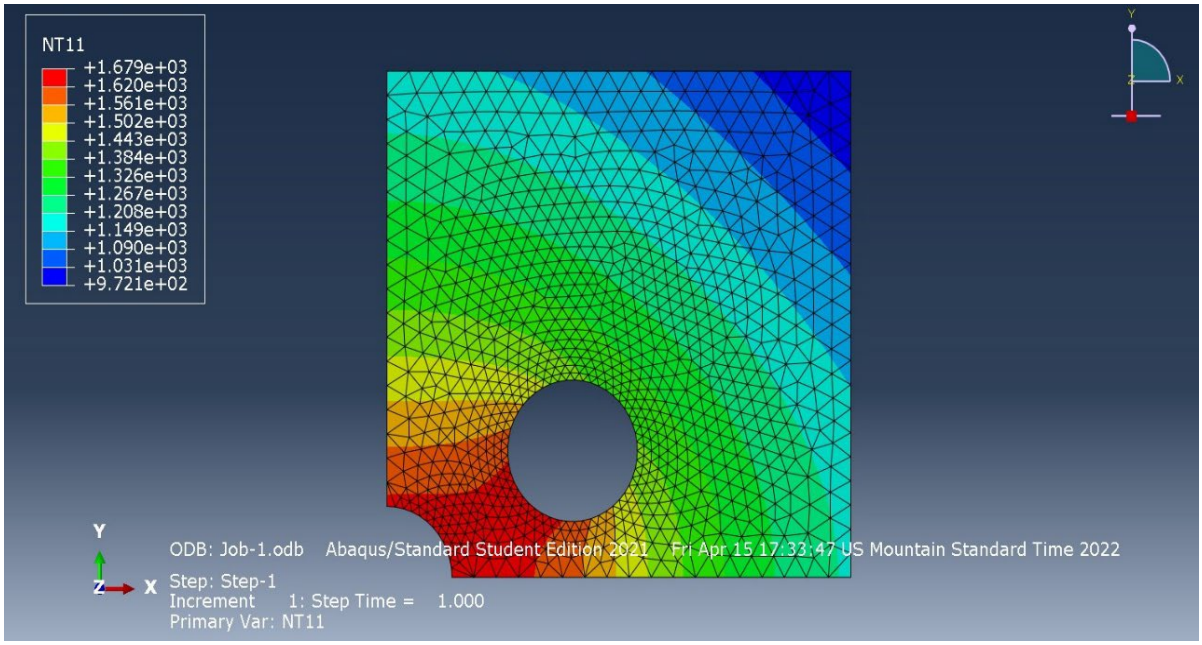


Fig 2(a): Temperature for Tri3 element type (Abaqus)

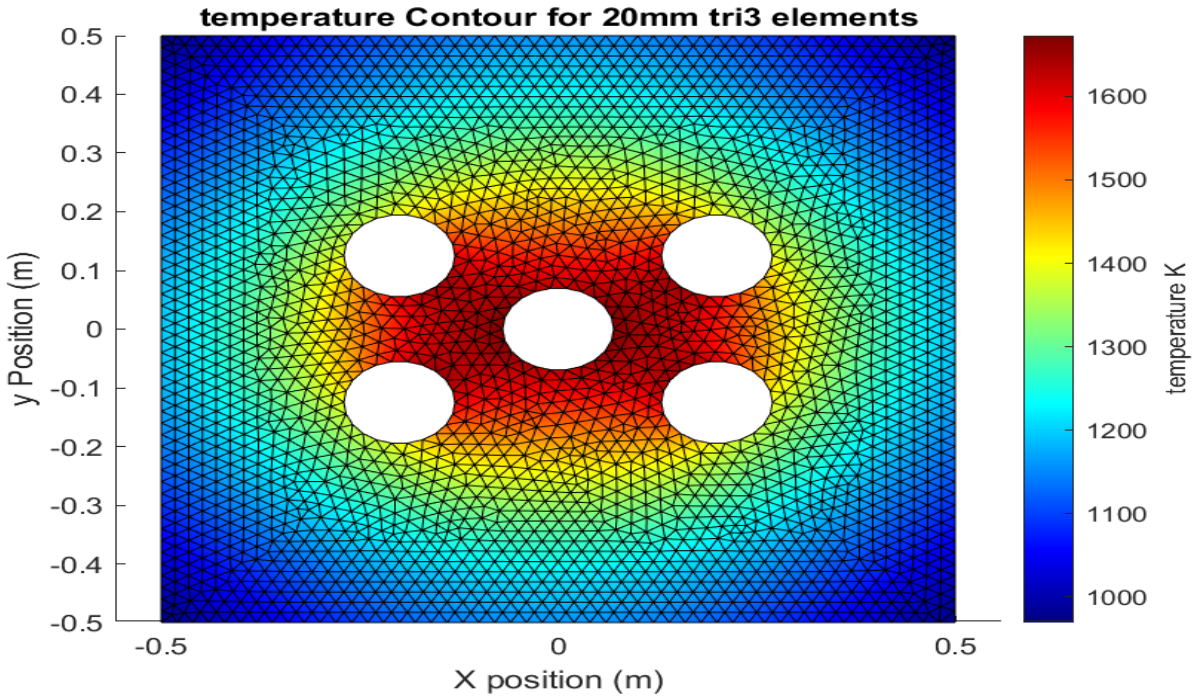


Fig 2(b): Temperature for Tri3 element type (MATLAB)

We notice that for this case, we notice that the temperature for MATLAB is 1672.7 K and which is very close to Abaqus's value of 1679 K, and we observe very similar patterns as well. We notice a similar thing for heat flux thus we can conclude that the results are very similar and Abaqus is a reliable estimator for our case.

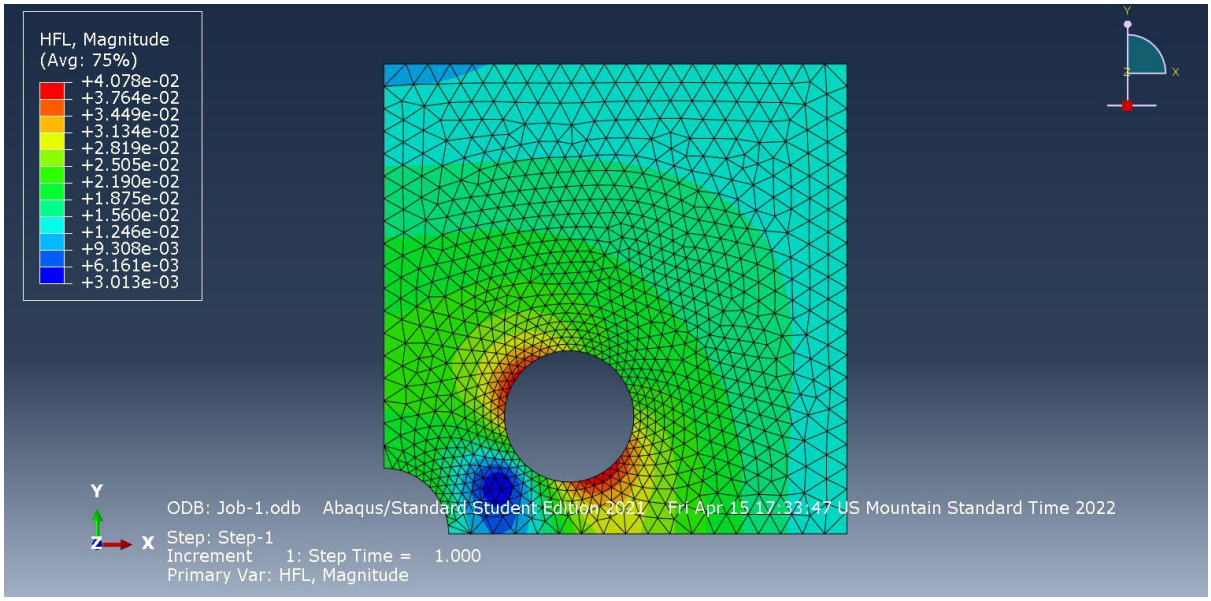


Fig 3.(a): Heat-flux for Tri3 element type(Abaqus)

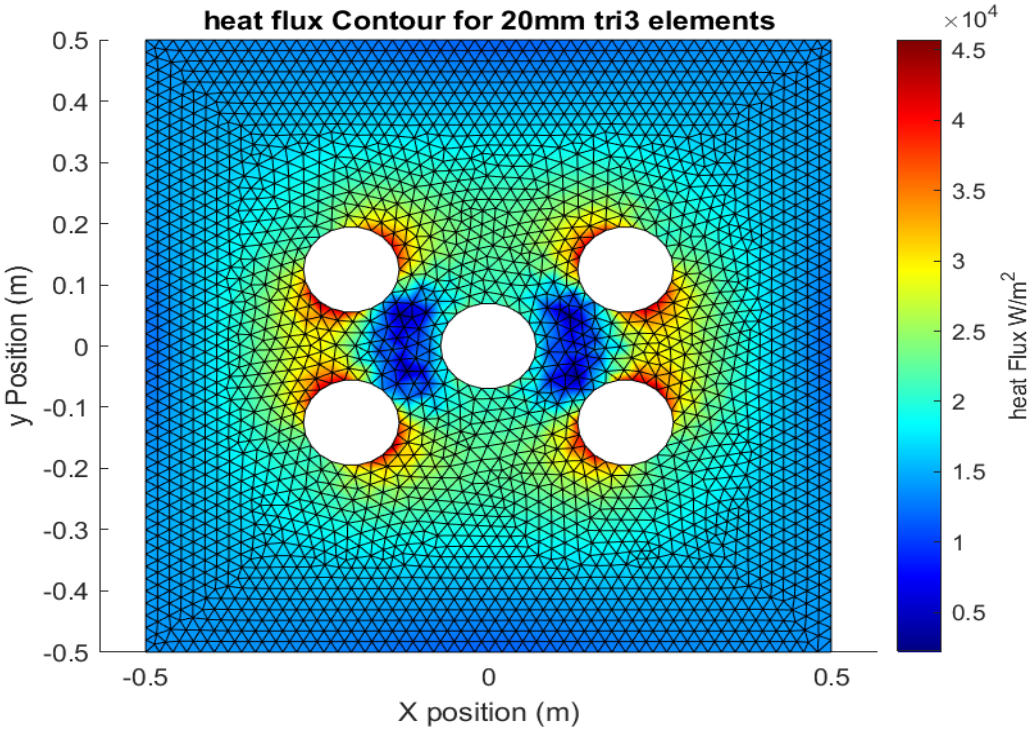


Fig 3.(b): Heat-flux for Tri3 element type(MATLAB)

We now do the same using Quad4 elements and observe similar results amongst MATLAB and Abaqus. The mesh quality is not that good at some regions near the fuel rods have some tri elements although the entire structure is Quad dominant.

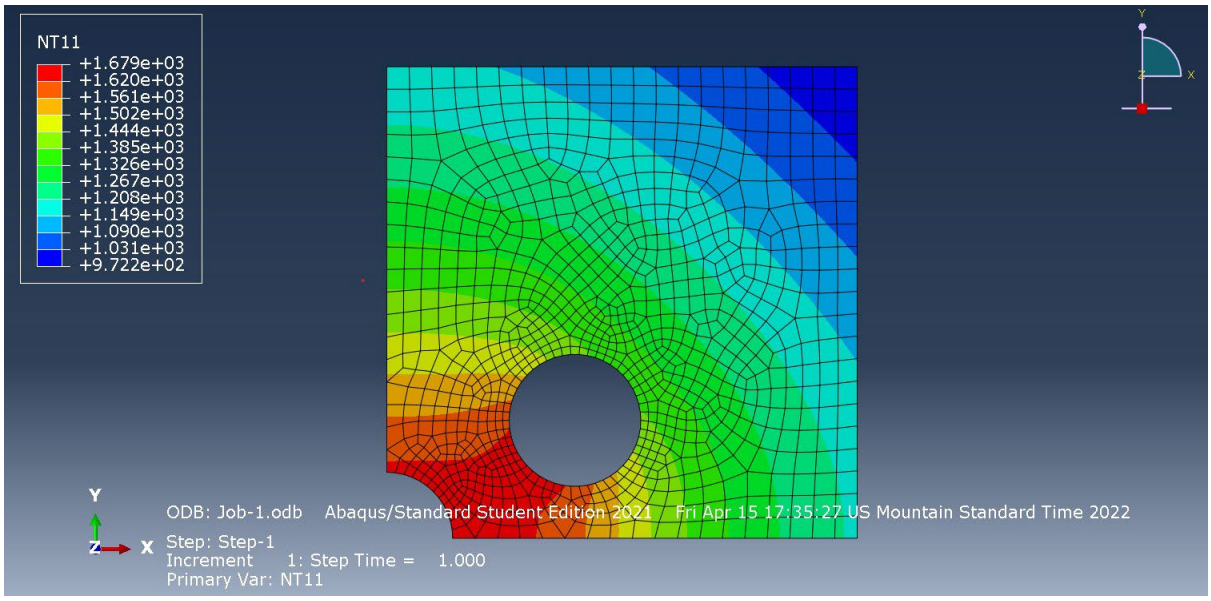


Fig 4(a): Temperature for QUAD element type (Abaqus)

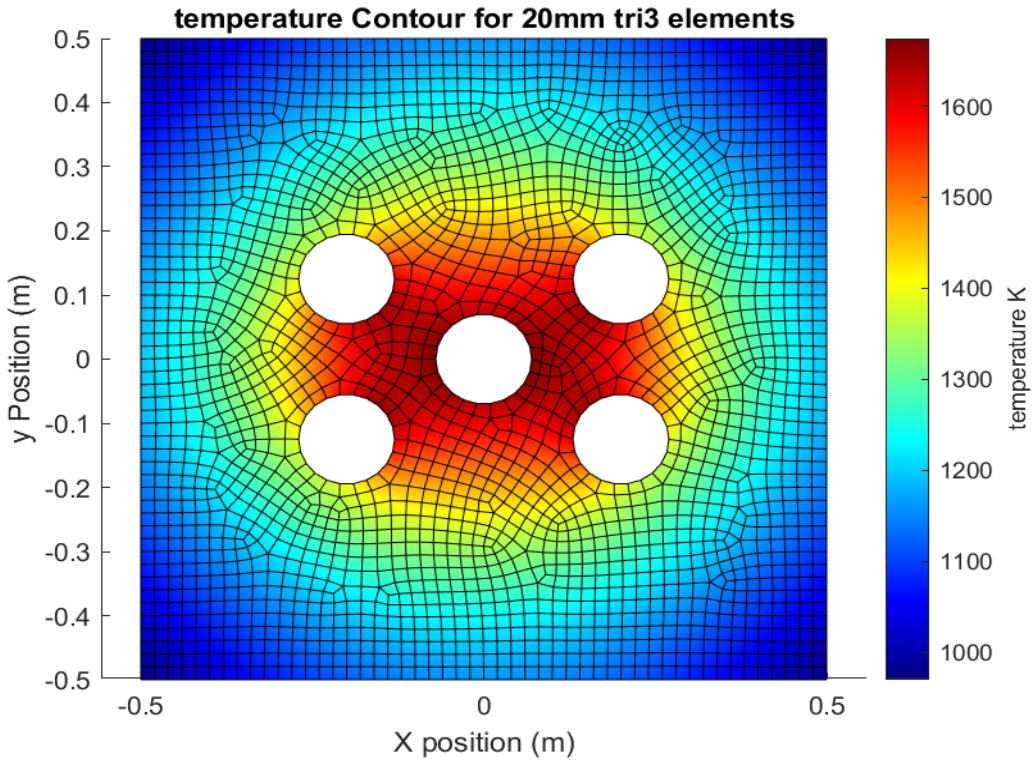


Fig 4(b): Temperature for QUAD element type(MATLAB)

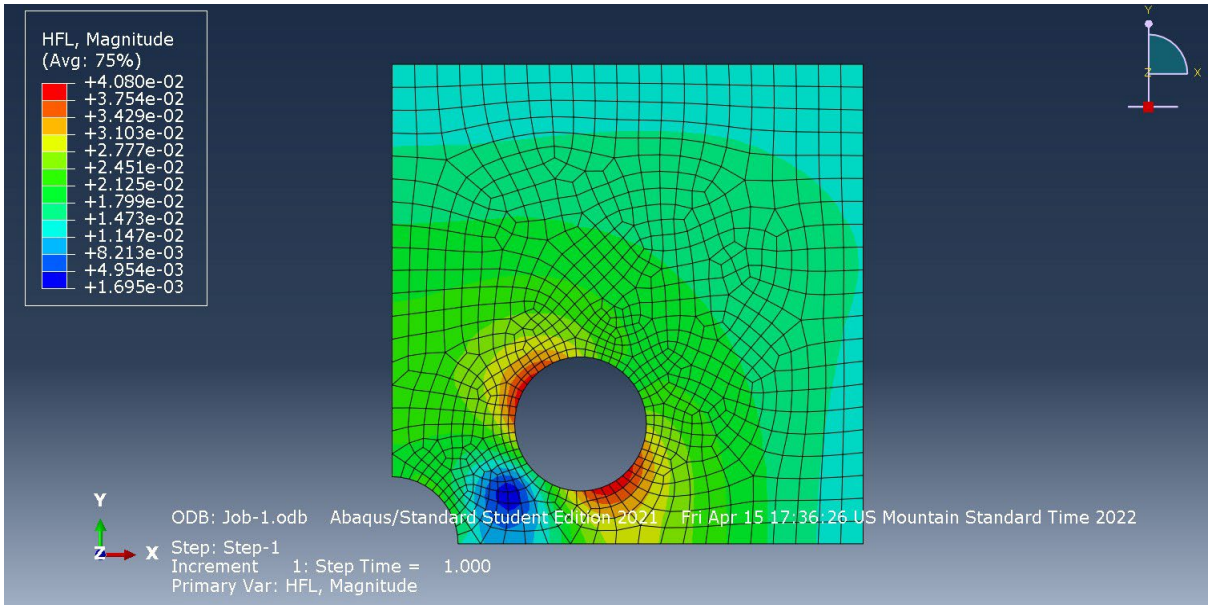


Fig 5(a): Heat-flux for QUAD element type(Abaqus)

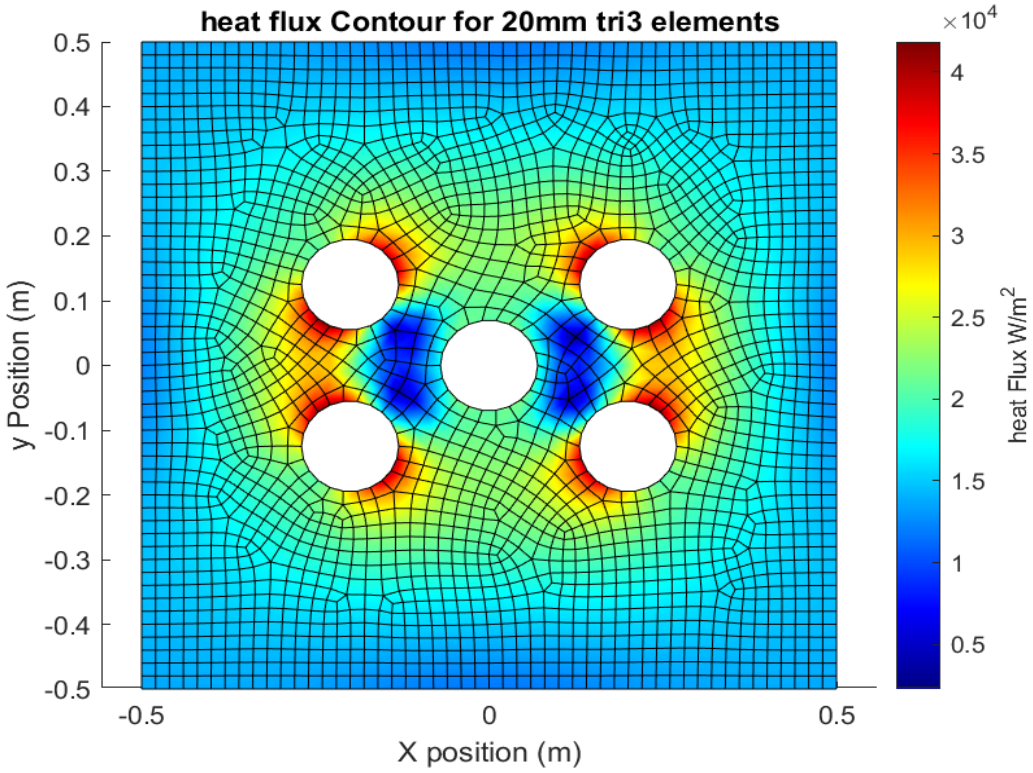


Fig 5(b): Heat-flux for QUAD element type (MATLAB)

The Abaqus solution is more accurate for larger mesh sizes as we are using mesh refinement tool to reduce the size of the mesh automatically in areas of interests thus providing a more accurate depiction of both, max value of temperature and heat flux but also their distribution.

The Abaqus value can converge to an optimized solution, but I don't believe it would converge to the same solution as the one we get from MATLAB unless and until the mesh we generated in Abaqus is the same as the one we use in MATLAB since the background equations for all FEA solvers for such simple cases generally follows **"d = K\F".**

SPlots for temperature of Tri3 elements for different mesh sizes are follows:

We perform manual convergence analysis for all, Tri3 and Tri6 elements reducing the mesh size from 80 mm to 2 mm to check for convergence which was observed to arrive faster for Tri6 than Tri3 elements as shown in the convergence plots.

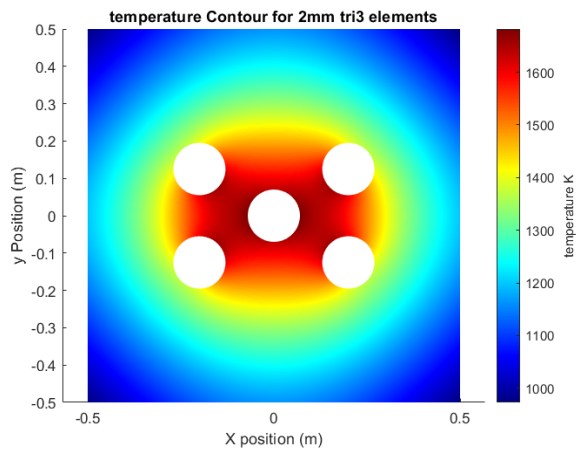


Fig. 6 Temperature contour for 2mm tri3 element

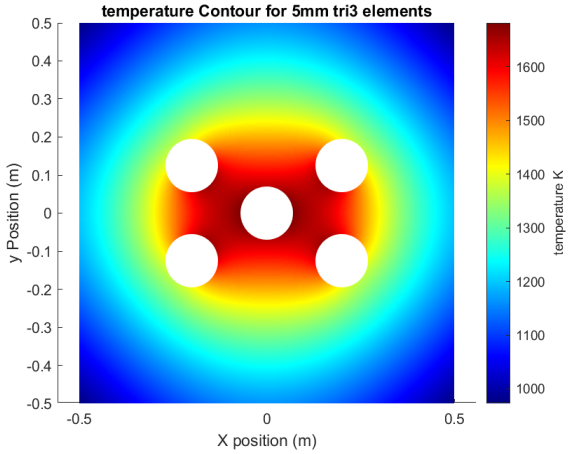


Fig. 7 Temperature contour for 5mm tri3 element

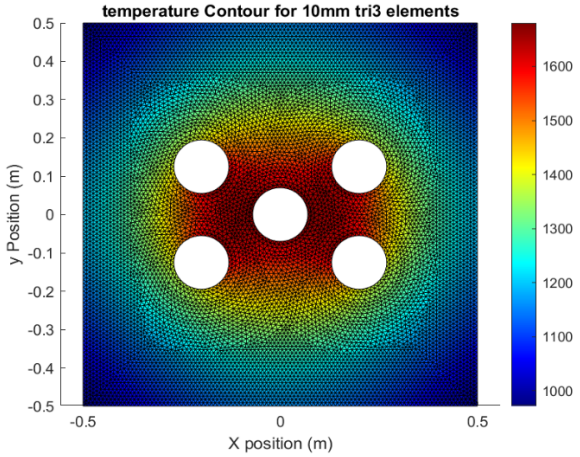


Fig. 8 Temperature contour for 10mm tri3 element

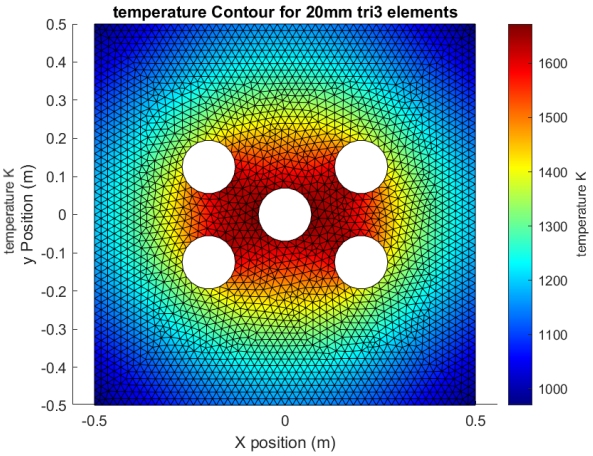


Fig. 9 Temperature contour for 20mm tri3 element

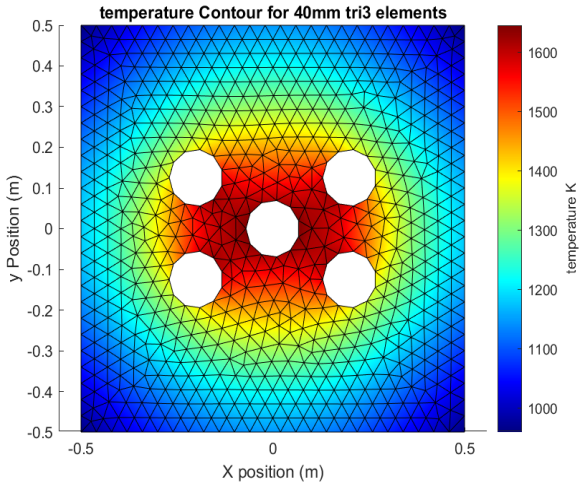


Fig. 10 Temperature contour for 2mm tri3 element

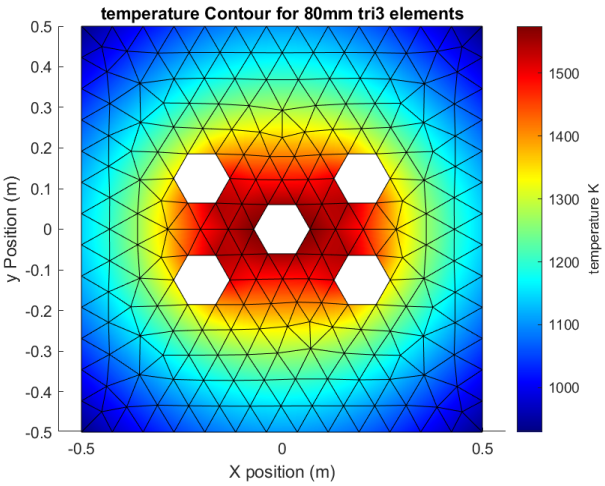


Fig. 11 Temperature contour for 2mm tri3 element

Plots for heat-flux of Tri3 elements for different mesh sizes are follows:

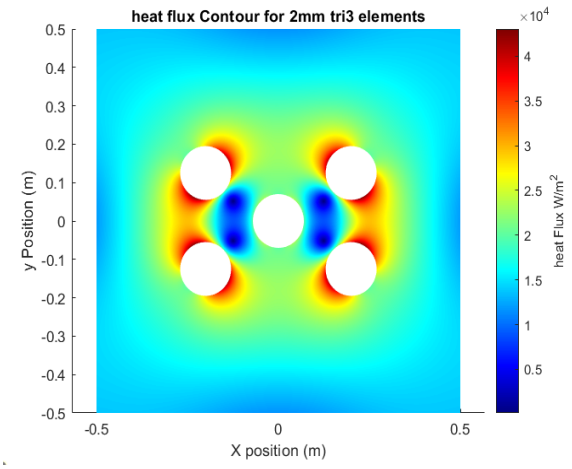


Fig. 12 Heat-flux contour for 2mm tri3 element

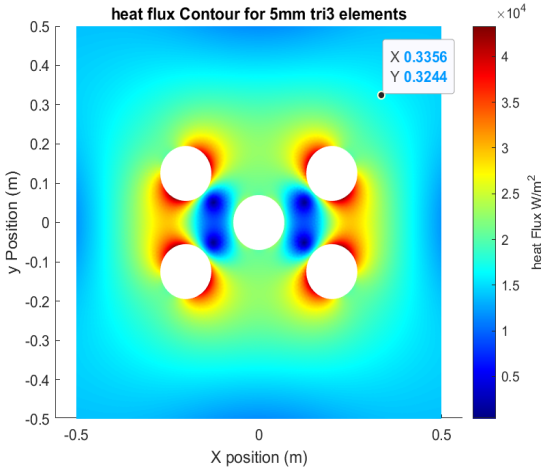


Fig. 13 Heat-flux contour for 5mm tri3 element

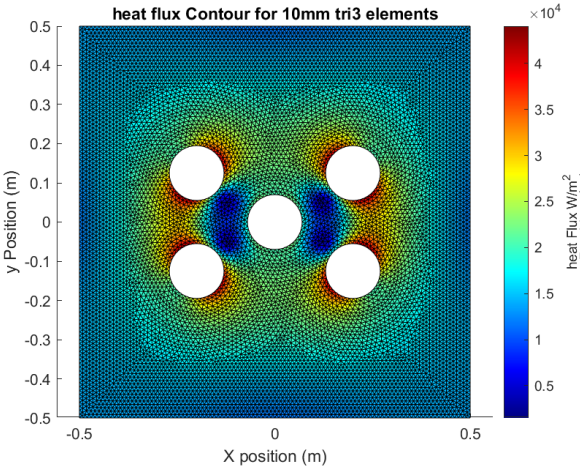


Fig. 14 Heat-flux contour for 10mm tri3 element

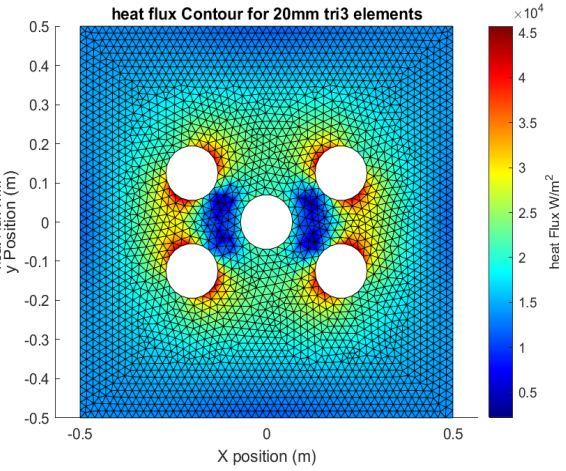


Fig. 15 Heat-flux contour for 20mm tri3 element

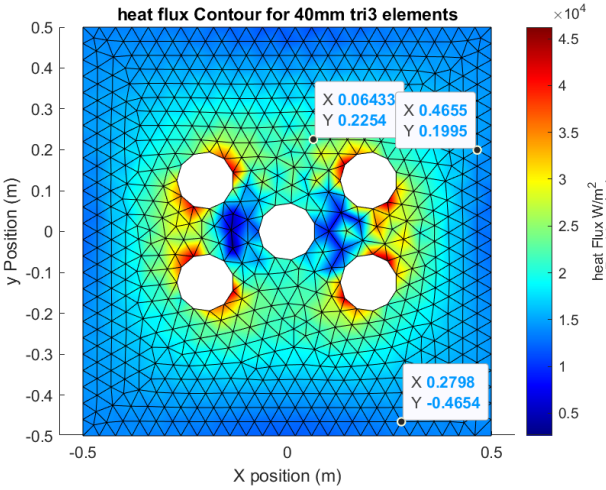


Fig. 16 Heat-flux contour for 40mm tri3 element

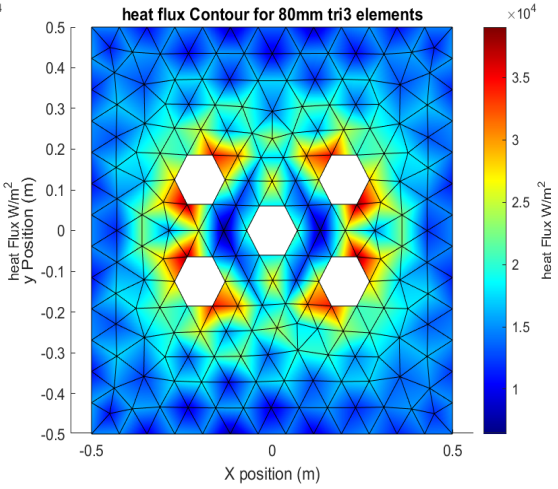


Fig. 17 Heat-flux contour for 80mm tri3 element

Max temp and heat flux table for different element sizes for each type:

Attached bellow is the table containing max temperature and max heat flux values for **Tri3 element** of different mesh sizes:

Size (mm)	Temp (Max)(K)	Heat Flux (Max)(W/mm^2)
80	1.57E+03	3.91E+04
40	1.65E+03	4.62E+04
20	1.67E+03	4.57E+04
10	1.68E+03	4.40E+04
5	1.68E+03	4.33E+04

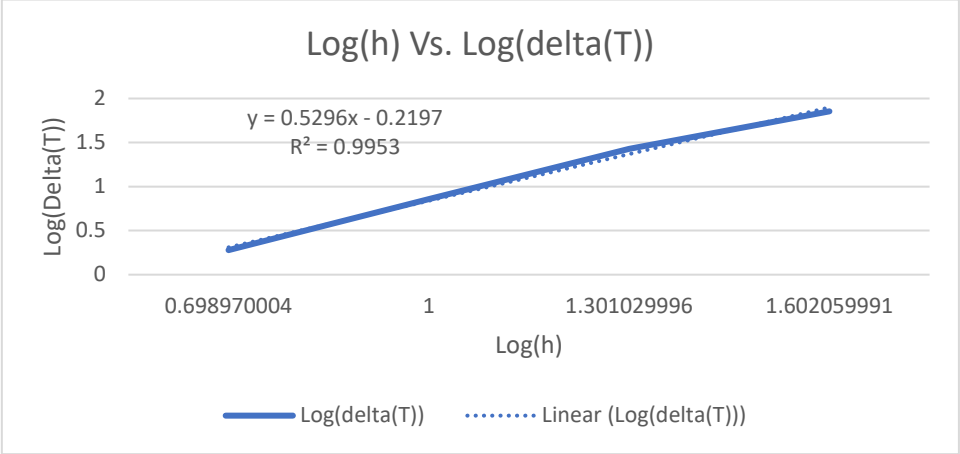
Table 1: Max Temperature and Heat Flux for Tri3 element type

We now shall use Richardson Extrapolation to estimate the error from each of the mesh sizes we analyzed. For this, we first plot a graph between Log(h) and Log (Delta(T)) to fit a curve which we shall use to fit a curve to extrapolate the Delta (T) values for finer mesh sizes.

Size (h) (mm)	Temp (Max)(K)	Delta(T)	Log(h)	Log(delta(T))
5	1.68E+03	1.90E+00	0.69897	0.278754
10	1.68E+03	7.20E+00	1	0.857332
20	1.67E+03	2.67E+01	1.30103	1.426511
40	1.65E+03	7.15E+01	1.60206	1.854306
80	1.57E+03		1.90309	

Table 2: Max Temperature and Heat Flux for Tri3 element type from Richardson extrapolation

With this, we get the graph and the fitted trend line:



Graph 1. Log(h) vs log(delta(T))

The equation of the curve is: y = 0.5296*x - 0.2197. We now use this to extrapolate value of Delta(T) which change in "h" and the values in red are the estimated values.

Size (h) (mm)	Delta(T)
80	
40	6.140114319
20	4.253544409
10	2.946629183
5	2.041267871
2.5	1.41408174
1.25	0.979600569
0.625	0.678615137
0.3125	0.47010845
0.15625	0.325666114
0.078125	0.225604151
0.0390625	0.156286548
0.01953125	0.108267003
0.009765625	0.075001617
0.004882813	0.051957129
0.002441406	0.035993134

Table 3. Size(h) & Delta Value(T)

We now use the highlighted values to calculate the error or Delta(T5) which is T5-T0 where T0 is the actual temperature of the body.

Delta(T5) = T5-T0 = 4.521181591

Thus, **T0 = 1681.8 K.**

We shall now use this value to estimate error in our temperature values we analyzed.

Size (h) (mm)	**Delta (T) (K)**	**Error (K)**
40	6.140114319	3.58E+01
20	4.253544409	9.10E+00
10	2.946629183	1.90E+00
5	2.041267871	0.00E+00

Table 4. Size(h), Delta(T) & Associated Error

Thus, the error values we obtained in our analysis according to Richardson extrapolation are mentioned in the above table in "Red". This concludes our error estimation using Richardson extrapolation.

Using the method of manufactured solutions to determine the convergence rate:

Since there is no method to analytically solve this problem, we shall use method of manufactured solution to construct the solution for our problem. We do this in order to determine the convergence rate of our solution.

First, we start by choosing a solutions "Ts" which we ensure it to be sufficiently smooth in order to provide an accurate result.

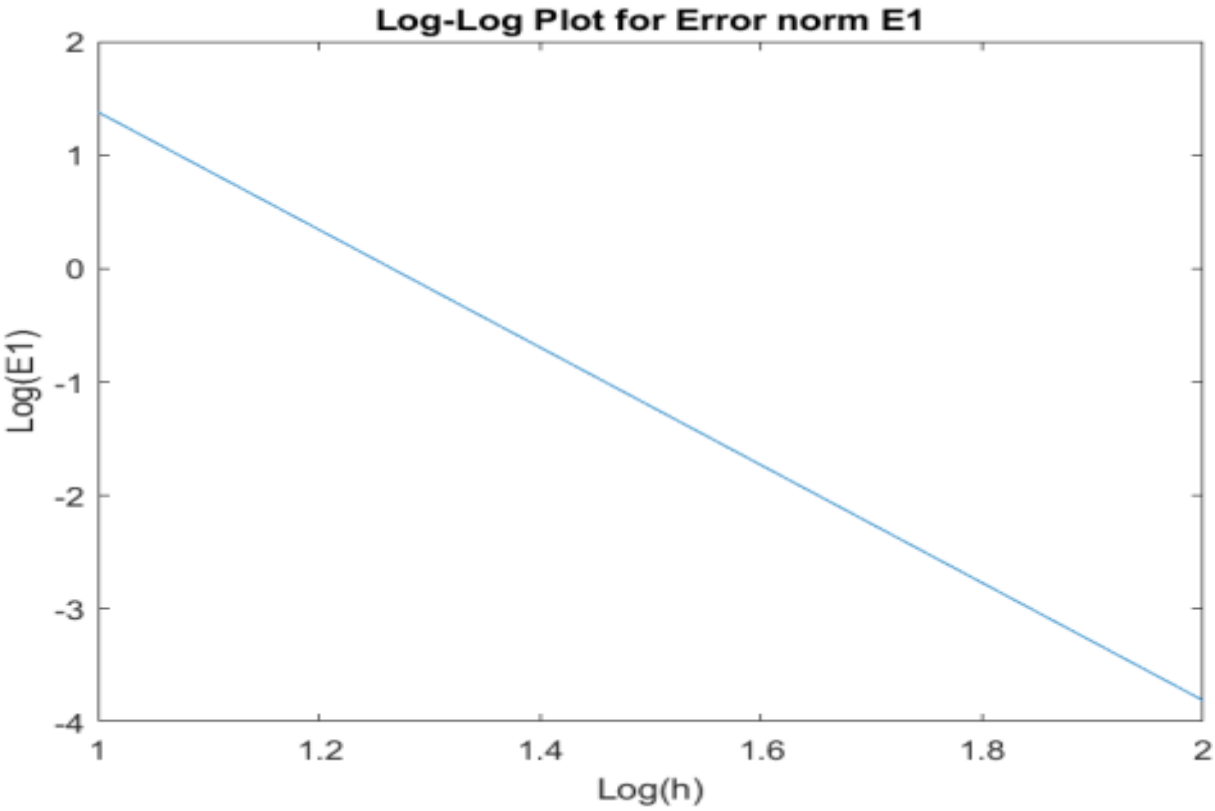
Ts = @(x,y)sin(x).*sin(y); (where Ts is the temperature star)

Using our strong and weak form of equation, we derive the function for qs and S.

qs= @(x,y)[-17*cos(x)*sin(y),-17*cos(y)*sin(x)]; (where qs is the flux)

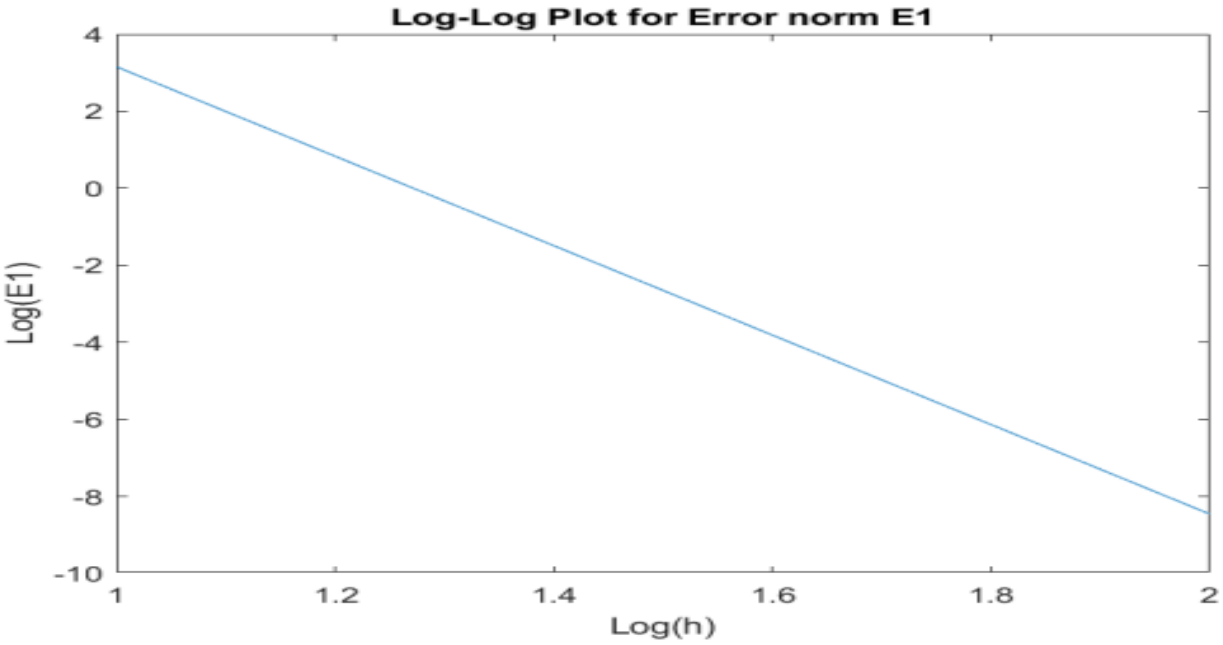
s = @(x,y)sin(x).*sin(y).*3.4e+1 (here, S is the body heat function)

After manufacturing our solution, we shall then calculate the E1 error norm for our tri3 element type for the FEM result. We obtain the following plot for the log(h) vs log(E1) graph which results in giving us the rate of convergence of 1.3792.



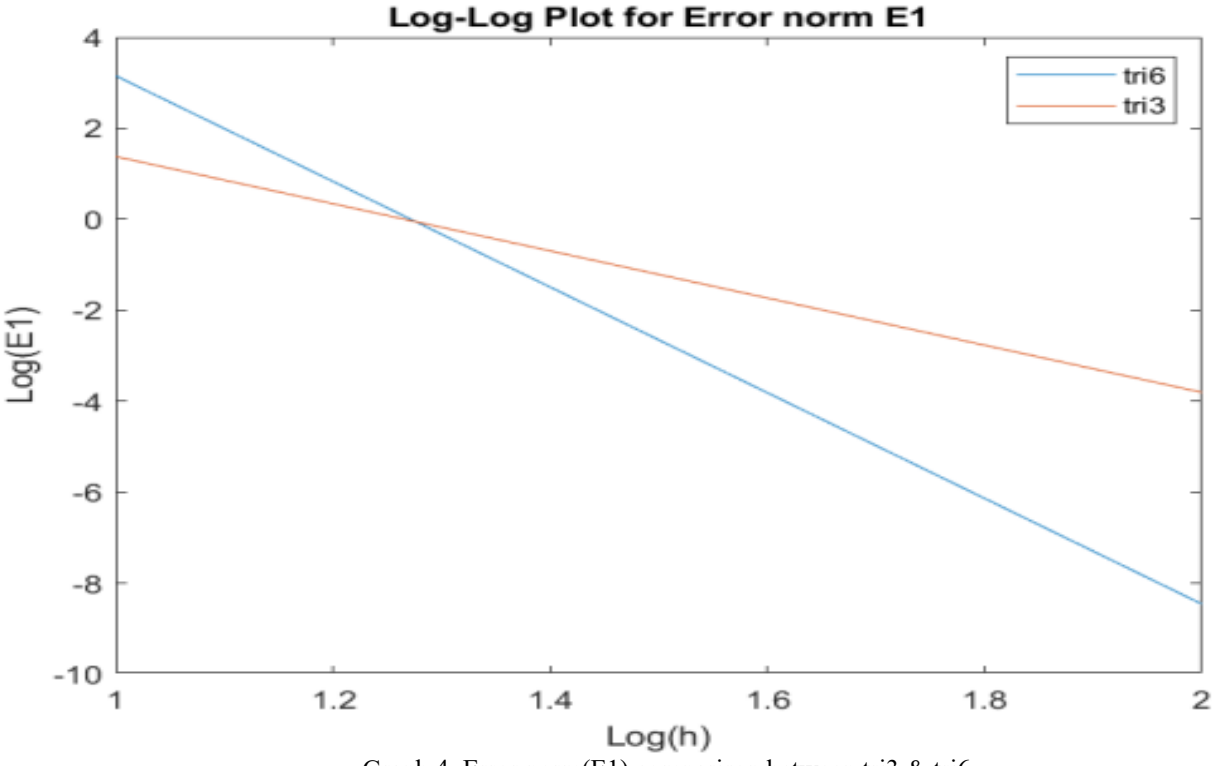
Graph 2. Error norm(E1) plot tri3 element

We also did this for our tri6 element, and it gives the following graph with rate of convergence of 3.1544:



Graph 3. Error norm(E1) plot tri6 element

On comparing these two, it is evident that element tri6 results in a faster convergence rate as compared to tri3 which we observable from the graph below,

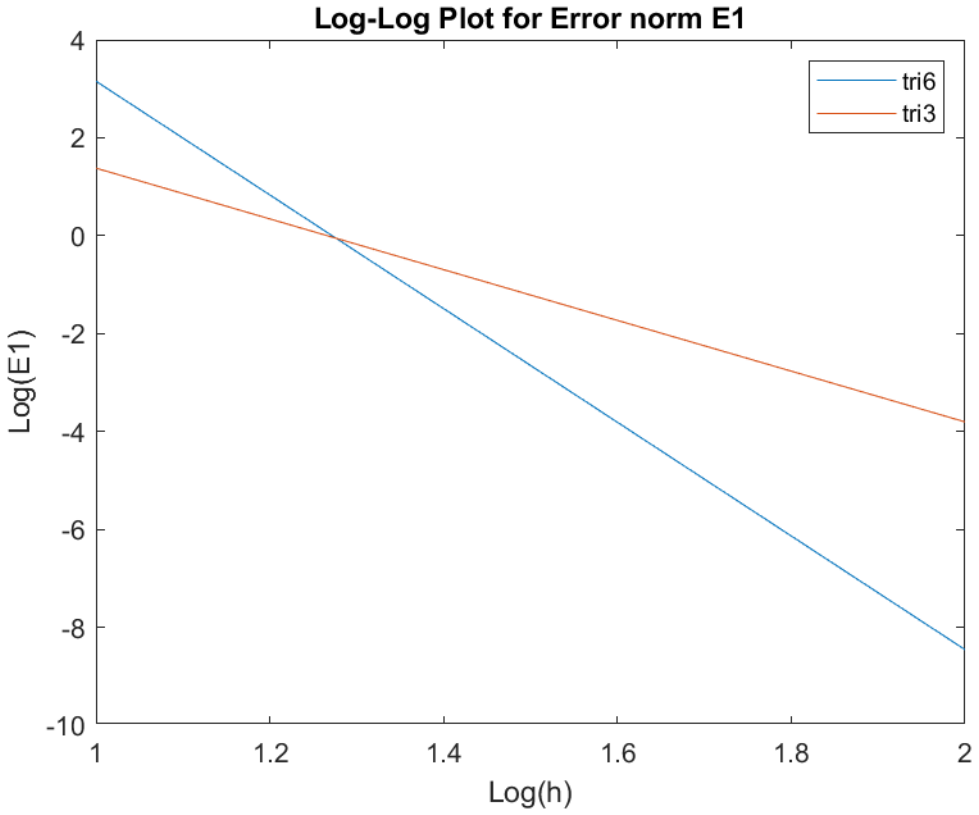


Graph 4. Error norm(E1) comparison between tri3 & tri6

BONUS:

We have also calculated the results for Tri6 element type for mesh sizes 80mm to 5mm. We notice that as a rule of thumb, the fewer elements you have the less accurate your result will be. Just using this logic, it would be obvious the TRI3 is the worst 2D element, however the TRI3 element has additional issues. the TRI3 element has issues with stiffness. TRI3 allows the FEA solver to generate a linear plane to interpolate deformation from using the 3 provided nodal points. This causes an issue with the accuracy of the FEA solution because, as mentioned previously in "Understanding Strain and FEA", a linear plane of deformation generates a constant strain and therefore stress over an entire element. Stress is always constantly changing as you move along a structure making the constant strain and stress over an element very inaccurate. For this reason, TRI3 elements tend to form results with too much stiffness and therefore undervalue stress leading results. TRI3 Elements do have one advantage, if the analysis is simple enough and you use enough TRI3 elements they can generate results very quickly.

TRI6 is a second-order or quadratic version of TRI3. Being a second order provides one big advantage, FEA is no longer limited to using a linear plane of deformation. Using a polynomial plot for deformation allows the software to apply a linear plot of stress and strain improving accuracy over TRI3 dramatically. However, TRI6 still have stiffness issues as with any triangle element making them less accurate than other 2D elements. The downside of the TRI6 compared to the TRI3 is with 2x the nodes it will demand substantially more computational power and time. **The value of temperature and heat flux were mentioned in the comparative study earlier** and we have attached the following graph for comparative study of convergence analysis of tri3 and tri6 elements which agrees with our statement above.



Graph 4. Error norm(E1) comparison between tri3 & tri6

Appendix:

```
clc;
clear all;

kappa=17;%W/mk
h=10;%W/m2-K
qbar=20000;%W/m
%nne=3;
T_ambient=0;

    mesh=abaqus_reader("C:\Users\ujjaw\Dropbox (ASU)\PC\Desktop\HOME\ASU\ASU subjects\FEM\project 2\T3-meshes\t3-mesh-h20.inp");
    T = zeros(length(mesh.x),1);
    F = zeros(length(mesh.x),1);


  %  D = ones(2)*kappa;
    K = spalloc(length(mesh.x),length(mesh.x),9*length(mesh.x));
    quad=[1/3;1/3;1/2];

    for c = mesh.conn
        Ke = zeros(length(c));
        xe = mesh.x(:,c);
        fe=zeros(length(c),1);
        for q = quad

            % if nne==4
            %  [N, dNdp] = shape_quad4(q);
           %  elseif nne==3
              [N, dNdp] = shape_tri3(q);
          %   elseif nne==6
            %  [N, dNdp] = shape_tri6(q);
          % end

            J = xe*dNdp;
            B = dNdp/J;
            Ke = Ke+ B*kappa*B'*det(J)*q(end);
            fe=fe+N*kappa*det(J)*q(end);
        end

      K(c,c) = K(c,c) + Ke;
      F(c)=F(c)+fe;
   end


  %%
      perimeter_edge_conn = mesh.edge_connectivity('perimeter');
      hole_edge_conn = mesh.edge_connectivity('holes');
```

```
    for ch = hole_edge_conn
       xh = mesh.x(:,ch);
       he = zeros(length(ch),1);
       for g = [0;2]
           %[Ne, dNdpe] = shape_tri3(g);
           Ne = 0.5*[1-g(1); 1+g(1)];
          dNdpe = [-0.5; 0.5];
           %dNdpe = [-0.5*(1-g(1)); 0.5*(1-g(1))];
           J2 = xh * dNdpe;
           he = he + Ne*qbar*norm(J2)*g(end);

       end
      F(ch)=F(ch)+he;
    end


    for cp = perimeter_edge_conn
       xp = mesh.x(:,cp);
       fg = zeros(length(cp),1);
       for u = [0;2]
           %[Ne, dNdpe] = shape_tri3(g);
           Np = 0.5*[1-u(1); 1+u(1)];
          dNdpp = [-0.5; 0.5];
          % dNdpp = [-0.5*(1-u(1)); 0.5*(1-u(1))];
           J3 = xp * dNdpp;
           fg = fg + Np*h*Np'*norm(J3)*u(end);
       end
      K(cp,cp)=K(cp,cp)+fg;

    end




T=K\F;
max_temperature=max(T)

%%flux
A = spalloc(length(mesh.x),length(mesh.x),9*length(mesh.x));
y = zeros(length(mesh.x),2);
for c = mesh.conn
    xe3 = mesh.x(:,c);
    de = T(c)';
    Ae = zeros(length(c));
    for q = quad
        [Nq,dNdpq] = shape_tri3(q);
        J3 = xe3*dNdpq;
        dNdxq = dNdpq/J3;
        qflux = -kappa*de*dNdxq;
```

```
        Ae = Ae + Nq*Nq'*det(J3)*q(end);
        y(c,:) = y(c,:) + Nq*qflux*det(J3)*q(end);
    end
    A(c,c) = A(c,c) + Ae;
end
qflux = A\y;
for c = 1:length(qflux)
    qfluxMag(c) = sqrt(qflux(c,1)^2+qflux(c,2)^2);
end
maxFlux = max(qfluxMag)


%%
    figure(1)
    p.faces = mesh.conn';
    p.vertices = mesh.x';
    p.facecolor = 'interp';
    p.facevertexcdata = T;
    patch(p)
    c1=colorbar;
    colormap(jet(256))
    c1.Label.String='temperature K';
    xlabel('X position (m)');
    ylabel('y Position (m)');
    title('temperature Contour for 20mm tri3 elements');
    axis equal;

    figure(2);
    p.faces = mesh.conn';
    p.vertices = mesh.x';
    p.facecolor = 'interp';
    p.facevertexcdata = qfluxMag';
    patch(p)
    c2=colorbar;
    colormap(jet(256))
    c2.Label.String='heat Flux W/m^2';
    xlabel('X position (m)');
    ylabel('y Position (m)');
    title('heat flux Contour for 20mm tri3 elements');
    axis equal;


%%
%manufactured solution
   % Example of a body heat source function.
     s = @(x,y)sin(x).*sin(y).*3.4e+1 %heat function
     qs= @(x,y)[-17*cos(x)*sin(y),-17*cos(y)*sin(x)]; %fluxstar
     Ts=@(x,y)sin(x).*sin(y);%temperaturestar
```

```
K=spalloc(length(mesh.x),length(mesh.x),9*length(mesh.x));
Fe=zeros(length(mesh.x),1);

%natural bc

 for c=mesh.conn
        xe=mesh.x(:,c);
        Ke=zeros(length(c));
        for q=[1/3;1/3;1/2]
            [N,dNdp]=shape_tri3(q);
            xq=xe*N;
            J=xe*dNdp;
            B=dNdp/J;

            Ke=Ke+B*kappa*B'*det(J)*q(end);
            f=s(xq(1,:),xq(2,:));
            Fe(c)=Fe(c)+N*f*det(J)*q(end);
        end
        K(c,c)=K(c,c)+Ke;

    end




   for ec=hole_edge_conn
        xe=mesh.x(:,ec);
        for q=quadrature(2)
            [N,dNdp]=shape2(q(1));
            J=xe*dNdp;
            n=[0 -1;1 0]*J/norm(J);
            x=xe*N;
            QS=qs(x(1),x(2));
            Q=dot(n,QS);
            Fe(ec)=Fe(ec)+N*Q*norm(J)*q(end);
        end
   end
     %essential bc
    fixed_nodes_temp=unique(perimeter_edge_conn);
    K(fixed_nodes_temp, :)= 0;
    K(fixed_nodes_temp,fixed_nodes_temp)=eye(length(fixed_nodes_temp));
    XY=mesh.x(:,fixed_nodes_temp);
    Fe(fixed_nodes_temp,1)=(Ts(XY(1,:),XY(2,:)))';


    T1=K\Fe;


    %
```

```
    figure(3)
    p.faces = mesh.conn';
    p.vertices = mesh.x';
    p.facecolor = 'interp';
    p.facevertexcdata = T1;
    patch(p)
    c1=colorbar;
    colormap(jet(256))
    c1.Label.String='temperature K';
    xlabel('X position (m)');
    ylabel('y Position (m)');
    title('temperature Contour (T_Exact) ');
    axis equal;

%%
%error norm

% e_L2=[];
% e_en=[];

e1_num=0;
e1_den=0;
for cerror=mesh.conn
    xerror=mesh.x(:,cerror);

  for q=quadrature(2)
      [N_error,dNdp_error]=shape_tri3(q);

      xee=xerror*N_error;
      T2=T1(cerror)'*N_error;
      J=xerror*dNdp_error;
      Temp_Exact=(Ts(xee(1),xee(2)));
      e1_num=(Temp_Exact-T2)^2;
      e1_den=Temp_Exact^2;

   end
end

     e1=sqrt(e1_num/e1_den)


%%
%convergence rate of tri3 element
e1 = [-1.044793462       %error norms for different element size
-2.318758763
-2.346787486
-2.744727495
]
```

```
h = [1.903089987
1.301029996
1
0.698970004
]
figure(4)
p = polyfit(h,e1,1);
plot(p);
slope = p(1)
xlabel('Log(h)');
ylabel('Log(E1)');
title('Log-Log Plot for Error norm E1');
```

```
%     p.faces = mesh.conn';
%     p.vertices = mesh.x';
%     p.facecolor = 'interp';
%     p.facevertexcdata = T;
%     patch(p)
%     c3=colorbar;
%     colormap(jet(256))
%     c1.Label.String='temperature K';
%     xlabel('X position (m)');
%     ylabel('y Position (m)');
%     title('temperature Contour for 20mm tri3 elements');
%     axis equal;

%

function [N,dNdp] = shape2(p) %shape function curve for linear function| 2 node
N   = (1/2)*[1-p;1+p]; %linear function
dNdp= [-0.5;0.5];  %derivative of linear function
end

function[N,dNdp]=shape_tri3(p)
N = [p(1);p(2);1-p(1)-p(2)];
dNdp =[1, 0; 0, 1; -1, -1];
```

```
end

function[N,dNdp]=shape_tri6(p)
N = [p(1)*(2*p(1)-1); p(2)*(2*p(2)-1); (1-p(1)-p(2))*(2*(1-p(1)-p(2))-1); 4*p(1)*p ↙
(2); 4*p(2)*(1-p(1)-p(2)); 4*p(1)*(1-p(1)-p(2))];
dNdp = [4*p(1) - 1,0; 0,4*p(2) - 1; 4*p(1)-3+4*p(2),-3+4*p(1)+4*p(2); 4*p(2), 4*p(1); ↙
-4*p(2), 4-4*p(1)-8*p(2);  4-8*p(1)-4*p(2), -4*p(1)];
end

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
```