Spring 2012

Numerical Solution of Two-Dimensional, Transient Temperature Distribution in a Rectangular Domain

MAE 3187 Computer Project

Ben Yarmis
Code Done in MATLAB
4/10/2012

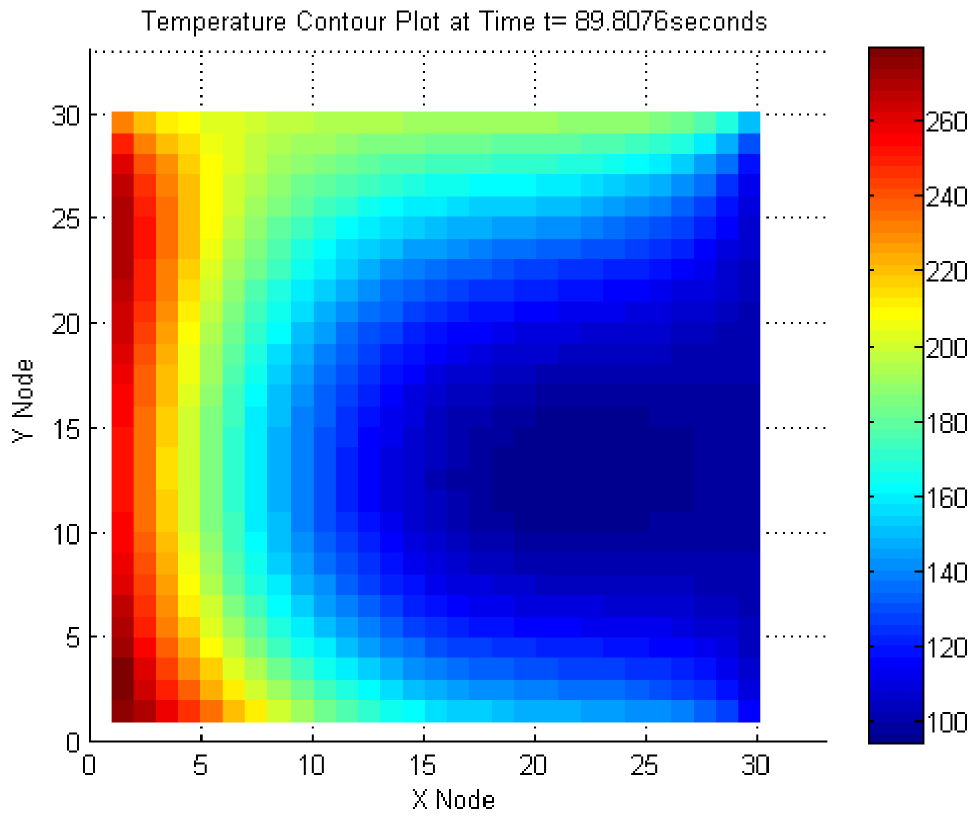
This problem can be simplified into a two-dimensional problem since there are no changes in the z-direction. Everything that happens along an edge happens along the whole edge face. There are also no differences between the top and the bottom of the aluminum slab, causing the temperature to remain constant throughout a vertical line. Thus, the problem can be simplified to a two-dimensional problem with the temperature being calculated along one face and extrapolated up if so desired.

The equations for the corner, edge, and node temperatures can be seen in the functions `tcorner`, `tledge` for the left edge, `tbedge` for the bottom edge, and `tnode`, respectively.

<u>30 Nodes, g=0</u>

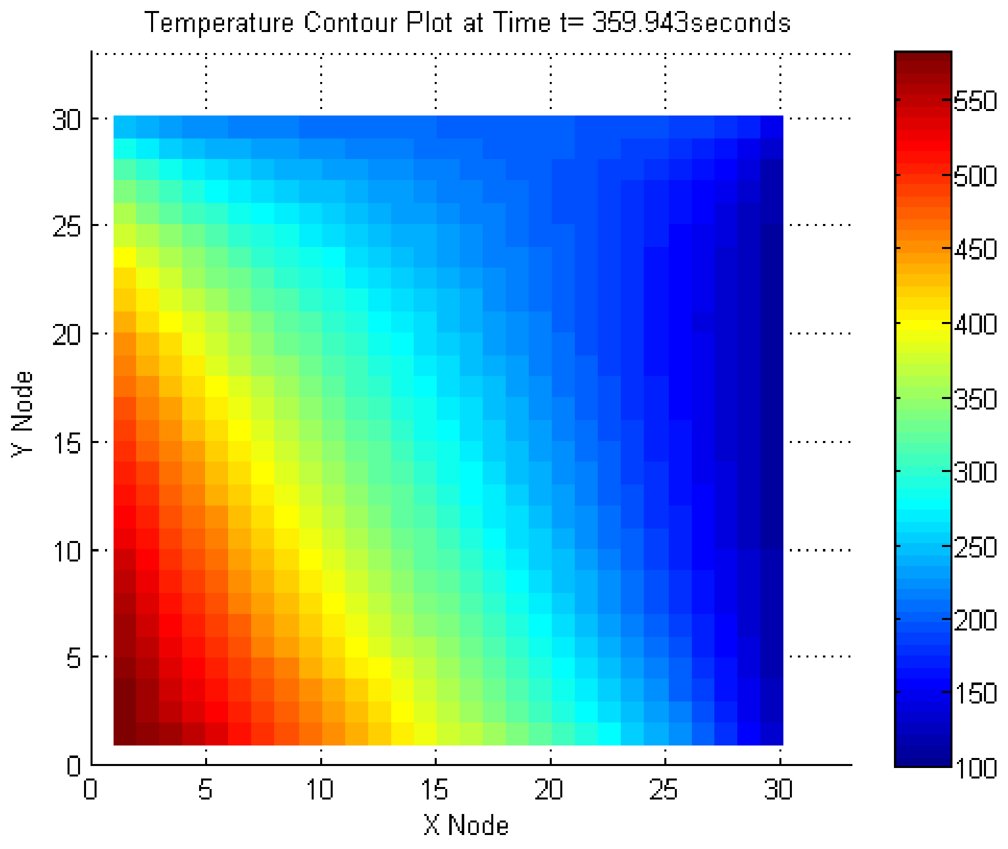
90 Seconds

```
Note: Case does not matter when entering text responses

How long should the program run for (seconds)?: 90
Should the program converge (Y is suggested)(Y/N)?: Y
Should there be heat generation (Y/N)?: N
When should the temperature be plotted (During/End/Never)?: E

The maximum temperature is 278.534939 degrees K

Elapsed time is 12.405768 seconds.
```



360 Seconds

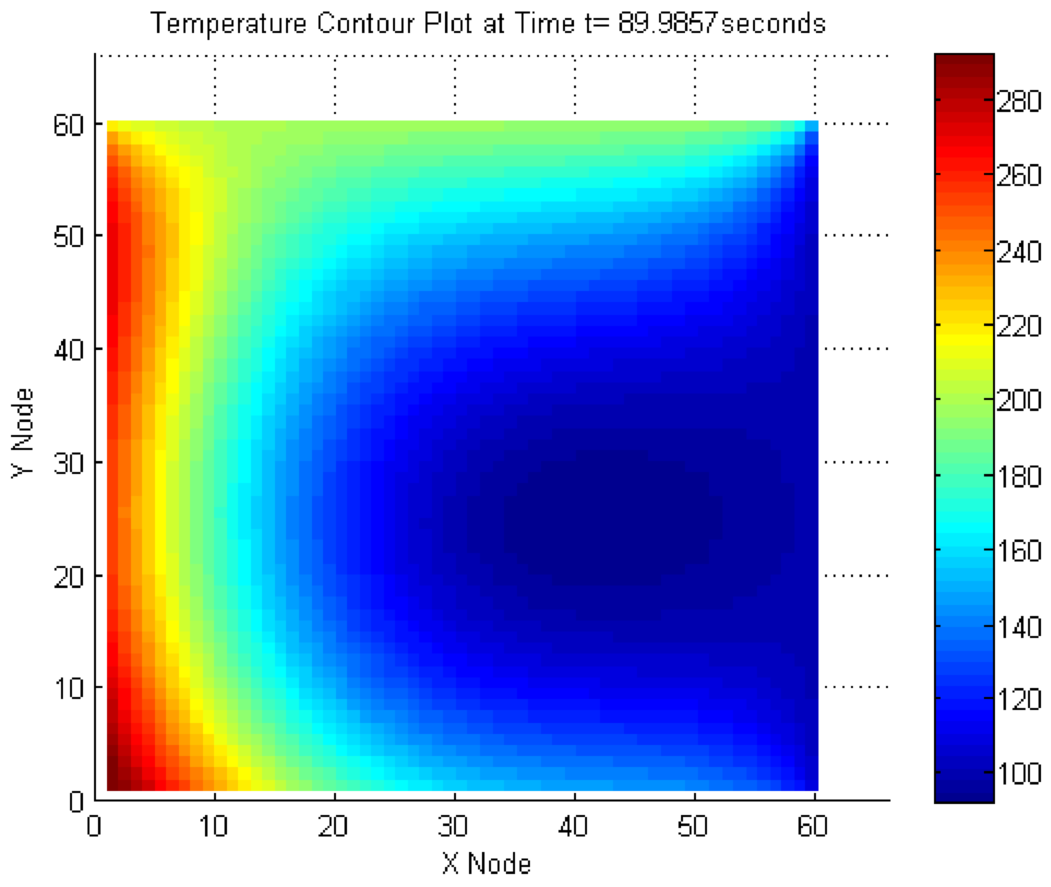
```
Note: Case does not matter when entering text responses

How long should the program run for (seconds)?: 360
Should the program converge (Y is suggested)(Y/N)?: Y
Should there be heat generation (Y/N)?: N
When should the temperature be plotted (During/End/Never)?: E

The maximum temperature is 582.041031 degrees K

Elapsed time is 14.902802 seconds.
```



60 Nodes, g=0 W/cm^3

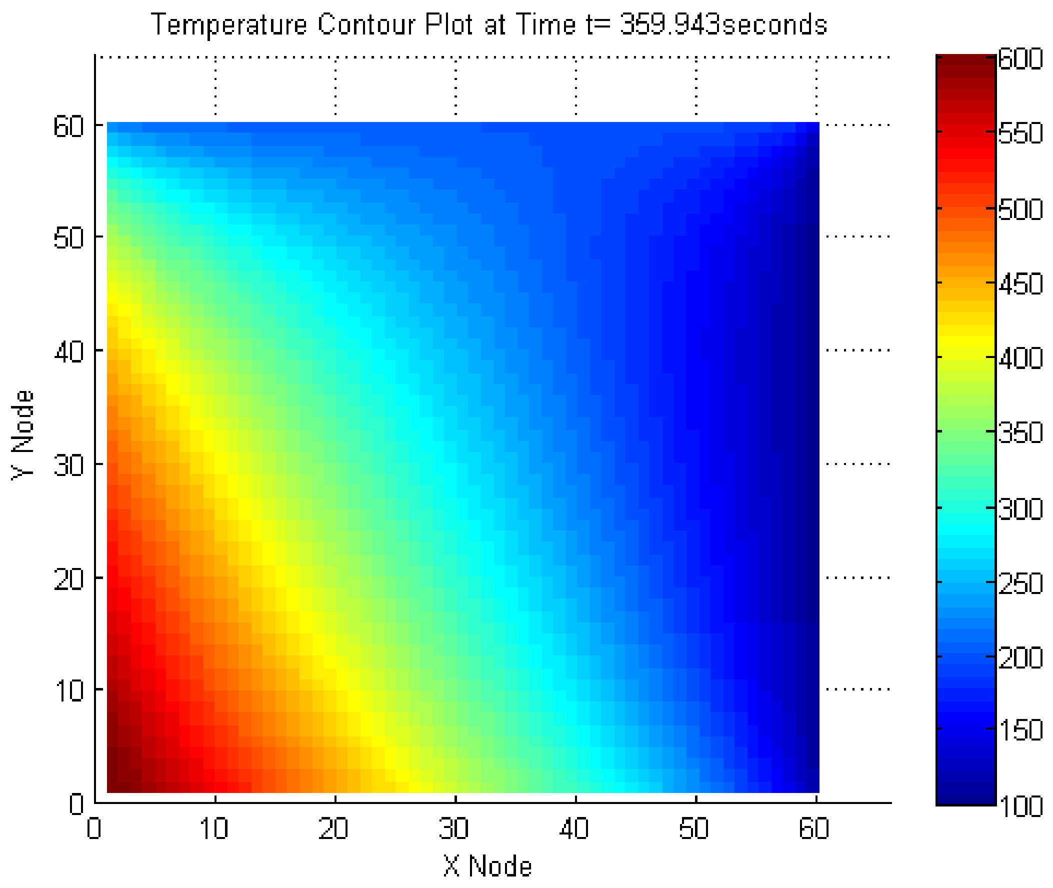
90 Seconds

```
Note: Case does not matter when entering text responses

How long should the program run for (seconds)?: 90
Should the program converge (Y is suggested)(Y/N)?: Y
Should there be heat generation (Y/N)?: N
When should the temperature be plotted (During/End/Never)?: E

The maximum temperature is 291.925976 degrees K

Elapsed time is 32.621137 seconds.
```



360 Seconds

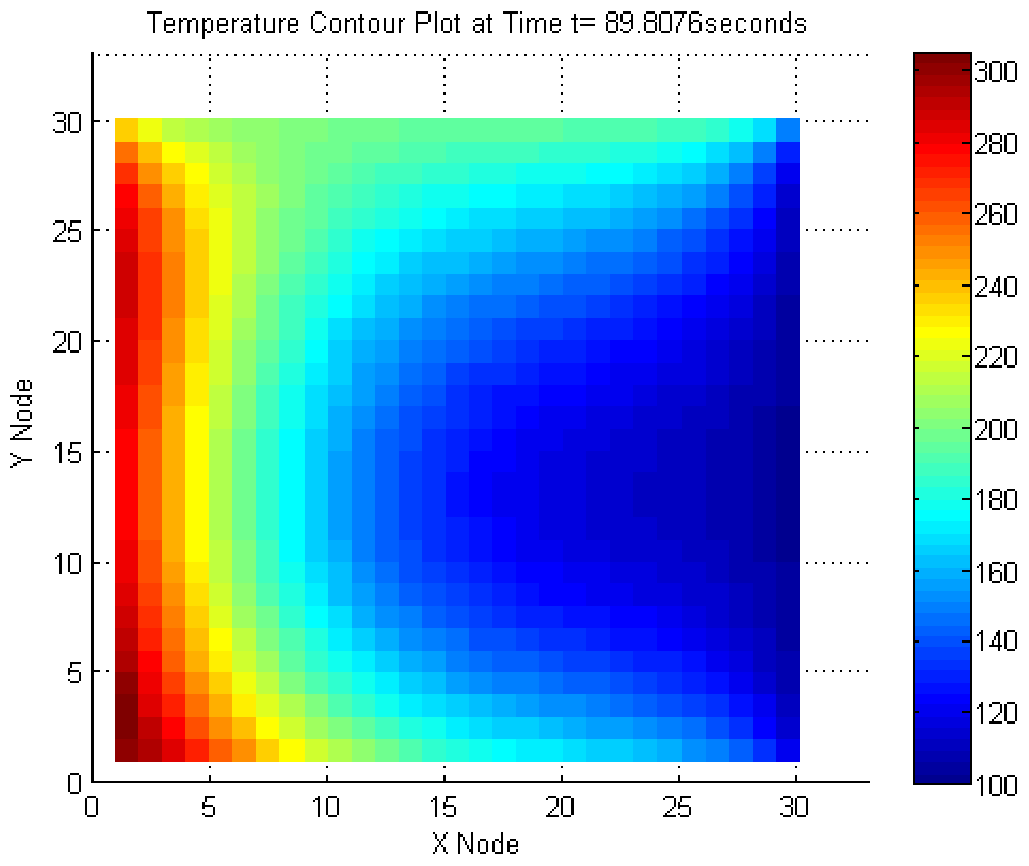
```
Note: Case does not matter when entering text responses

How long should the program run for (seconds)?: 360
Should the program converge (Y is suggested)(Y/N)?: Y
Should there be heat generation (Y/N)?: N
When should the temperature be plotted (During/End/Never)?: E

The maximum temperature is 601.069861 degrees K

Elapsed time is 107.063749 seconds.
```



30 Nodes, g=1 W/cm^3

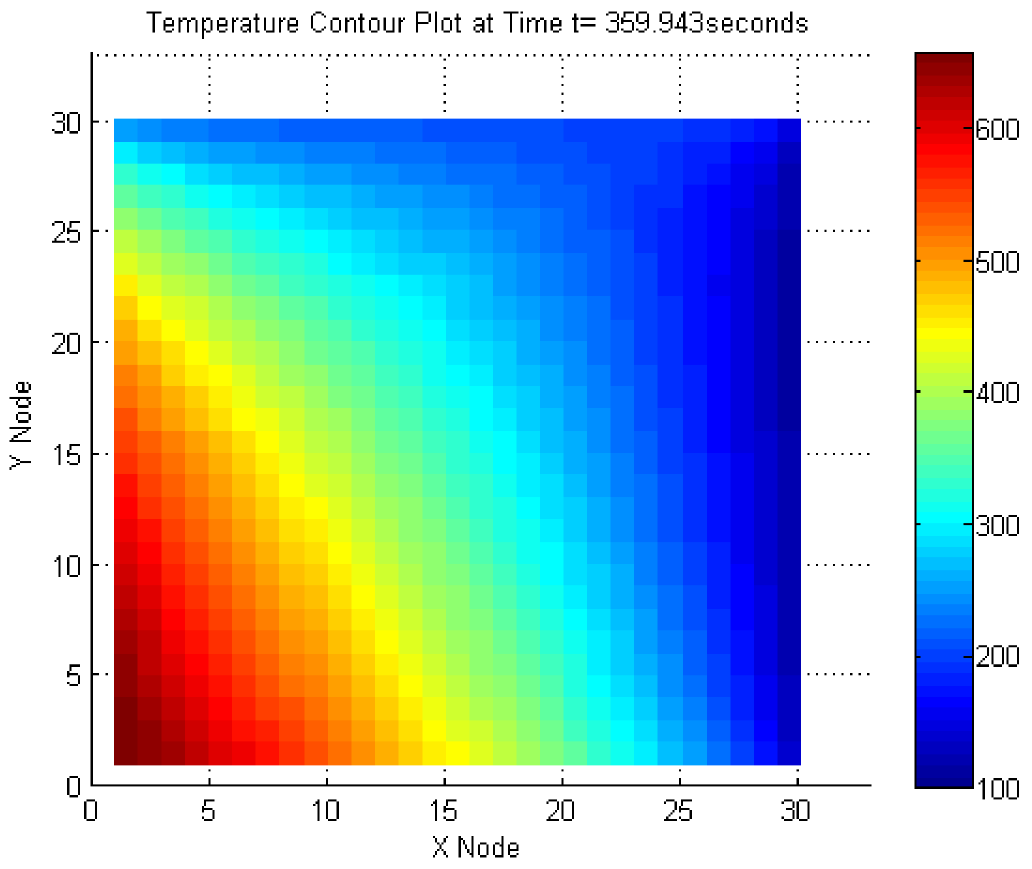
90 Seconds

```
Note: Case does not matter when entering text responses

How long should the program run for (seconds)?: 90
Should the program converge (Y is suggested)(Y/N)?: Y
Should there be heat generation (Y/N)?: Y
When should the temperature be plotted (During/End/Never)?: E

The maximum temperature is 304.766487 degrees K

Elapsed time is 8.663772 seconds.
```



360 Seconds

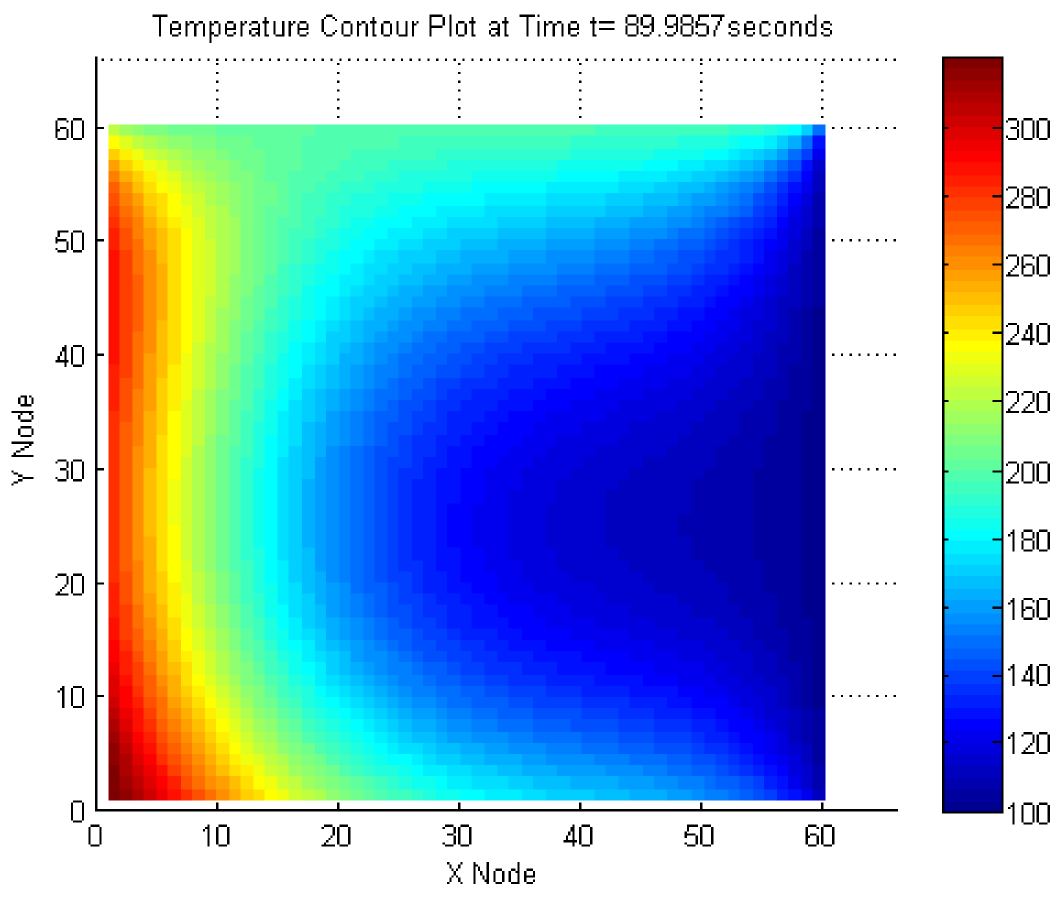
```
Note: Case does not matter when entering text responses

How long should the program run for (seconds)?: 360
Should the program converge (Y is suggested)(Y/N)?: Y
Should there be heat generation (Y/N)?: Y
When should the temperature be plotted (During/End/Never)?: E

The maximum temperature is 656.007221 degrees K

Elapsed time is 19.493767 seconds.
```



60 Nodes, g=1 W/cm^3

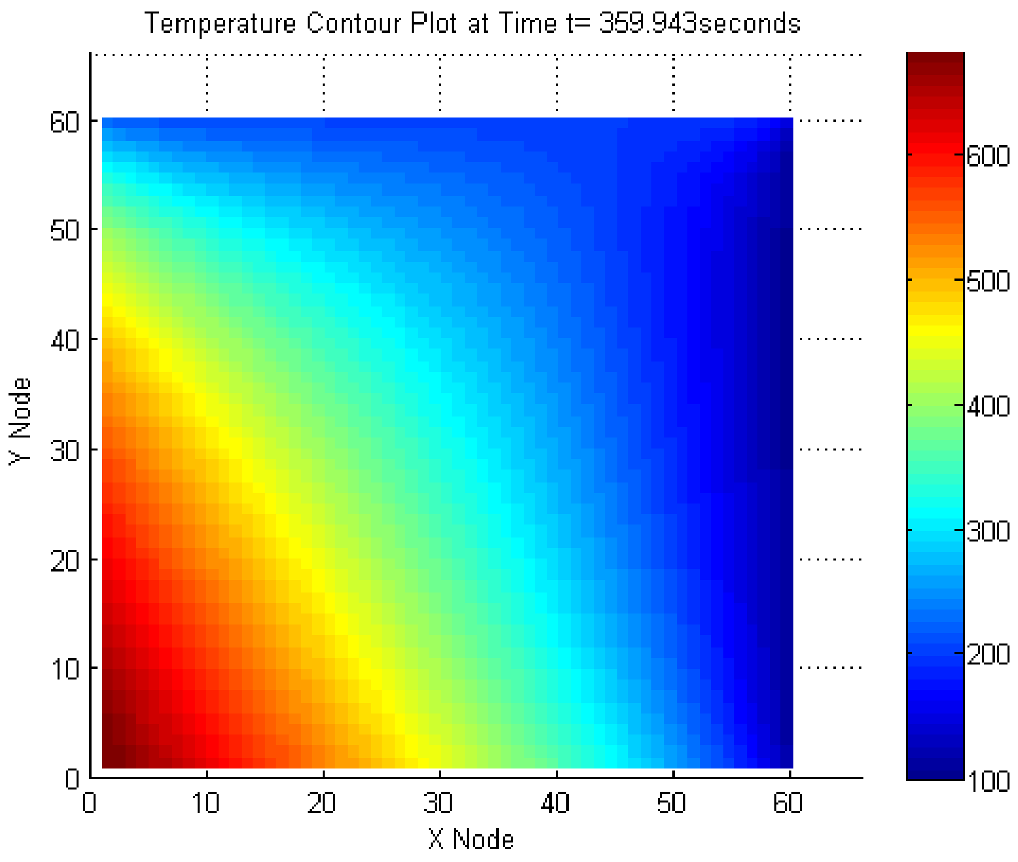
90 Seconds

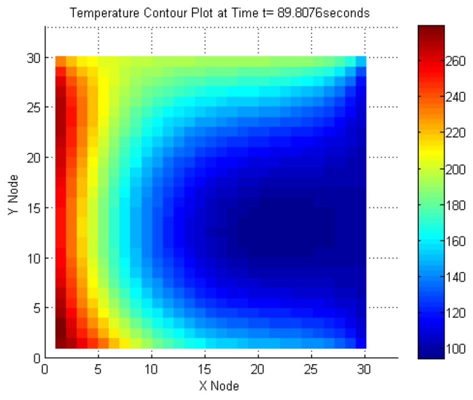
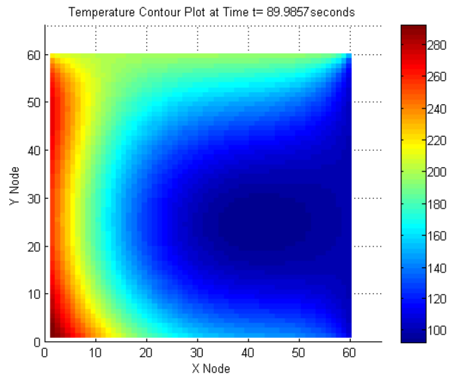
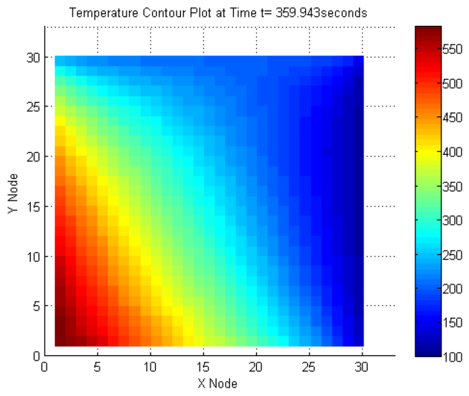
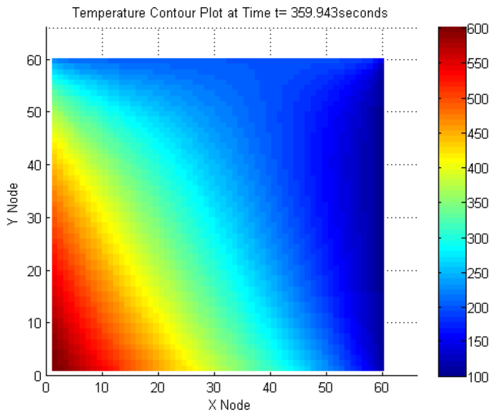
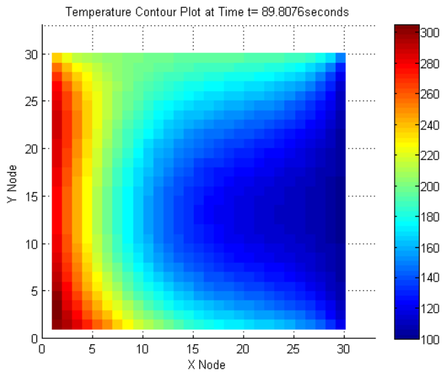
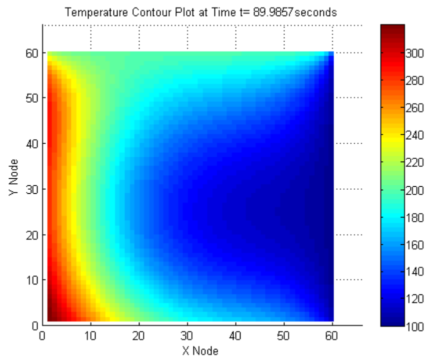
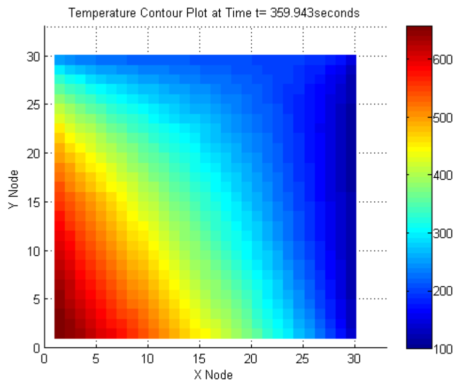
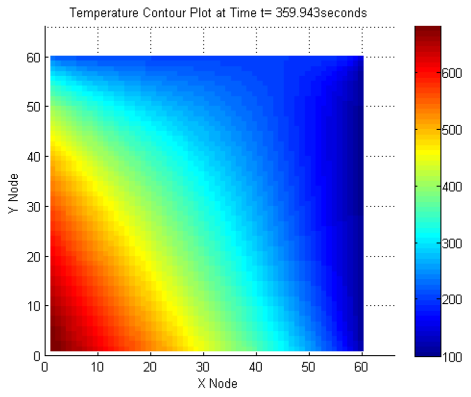
```
Note: Case does not matter when entering text responses

How long should the program run for (seconds)?: 90
Should the program converge (Y is suggested)(Y/N)?: Y
Should there be heat generation (Y/N)?: Y
When should the temperature be plotted (During/End/Never)?: E

The maximum temperature is 319.965116 degrees K

Elapsed time is 39.147708 seconds.
```



360 Seconds

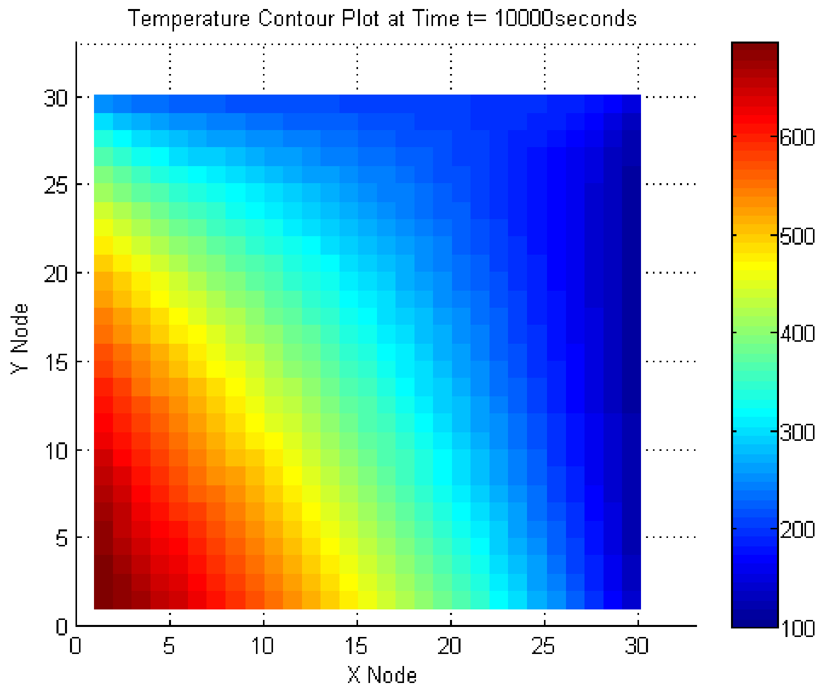
```
Note: Case does not matter when entering text responses

How long should the program run for (seconds)?: 360
Should the program converge (Y is suggested)(Y/N)?: Y
Should there be heat generation (Y/N)?: Y
When should the temperature be plotted (During/End/Never)?: E

The maximum temperature is 679.793983 degrees K

Elapsed time is 108.583868 seconds.
```



	30 Nodes	60 Nodes
90 Second , No g		
360 Second , No g		
90 Second , Yes g		
360 Second , Yes g		

As evident by the above table, increasing the mesh, while improving the accuracy of the temperature data, does not significantly change the overall results of the calculations.

<u>Steady State (t=10,000 and 10,500 seconds)</u>



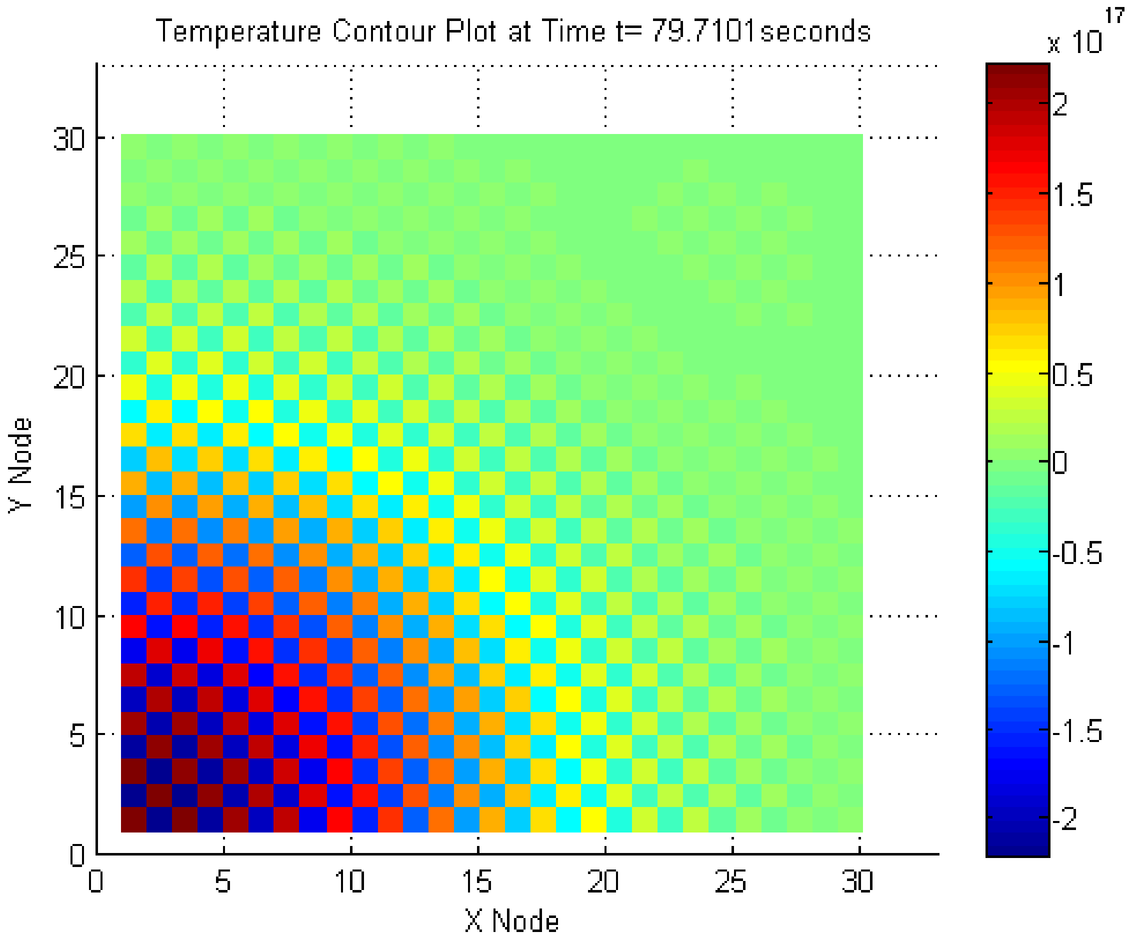
```
The maximum temperature
is 693.939626 degrees K
```

From the modified code to determine the minimum steady-state time

```
The time it takes to reach steady-state is 1606.794963 seconds

The maximum temperature is 693.703183 degrees K

Elapsed time is 32.786768 seconds.
```

Numerical Instability

```
Note: Case does not matter when entering text responses

How long should the program run for (seconds)?: 80
Should the program converge (Y is suggested)(Y/N)?: N
Should there be heat generation (Y/N)?: N
When should the temperature be plotted (During/End/Never)?: E

The maximum temperature is 221708057999440990.000000 degrees K

Elapsed time is 10.265288 seconds.
```



This output demonstrates the concept of garbage-in, garbage-out and numerical instability. The temperatures vary widely over short distances and are unreasonably high.

Code

Note that this code is for one instance of solving the problem and variables can and do change to change the mesh size, among other parameters. This code also prompts the user for many parameters that change based on the problem being solved.

```
clc
clear all
close all
tic

disp('Note: Case does not matter when entering text responses')
disp(' ')

%This allows the 'getfigure' command to grab the whole window, as opposed
%to part of the window that renders due to Windows 7 aero effects
opengl('software')

k=250; %W/(m-k)
alpha=8.418e-5; %Thermal diffusivity in (m^2)/s
dx=.01;
xmax=.3; ymax=.3; thick=.02;
nx=xmax/dx;

%User Input
tmax=input('How long should the program run for (seconds)?: ');

[deltat g]=converge(alpha,dx,1.1);
%End User Input
Fo=alpha*deltat/dx^2; %Fourier Number

t=0:deltat:tmax;
[timemax ~]=size(t');

T=zeros(nx,nx,timemax);

q1=100*(10000); %Watts per meter squared
q2=50* (10000); %Watts per meter squared

%The temperatures along the edges and initally along the internal nodes
T1=100; %K
T2=200; %K
Ti=10;

for i=1:nx
    for j=1:nx
        T(i,j,1)=Ti;
    end
end

%Sets the initial conditions along X=L and Y=L
T(1,:,1) =T2;
T(:,nx,1)=T1;
```

```
%User Input
plotnum=plotq();
%End User Input

for time=1:timemax-1
    for i=2:nx
        for j=1:nx-1

            %Corner Node
            if j==1 && i==nx
                T(i,j,time+1)=tcorner(T(i,j,time),T(i-1,j,time),...
                    T(i,j+1,time), Fo);

            %Bottom Edge
            elseif i==nx && j~=nx
                T(i,j,time+1)=tbedge(T(i,j,time),T(i-1,j,time),...
                    T(i,j-1,time),T(i,j+1,time),Fo,q2,k,dx);

            %Left Edge
            elseif i~=1 && j==1
                T(i,j,time+1)=tledge(T(i,j,time),T(i,j+1,time),...
                    T(i-1,j,time),T(i+1,j,time),Fo,q1,k,dx);

            %Internal Node
            elseif i~=1 && j~=1 && i~=nx && j~=nx
                T(i,j,time+1)=tnode(T(i,j,time),T(i,j+1,time),...
                    T(i+1,j,time),T(i,j-1,time),T(i-1,j,time),Fo,g,k,dx);
            end

            %Manually sets the temperature along the upper and right edges
            %along with the corner where they meet
            T(1,:,time+1) =T2;
            T(:,nx,time+1)=T1;
            T(1,nx,time+1)=(T1+T2)/2;
        end
    end

    if plotnum==1
        S=surf(flipud(T(:,:,time)),'LineStyle','none');
        axis([0 1.1*xmax/dx 0 1.2*xmax/dx])
        xlabel('X Node'); ylabel('Y Node');
        text(.125*xmax/dx, 1.1*xmax/dx,...
            ['Temperature Contour Plot at Time t= ', num2str(t(time)),
'seconds'])
        colorbar
        M(time)=getframe; %#ok<SAGROW>
    end

end

if plotnum==1
    movie(M,5,90)
elseif plotnum==0
    surf(flipud(T(:,:,timemax)),'LineStyle','none')
    xlabel('X Node'); ylabel('Y Node');
    title(['Temperature Contour Plot at Time t= ', num2str(t(timemax)),
'seconds']);
    axis([0 1.1*xmax/dx 0 1.1*xmax/dx])
    colorbar
    fprintf('\nThe maximum temperature is %f degrees
K\n\n',max(max(T(:,:,timemax))))
elseif plotnum==-1
    fprintf('\nThe maximum temperature is %f degrees
K\n\n',max(max(T(:,:,timemax))))
end

toc
```

<u>Code for Finding the Minimum Steady-State Time</u>

Note that the majority of the code is the same, except with the change in the major loop as noted below

```
time=1;
difference=100;
while difference>.1
    for i=2:nx
        for j=1:nx-1

            %Corner Node
            if j==1 && i==nx
                T(i,j,time+1)=tcorner(T(i,j,time),T(i-1,j,time),...
                    T(i,j+1,time), Fo);

            %Bottom Edge
            elseif i==nx && j~=nx
                T(i,j,time+1)=tbedge(T(i,j,time),T(i-1,j,time),...
                    T(i,j-1,time),T(i,j+1,time),Fo,q2,k,dx);

            %Left Edge
            elseif i~=1 && j==1
                T(i,j,time+1)=tledge(T(i,j,time),T(i,j+1,time),...
                    T(i-1,j,time),T(i+1,j,time),Fo,q1,k,dx);

            %Internal Node
            elseif i~=1 && j~=1 && i~=nx && j~=nx
                T(i,j,time+1)=tnode(T(i,j,time),T(i,j+1,time),...
                    T(i+1,j,time),T(i,j-1,time),T(i-1,j,time),Fo,g,k,dx);
            end

            %Manually sets the temperature along the upper and right edges
            %along with the corner where they meet
            T(1,:,time+1) =T2;
            T(:,nx,time+1)=T1;
            T(1,nx,time+1)=(T1+T2)/2;
        end
    end

time=time+1;
```

```
difference=sum(sum(T(:,:,time)-T(:,:,time-1)));
end
```

<u>Functions</u>

Tcorner

```
function [t]=tcorner(T,Tm10,T0m1,Fo)
%A function for computing the subsequent temperature of nodes along corner
%nodes
%
%Inputs are:
%               T    -- The temperature at the node at a given time
%               Tm10 -- The temperature at node m-1 at a given time
%               T0m1 -- The temperature at node m-1 at a given time
%               Fo   -- The Fourier Number

t=2*Fo*(Tm10+T0m1)+(1-4*Fo)*T;

end
```

tledge

```
function [t]=tledge(T,T10,T01,T0m1,Fo,q,k,dx)
%A function for computing the subsequent temperature of nodes along the
%left edge.
%
%Inputs are:
%               T    -- The temperature at the node at a given time
%               T10  -- The temperature at node m+1 at a given time
%               T01  -- The temperature at node n+1 at a given time
%               T0m1 -- The temperature at node n-1 at a given time
%               Fo   -- The Fourier Number
%                q   -- The heat flux along the left edge
%                k   -- The conduction coefficient
%                dx  -- The step distance

t=Fo*(2*T10+T01+T0m1+q/k*dx)+(1-4*Fo)*T;

end
```

tbedge

```
function [t]=tbedge(T,T10,T01,Tm10,Fo,q,k,dx)
%A function for computing the subsequent temperature of nodes along the
%bottom edge
%
%Inputs are:
%               T    -- The temperature at the node at a given time
%               T10  -- The temperature at node m+1 at a given time
%               T01  -- The temperature at node n+1 at a given time
%               Tm10 -- The temperature at node m-1 at a given time
%               Fo   -- The Fourier Number
%                q   -- The heat flux along the bottom edge
%                k   -- The conduction coefficient
%                dx  -- The step distance

t=Fo*(2*T10+T01+Tm10+(q/k)*dx)+(1-4*Fo)*T;

end
```

tnode

```
function [t]=tnode(T,T10,T01,Tm10,T0m1,Fo,g,k,dx)
%A function for computing the subsequent temperature of nodes along an
%internal node
%
%Inputs are:
%               T    -- The temperature at the node at a given time
%               T10  -- The temperature at node m+1 at a given time
%               T01  -- The temperature at node n+1 at a given time
%               Tm10 -- The temperature at node m-1 at a given time
%               T0m1 -- The temperature at node n-1 at a given time
%               Fo   -- The Fourier Number
%                g   -- The heat generation term
%                k   -- The heat conduction coefficient
%                dx  -- The step distance

t=Fo*(T10+Tm10+T01+T0m1+(g/k)*dx^2)+(1-4*Fo)*T;

end
```

plotq

```
function [n]=plotq()
%This function asks the user if the temperature should be plotted during
%the calculations or not.
%
%There are no inputs currently.

plotquestion=input('When should the temperature be plotted
(During/End/Never)?: ', 's');
plotansy=strcmpi('D',plotquestion);
plotansn=strcmpi('E',plotquestion);
plotans0=strcmpi('N',plotquestion);

if plotansy==1
    n=1;
elseif plotansn==1
    n=0;
elseif plotans0==1
    n=-1;
end
```

```
end
```

converge

```
function [deltat g]=converge(alpha,dx,fudge_factor)
%This function finds the timestep such that the function either converges
%or diverges.  It asks the user if the function should diverge or not.
%
%It also prompts the user for heat generation.

%Inputs are:
%            alpha          -- The thermal diffusivity of the material
%            dx             -- The step distance
%            fudge_factor   -- The factor by which, should it be
%                              decided that the simulation should
%                              diverge, how quickly it should diverge

converge=input('Should the program converge (Y is suggested)(Y/N)?: ', 's');
convergey=strcmpi('Y',converge);
convergen=strcmpi('N',converge);

if convergey==1
    deltat=.8*(1/(2*alpha))*(1/(1/(dx^2)+1/(dx^2)));
elseif convergen==1
    deltat=(1/(2*alpha))*(1/(1/(dx^2)+1/(dx^2)))*fudge_factor;
else
    disp('That is not a valid input')
end

ganswer=input('Should there be heat generation (Y/N)?: ', 's');
ganswersy=strcmpi('Y',ganswer);
ganswersn=strcmpi('N',ganswer);

if ganswersy==1
    g=1e6; %Watts per meter cubed
elseif ganswersn==1
    g=0;
else
    disp('That is not a valid input')
end

end
```