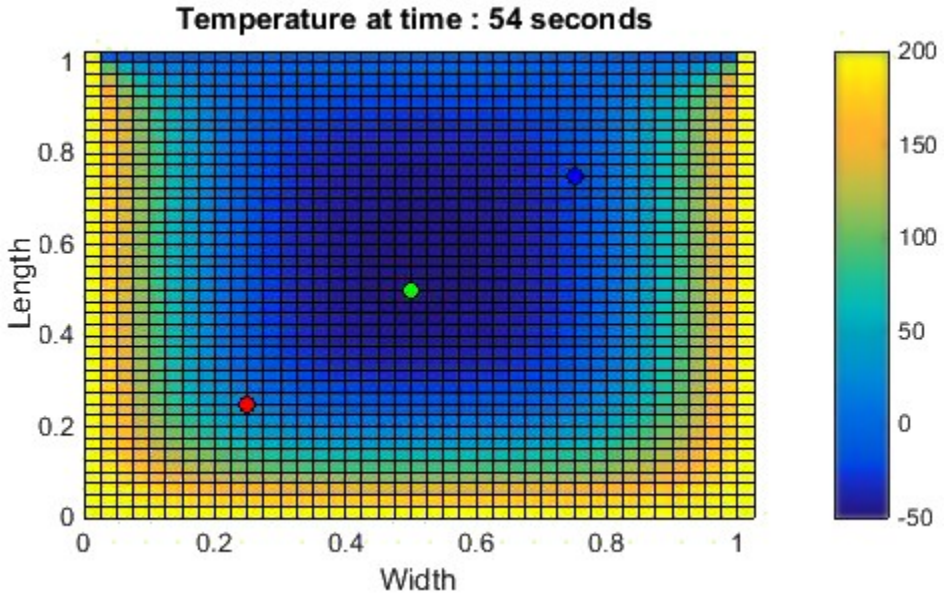
Heat Diffusion in 2D Square Plate Using Finite Difference Method with Steady-State Solution



Presented By:

Amr Mousa Mohamed

201305701

Supervised By:

Dr.Mohamed Tawfik

Eng.Heba Allaa-Eldin

Table of Contents

ABSTRACT:

The object of this project is to solve the 2D heat equation using finite difference method and to get the solution of diffusing the heat inside a square plate with specific boundary conditions.

It's a simple MATLAB code that can solve for different materials such as (copper, aluminum, silver, etc….) or it allows the user to add his own material by entering the thermal conductivity factor, specific heat and density. Many different boundary conditions that are fixed with time "Dirichlet Conditions" can be applied. It solves also for the steady-state temperature of the plate and tell the user the time this plate will take to reach this steady-state with an error tolerance selected before by the user. The accuracy of the solution will depend mainly on the number of nodes in x and y directions that can be selected also before meshing and start iterating for solution. The code can solve the time-derivative part of the equation with 2 ways (Euler and 2^{nd} order Runge-Kutte) and the space-derivatives with central finite difference

Finally after solution, Graphical simulation in time appears to show how the heat diffuses throughout the plate within time interval chosen.

Introduction:

In this course we learned about how to solve a complex problems in ordinary differential equations and partial differential equations using a simple method called finite difference.

A finite difference is a mathematical expression of the form f(x + b) − f(x + a). If a finite difference is divided by b − a, one gets a difference quotient. The approximation of derivatives by finite differences plays a central role in finite difference methods for the numerical solution of differential equations, especially boundary value problems.

In this report, we will use the central finite difference in space and forward finite difference in time to solve the heat equation in a 2D-Plate with some fixed initial conditions called "Dirichlet Conditions" to obtain a solution that describes the temperature of each node in space in any time within the interval entered before debugging.

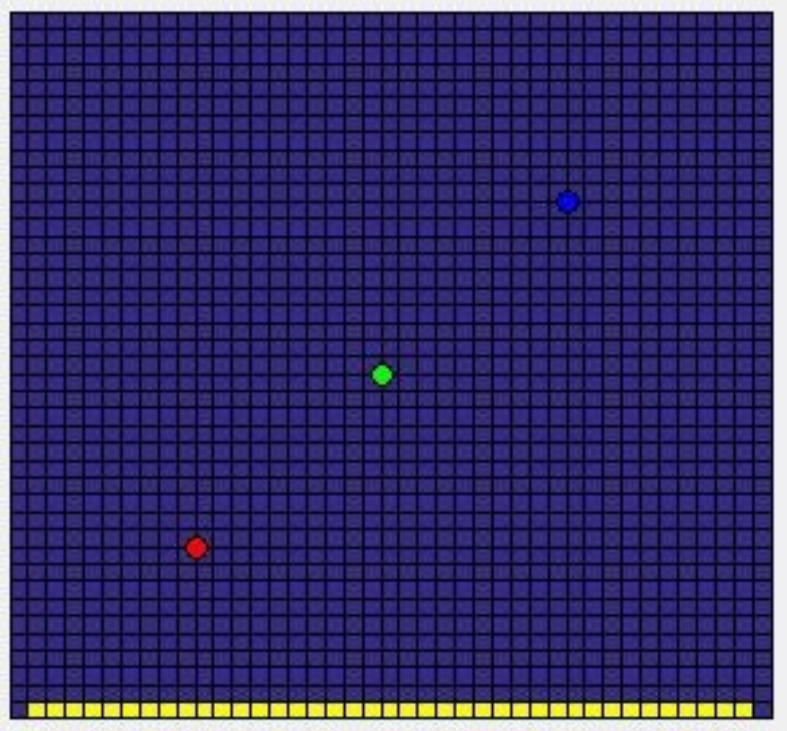
We will also find more about the steady state of heat diffusion throughout the plate which is defined as the state when the system reach it, the solution to the heat equation will independent on time. This can be interpreted physically as the change in temperature with time will become insignificant, can be neglected and the truncation error here will be defined as a tolerance.

Problem Statement:

Given:

- Initial temperature in a 2-D plate
- Boundary conditions along the boundaries of the plate.

Find: Temperature in the plate as a function of time and position.



Derivations:

1. <u>Solving the heat equation with central finite difference in position and forward finite difference in time using Euler method</u>

Given the heat equation in 2d

$$\rho C_p \frac{\partial T}{\partial t} = k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$

Where

- ρ is the material density
- C_p is the specific heat
- K is the thermal conductivity

T(x, 0, t) = given

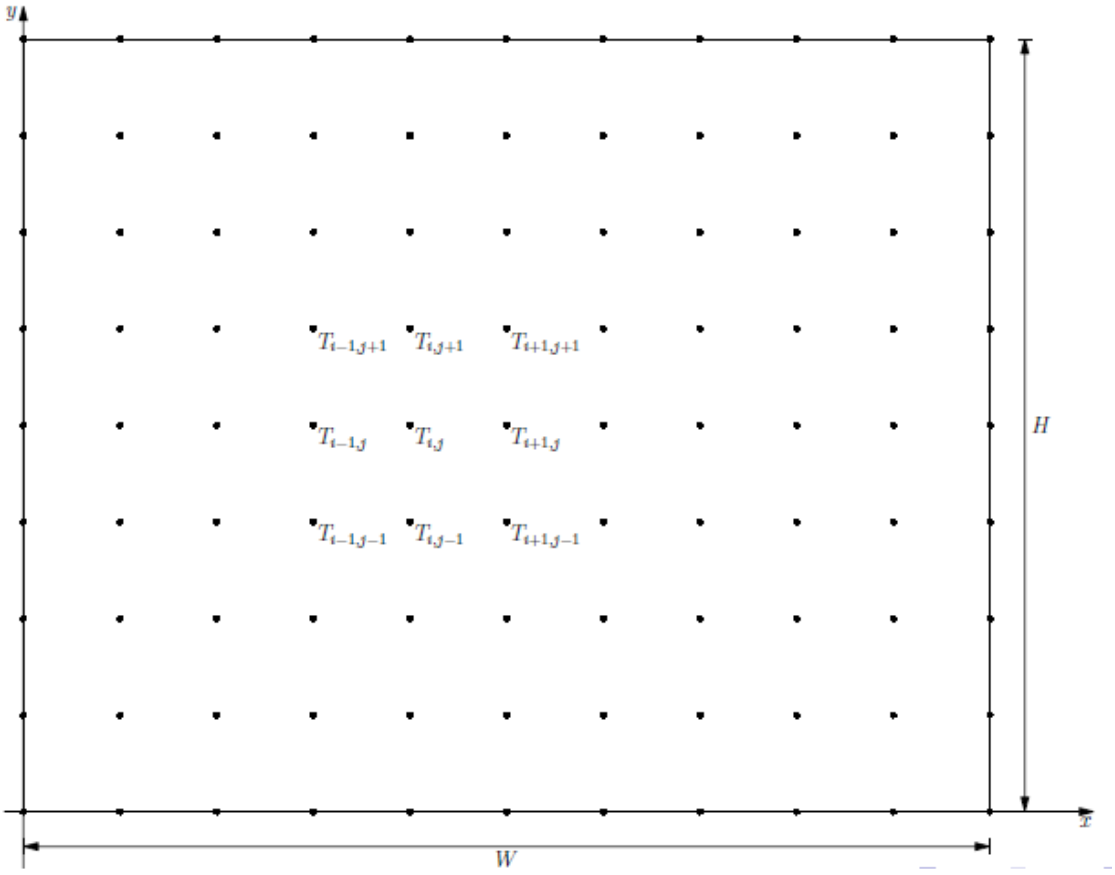
T(x, H, t) = given

T(0, y, t) = given

T(W, y, t) = given

T(x, y, 0) = given

Second step is to discretize the temperatures in the plate, and convert the heat equation to finite-difference form.



Let's agree on that notation first

$$T^{k}_{i,j}$$

i; j = location (node numbers)

k = time (time step number)

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

$$\frac{T^{k+1}_{i,j} - T^{k}_{i,j}}{\Delta t} = \alpha \left[\left(\frac{T^{k}_{i-1,j} - 2T^{k}_{i,j} + T^{k}_{i+1,j}}{\Delta x^2} \right) + \left(\frac{T^{k}_{i,j-1} - 2T^{k}_{i,j} + T^{k}_{i,j+1}}{\Delta y^2} \right) \right]$$

$$T_{i,j}^{k+1} = T_{i,j}^{k} + \alpha\,\Delta t\left[\left(\frac{T_{i-1,j}^{k} - 2T_{i,j}^{k} + T_{i+1,j}^{k}}{\Delta x^2}\right) + \left(\frac{T_{i,j-1}^{k} - 2T_{i,j}^{k} + T_{i,j+1}^{k}}{\Delta y^2}\right)\right]$$

$$T_{i,j}^{k+1} = T_{i,j}^{k} - \frac{2\alpha\,\Delta t\,T_{i,j}^{k}}{\Delta x^2} - \frac{2\alpha\,\Delta t\,T_{i,j}^{k}}{\Delta y^2} + \alpha\,\Delta t\left[\left(\frac{T_{i-1,j}^{k} + T_{i+1,j}^{k}}{\Delta x^2}\right) + \left(\frac{T_{i,j-1}^{k} + T_{i,j+1}^{k}}{\Delta y^2}\right)\right]$$

$$T_{i,j}^{k+1} = \left(1 - \frac{2\alpha\,\Delta t}{\Delta x^2} - \frac{2\alpha\,\Delta t}{\Delta y^2}\right)T_{i,j}^{k} + \alpha\,\Delta t\left[\left(\frac{T_{i-1,j}^{k} + T_{i+1,j}^{k}}{\Delta x^2}\right) + \left(\frac{T_{i,j-1}^{k} + T_{i,j+1}^{k}}{\Delta y^2}\right)\right]$$

Where $\left(1 - \frac{2\alpha\,\Delta t}{\Delta x^2} - \frac{2\alpha\,\Delta t}{\Delta y^2}\right) \geq 0$

$$1 \geq \left(\frac{2\alpha}{\Delta x^2} + \frac{2\alpha}{\Delta y^2}\right)\Delta t$$

$$\Delta t \leq \frac{1}{2\alpha\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)}$$ is the stability condition

and the final equation is

$$T_{i,j}^{k+1} = \left(1 - \left(\frac{2\alpha}{\Delta x^2} + \frac{2\alpha}{\Delta y^2}\right)\Delta t\right)T_{i,j}^{k} \quad \alpha\,\Delta t\left[\left(\frac{T_{i-1,j}^{k} + T_{i+1,j}^{k}}{\Delta x^2}\right) + \left(\frac{T_{i,j-1}^{k} + T_{i,j+1}^{k}}{\Delta y^2}\right)\right]$$

2. Solving the heat equation with central finite difference in position and forward finite difference in time using Euler method

Given the heat equation in 2d

$$\rho C_p \frac{\partial T}{\partial t} = k\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$

Where

- ρ is the material density
- C_p is the specific heat
- K is the thermal conductivity

$$T(x, 0, t) = \text{given}$$

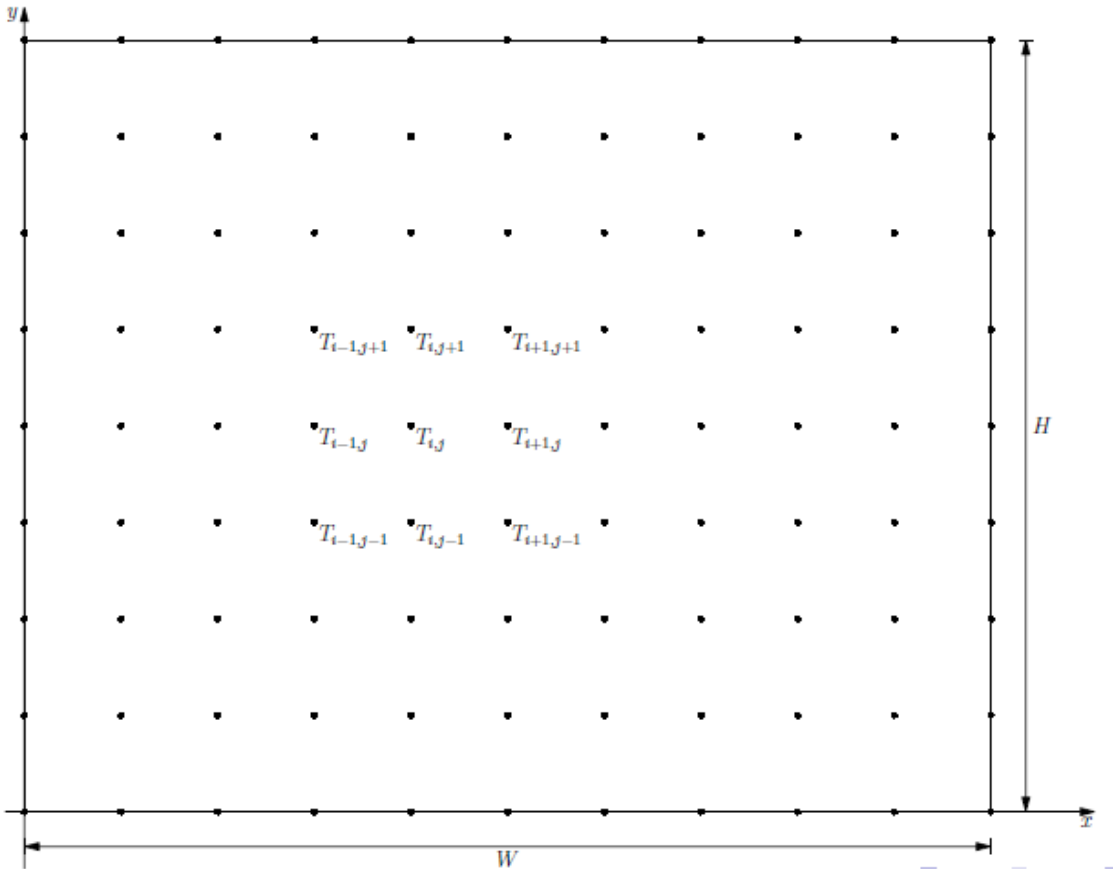
$$T(x, H, t) = \text{given}$$

$$T(0, y, t) = \text{given}$$

$$T(W, y, t) = \text{given}$$

$$T(x, y, 0) = \text{given}$$

Again we discretize the temperatures in the plate, and convert the heat equation to finite-difference form.



Same notation we have here

$$T^{k}_{i,j}$$

i; j = location (node numbers)

k = time (time step number)

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right)$$

$$\frac{\partial T}{\partial t} = \alpha \left[\left(\frac{T^k_{i-1,j} - 2T^k_{ij} + T^k_{i+1,j}}{\Delta x^2} \right) + \left(\frac{T^k_{ij-1} - 2T^k_{ij} + T^k_{ij+1}}{\Delta y^2} \right) \right]$$

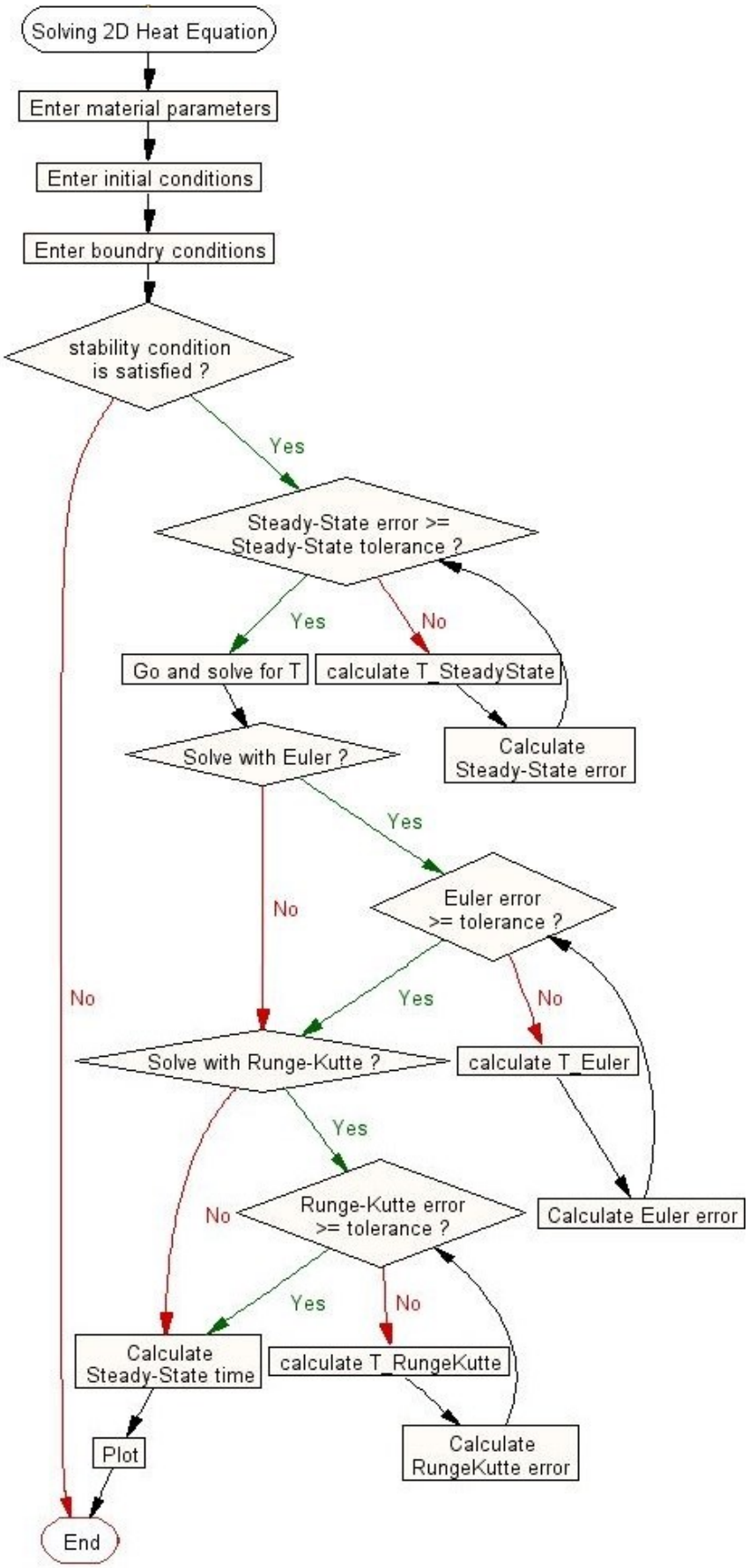
$$\frac{\partial T}{\partial t} = \qquad\qquad T(i,j)$$

$$k1 = T(i,j)$$

$$k2 = T(i,j) + k1\Delta t$$

$$T_{ij}^{k+1} = T_{ij}^{k} + \frac{\Delta t}{2}(k1 + k2)$$

Code Flowchart:



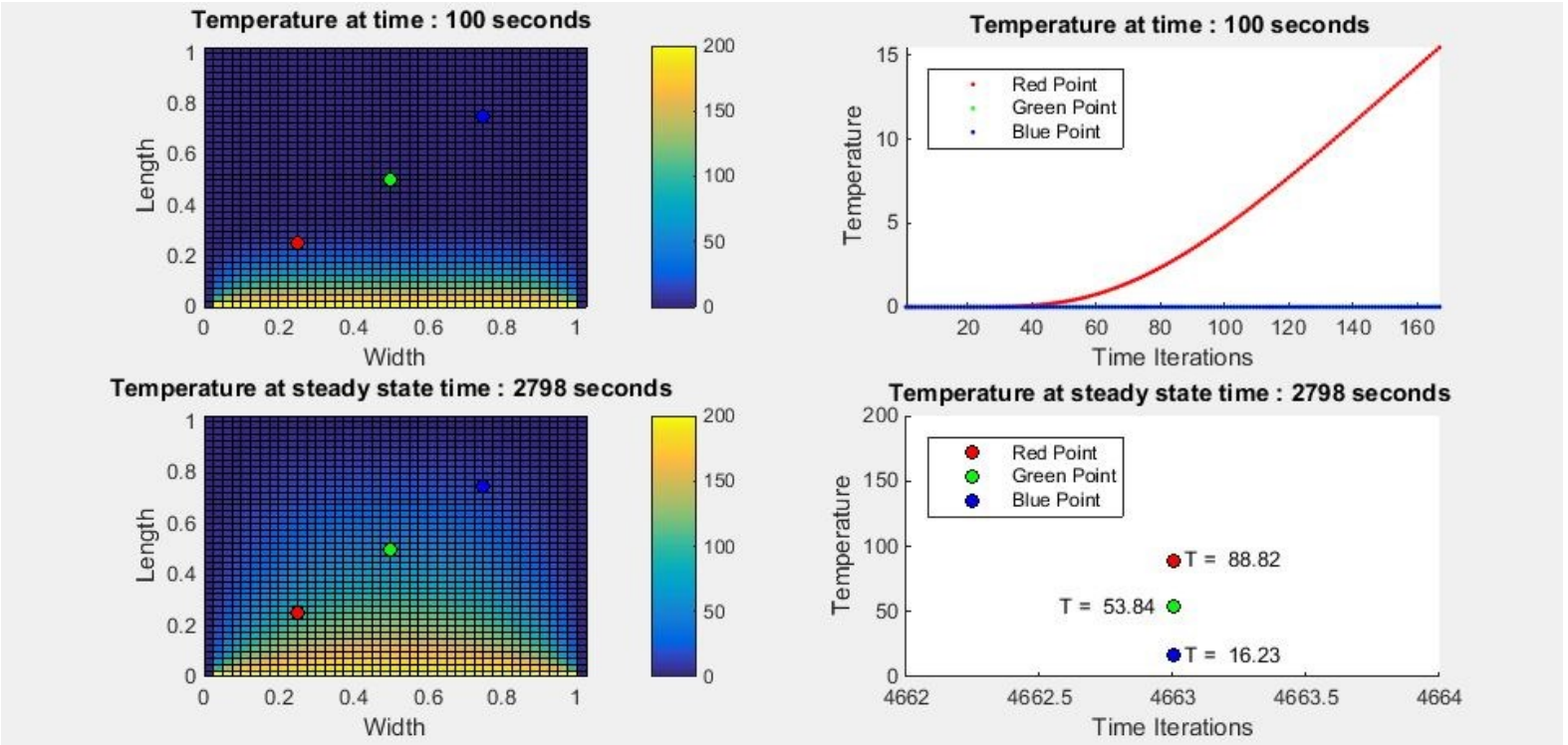
How to use:

1. Open code.m file
2. From line 24 : 52, we should uncomment your material or add your specific material if you want to use another one.
3. From line 55 : 58, Add your plate dimensions and the number of nodes in x and y directions
   Note that: increasing number of nodes means more accuracy but with low performance and long time for iterations, so start with 40 nodes and increase it incrementally if desired.
4. From line 59 : 64, Enter your problem initial conditions
5. From line 65 : 67, Enter the desired simulation time in seconds and your delta t .
6. From line 68 : 70, Enter numerical simulation tolerance which is default 0.5 deg Celsius and the steady-state tolerance which is default 0.001 deg Celsius.
7. Finally, run the code and select ‘Euler method’ or ‘Runge-Kutte’ to solve the time part of the equation with when a question dialog a

Results:

1. Different Materials with same conditions

Aluminum



Command Window

```
This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Aluminium
The solution is  based on "Dirichlet Boundry Conditions" with initial values
T(x,0,t)=200 , T(x,1,t)=0 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=0
The plate takes 2798 seconds to reach steady-state temperature with tolerence 0.50
Now, Simulation is running with final time 100 seconds and step 0.60 second
```

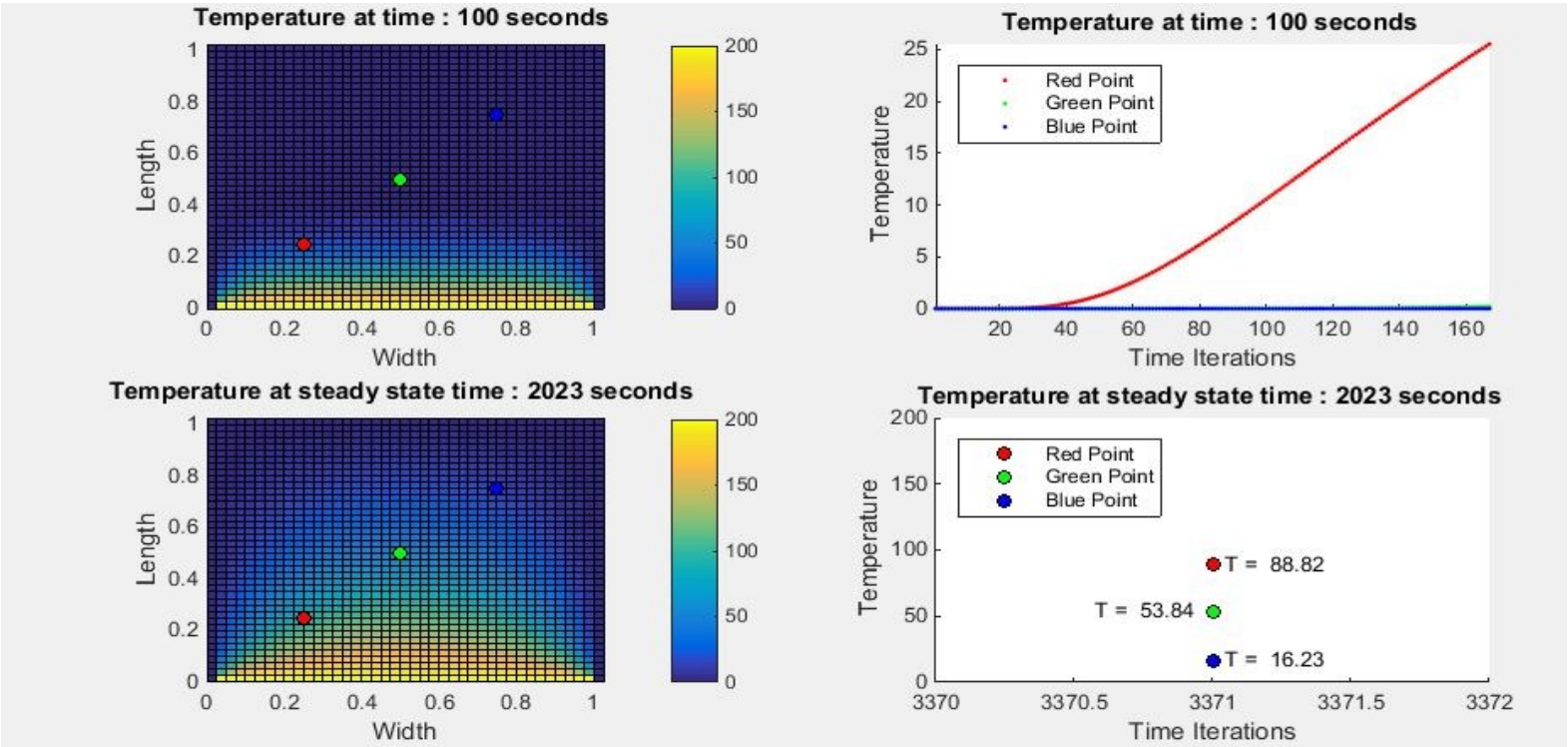

Comment:

In this trail the material was Aluminum and the boundary conditions as shown in the command window.

The diffusion of the heat was slow in comparison with the next two materials (copper and silver).

It took 2798 second to reach the Steady-State shown in the lower-left plot.

Copper




```
Command Window
This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Copper
The solution is  based on "Dirichlet Boundry Conditions" with initial values
T(x,0,t)=200 , T(x,1,t)=0 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=0
The plate takes 2023 seconds to reach steady-state temperature with tolerence 0.50
Now, Simulation is running with final time 100 seconds and step 0.60 second
```

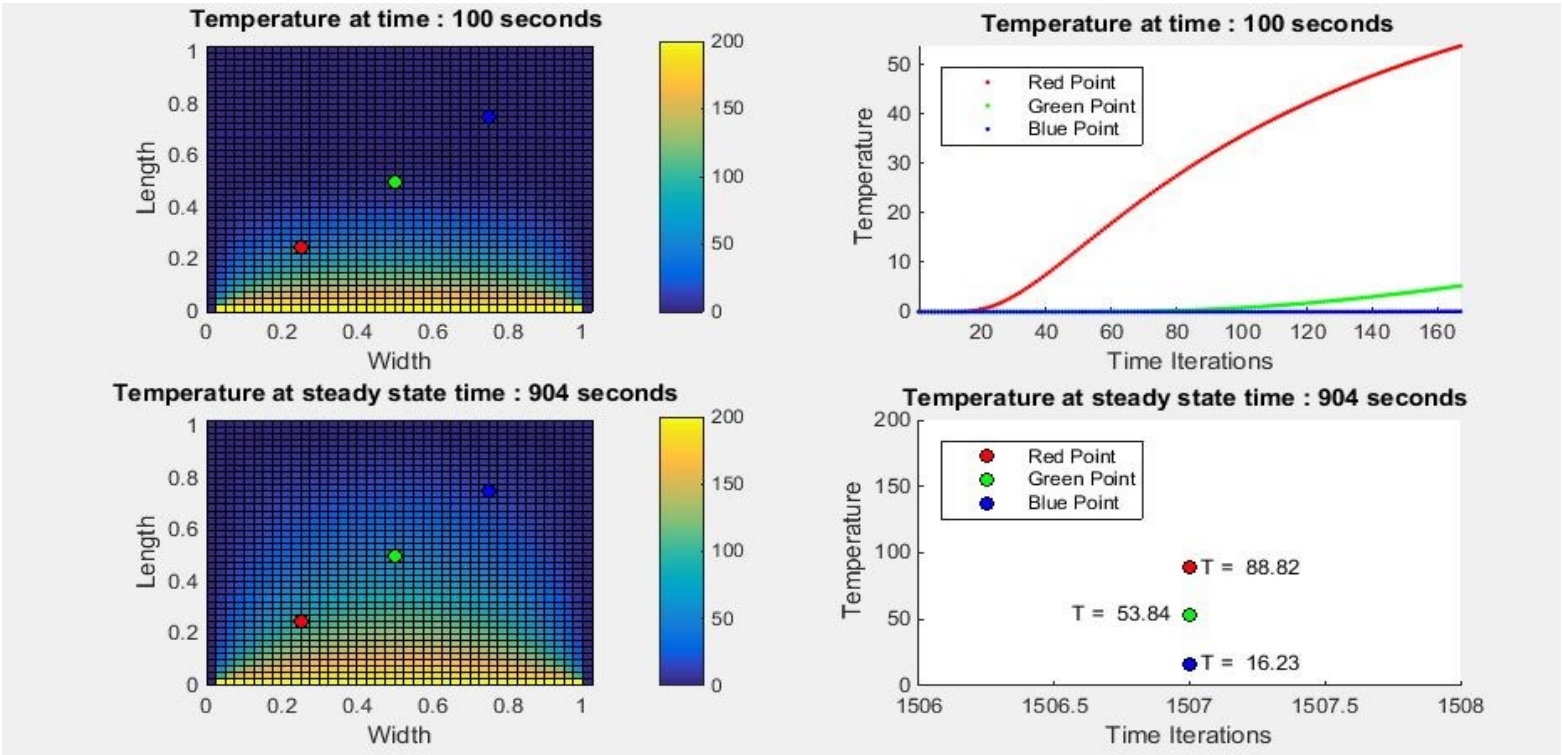

Comment:

In this trail the material was Copper and the boundary conditions as shown in the command window.

The diffusion of the heat was faster than the Aluminum due to the larger alpha

It took 2023 second to reach the Steady-State shown in the lower-left plot.

Silver




```
Command Window
  This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Silver
  The solution is  based on "Dirichlet Boundry Conditions" with initial values
  T(x,0,t)=200 , T(x,1,t)=0 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=0
  The plate takes 904 seconds to reach steady-state temperature with tolerence 0.50
fx Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

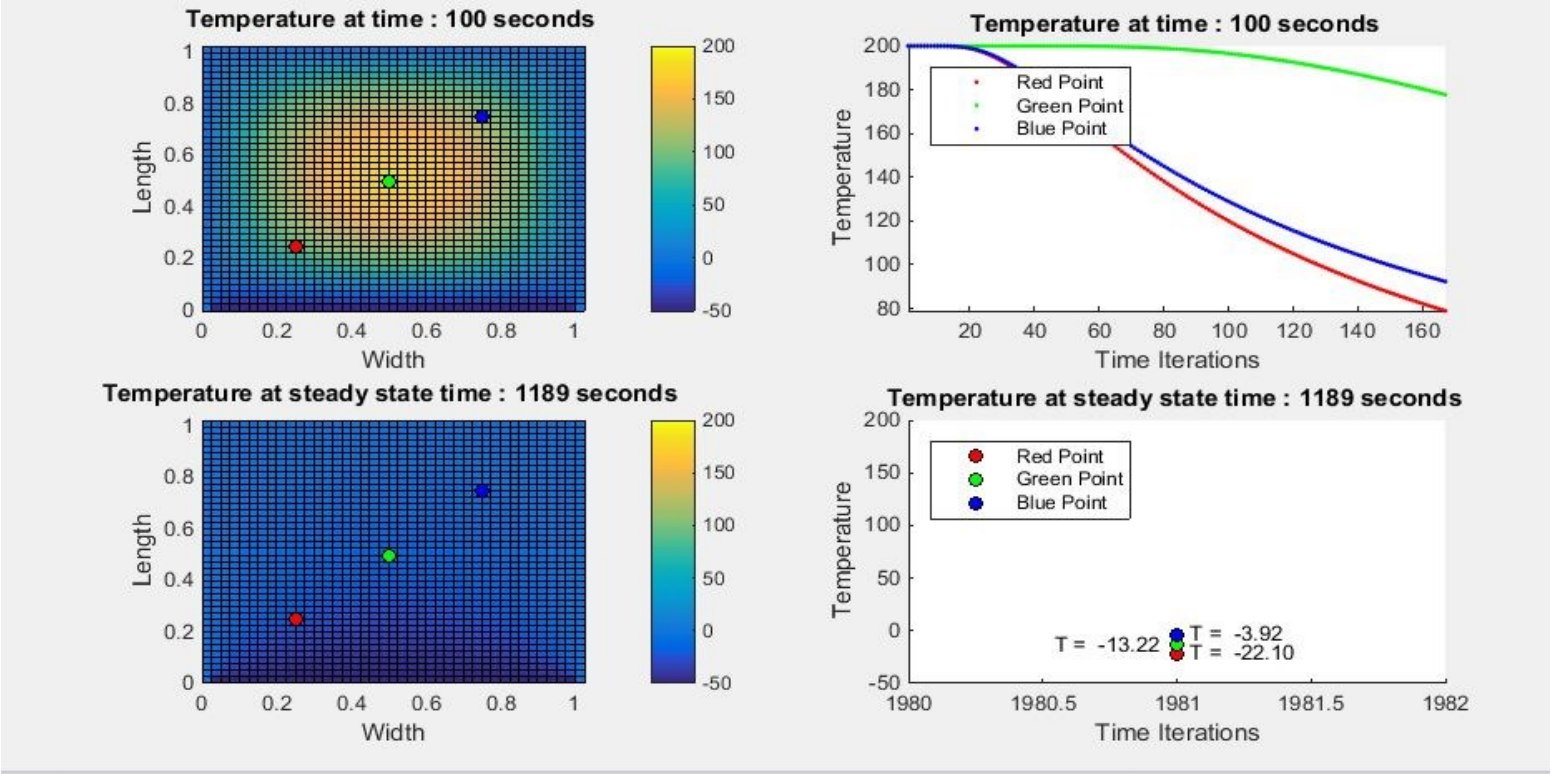
In this trail the material was Silver and the boundary conditions as shown in the command window.

The diffusion of the heat was the fastest due to the largest alpha.

It took just 904 second to reach the Steady-State which is less than both of Aluminum and Copper

2. Different boundary conditions with the same initial (200 deg C)

Trial A

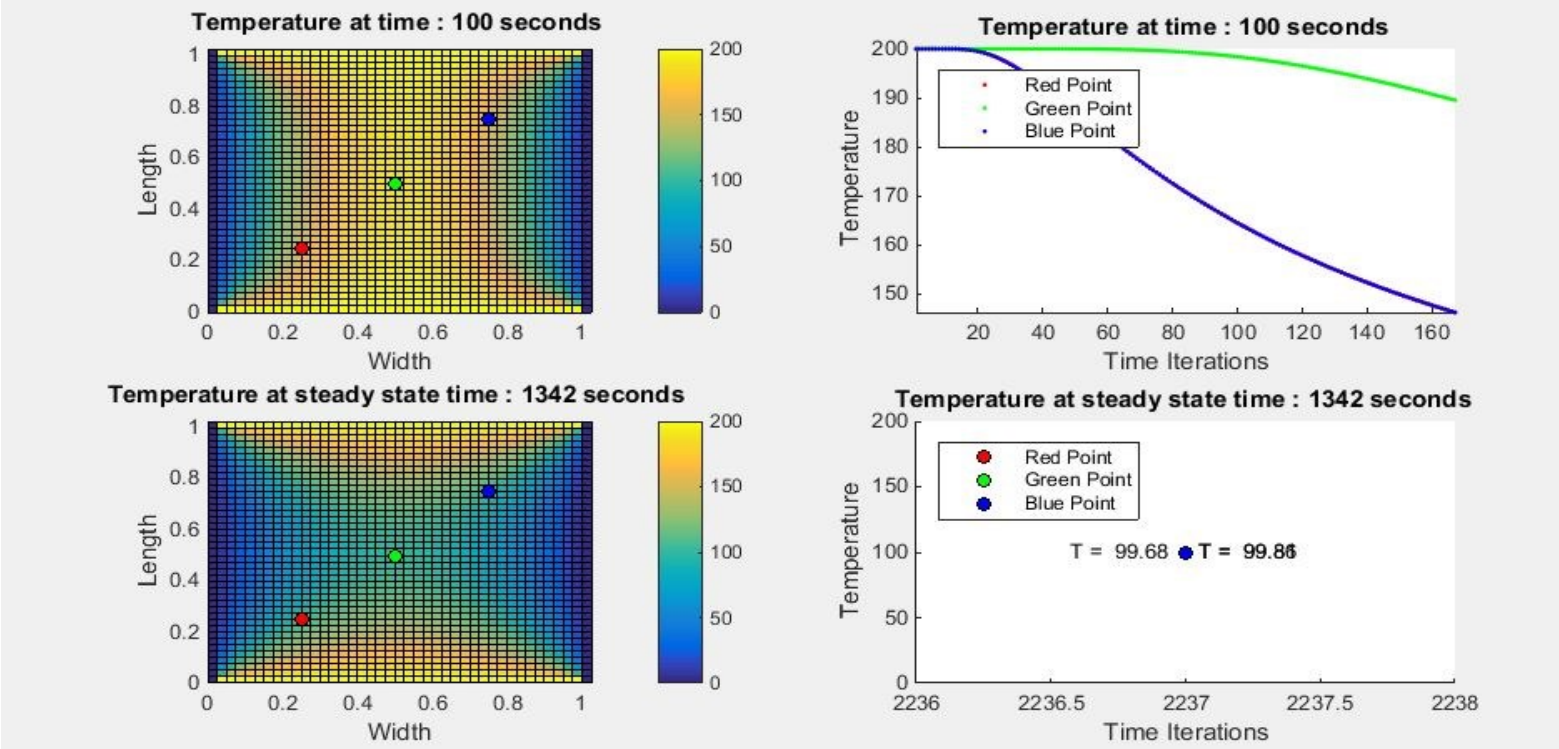



```
Command Window
  This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Silver
  The solution is  based on "Dirichlet Boundry Conditions" with initial values
  T(x,0,t)=-50 , T(x,1,t)=0 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=200
  The plate takes 1189 seconds to reach steady-state temperature with tolerence 0.50
fx Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

In this trail the boundary conditions was 0, 0, 0, -50 is shown in the command window.

Trial B



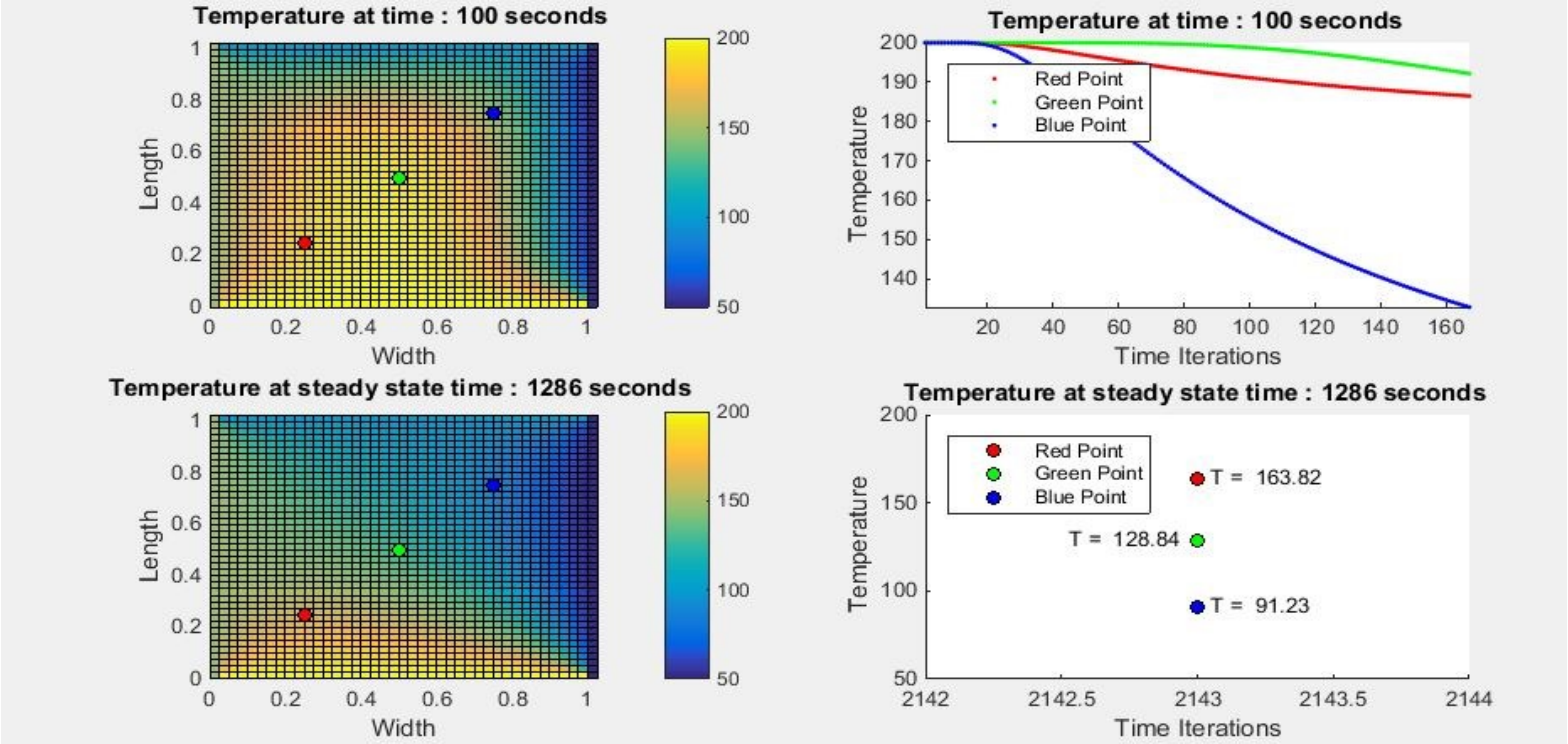

```
Command Window
  This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Silver
  The solution is  based on "Dirichlet Boundry Conditions" with initial values
  T(x,0,t)=200 , T(x,1,t)=200 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=200
  The plate takes 1342 seconds to reach steady-state temperature with tolerence 0.50
fx Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

In this trail the boundary conditions was 200, 0, 200, 0 is shown in the command window.

What was interesting here is that we have symmetry around the center point. The red and the blue points were in the same temperature all the time and finally the green point acts the same.

Trial C



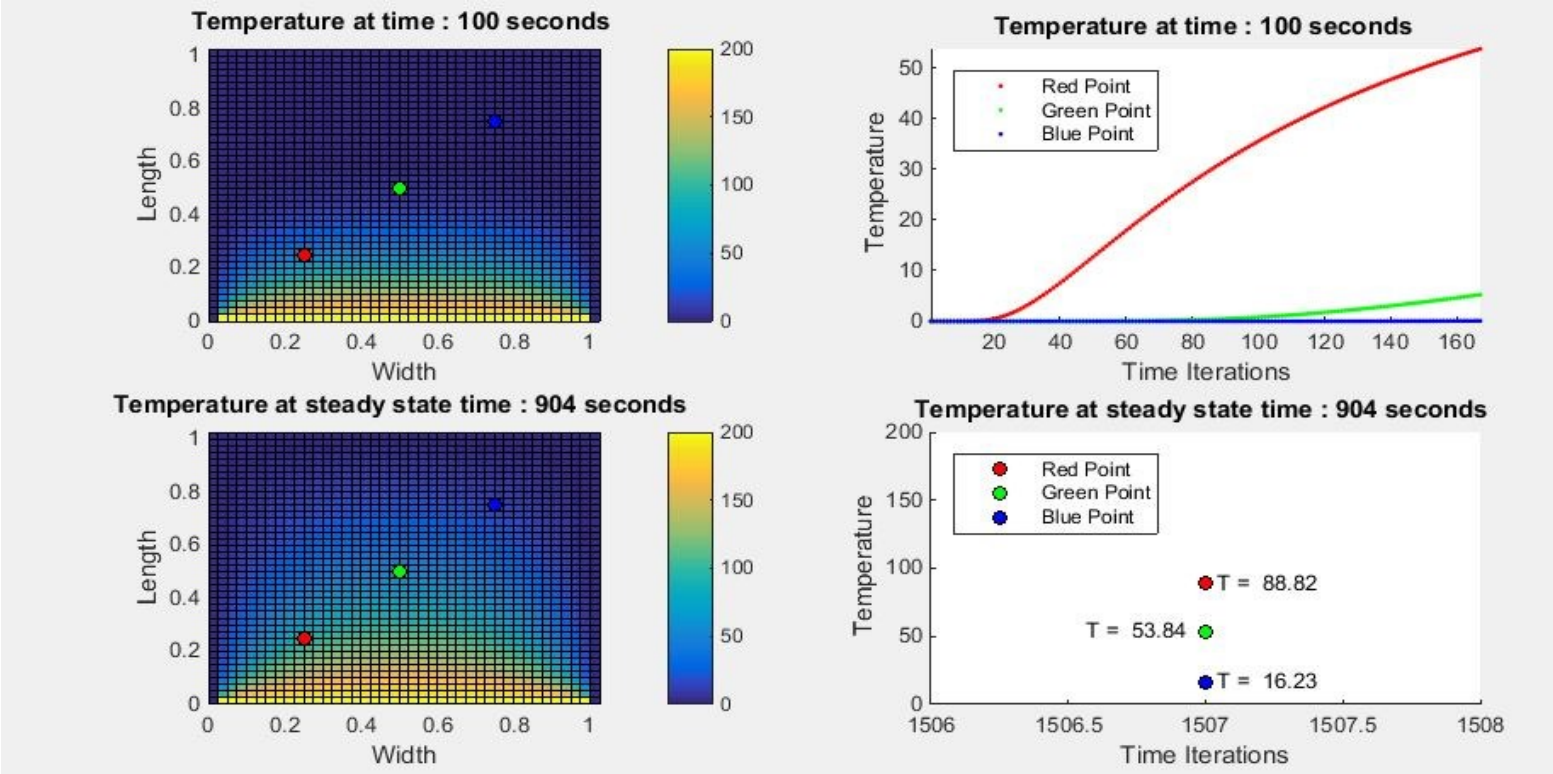

```
Command Window
This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Silver
The solution is  based on "Dirichlet Boundry Conditions" with initial values
T(x,0,t)=200 , T(x,1,t)=100 , T(0,y,t)=150 , T(1,y,t)=50 , T(x,y,0)=200
The plate takes 1286 seconds to reach steady-state temperature with tolerence 0.50
Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

In this trail the boundary conditions was 200, 150, 100 and 50 is shown in the command window.

3. Different initial condition with the same boundaries (200, 0, 0 & 0)

Trial A



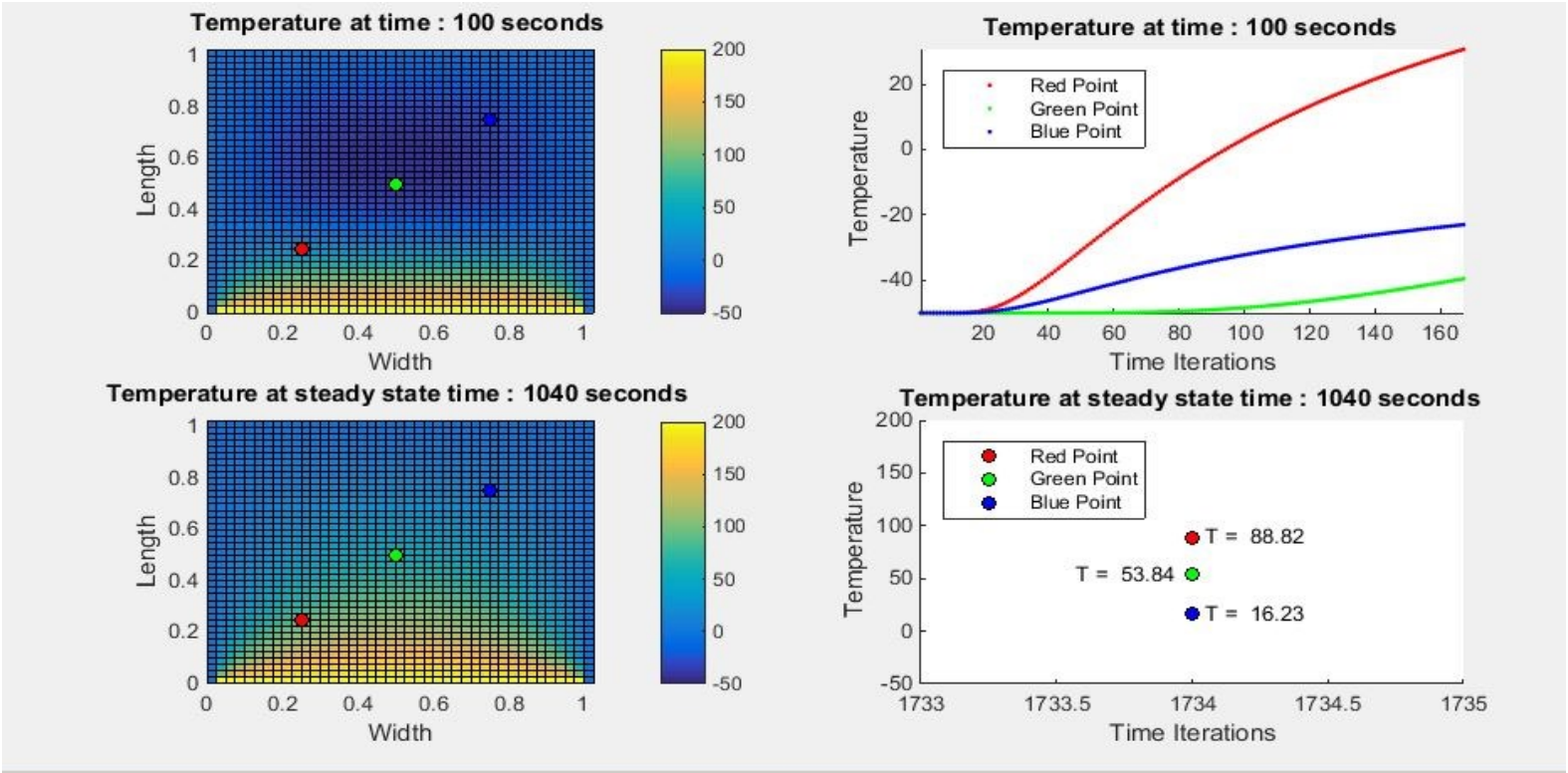

```
Command Window
The solution is  based on "Dirichlet Boundry Conditions" with initial values
T(x,0,t)=200 , T(x,1,t)=0 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=0
The plate takes 904 seconds to reach steady-state temperature with tolerence 0.50
Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

In this trail the initial condition was 0.

It took 904 seconds to reach Steady-State

Trial B



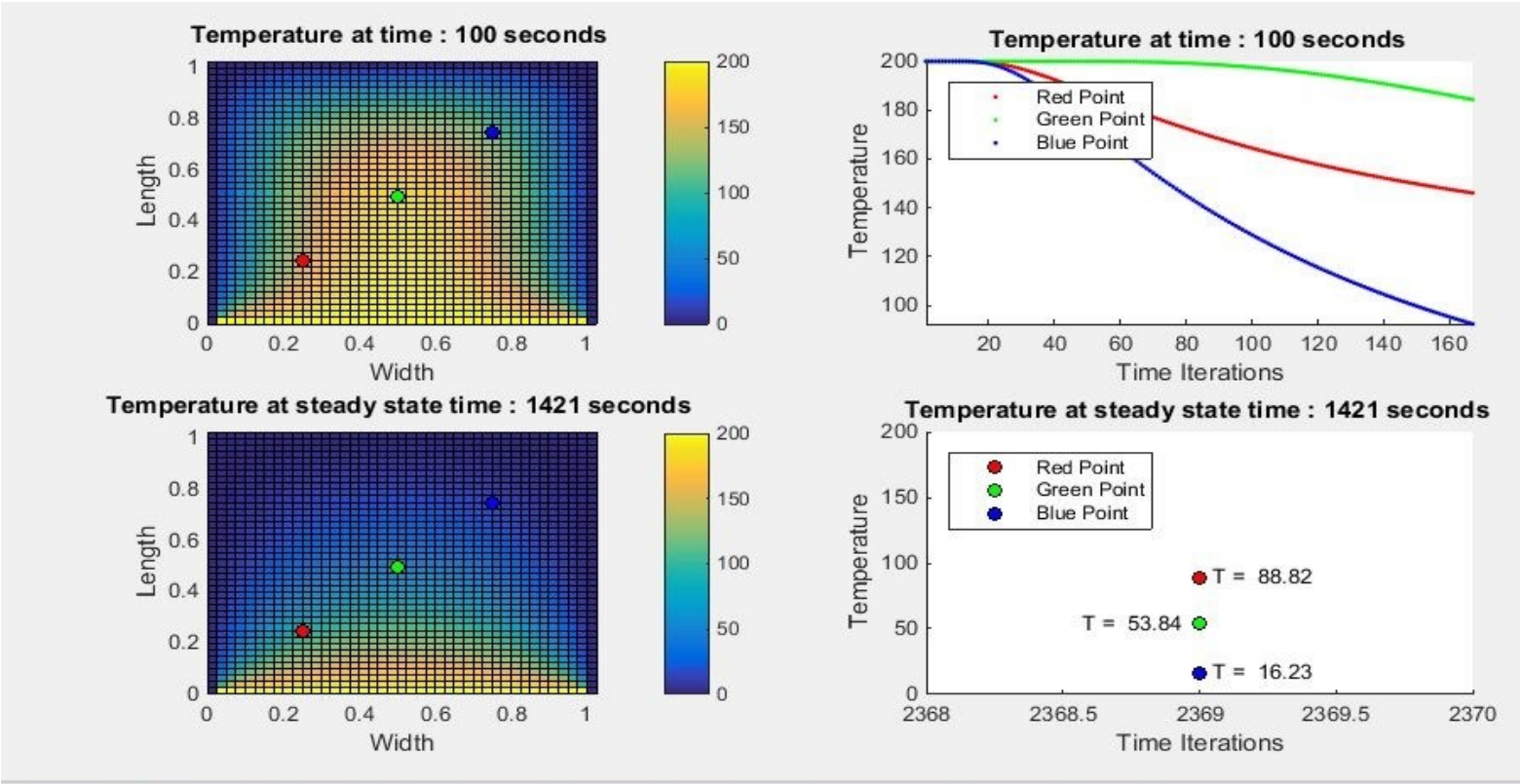

```
Command Window
  This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Silver
  The solution is  based on "Dirichlet Boundry Conditions" with initial values
  T(x,0,t)=200 , T(x,1,t)=0 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=-50
  The plate takes 1040 seconds to reach steady-state temperature with tolerence 0.50
fx Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

In this trail the initial condition was -50.

It took 1040 seconds to reach Steady-State (>904 in the trial A because of the larger difference between 200 & -50)

Trial C




```
Command Window
The solution is  based on "Dirichlet Boundry Conditions" with initial values
T(x,0,t)=200 , T(x,1,t)=0 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=200
The plate takes 1421 seconds to reach steady-state temperature with tolerence 5.0e-01
Now, Simulation is running with final time 100 seconds and step 0.60 second
```

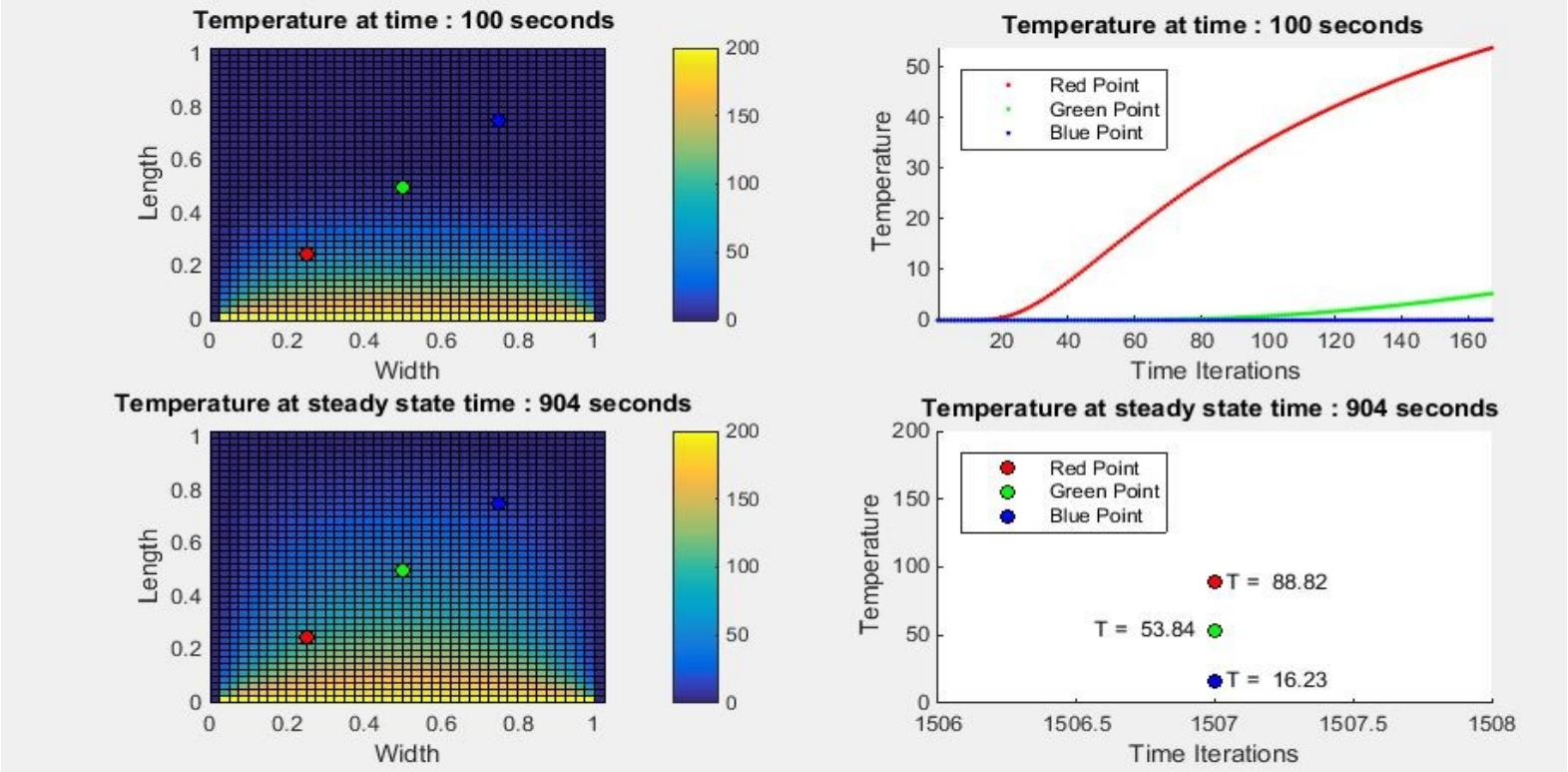

Comment:

In this trail the initial condition was 200.

It took 421 seconds to reach Steady-State.

4. Runge-Kutta and Euler in the same conditions

Trial A



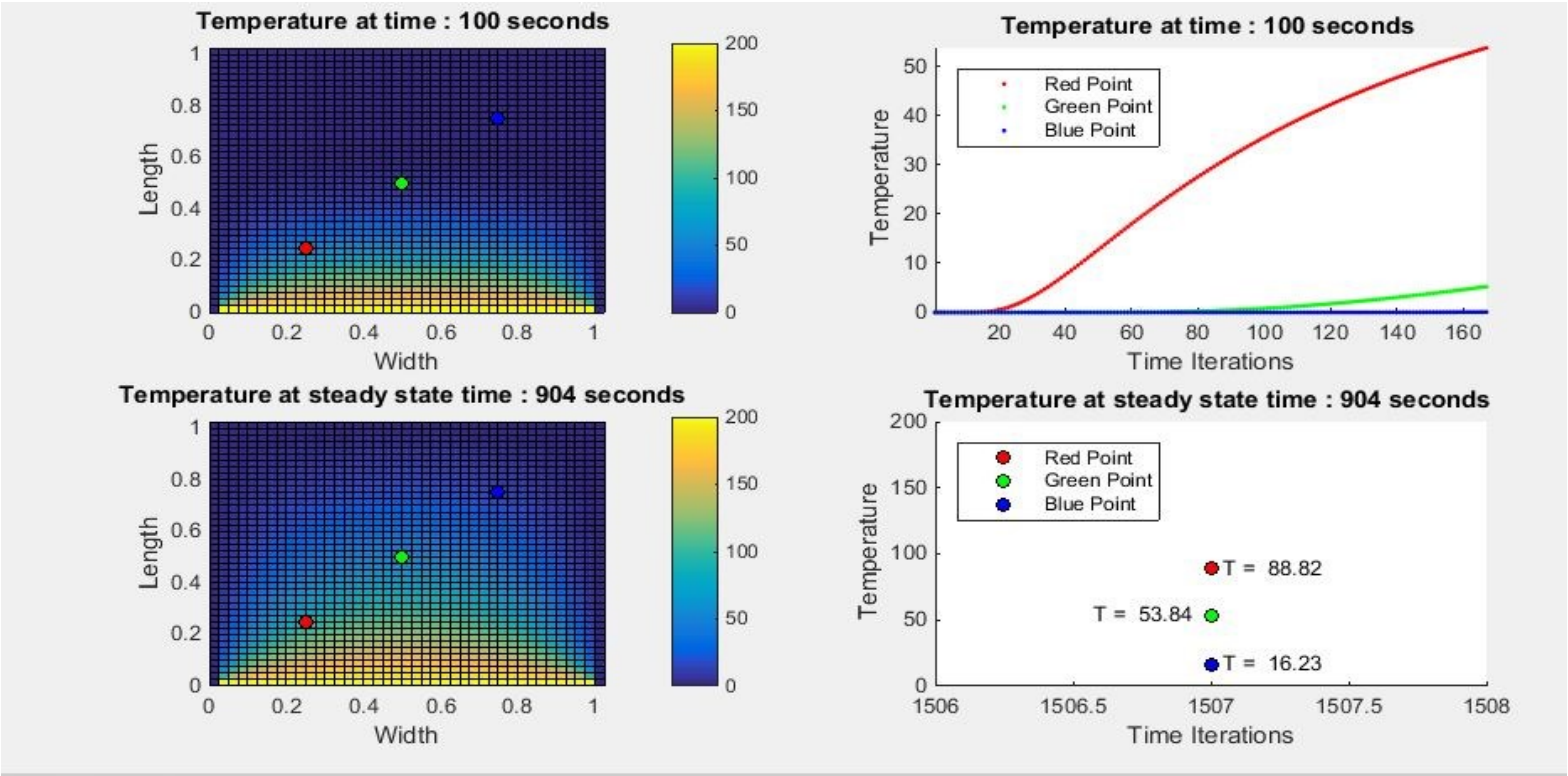

```
Command Window
The solution is  based on "Dirichlet Boundry Conditions" with initial values
T(x,0,t)=200 , T(x,1,t)=0 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=0
The plate takes 904 seconds to reach steady-state temperature with tolerence 0.50
Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

This trail was conducted using Euler method, the initial condition was 0 and the boundary conditions was 200, 0, 0, 0 as shown in the command window.

It took 904 seconds to reach Steady-State

Trial B




```
Command Window
  This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Silver
  The solution is  based on "Dirichlet Boundry Conditions" with initial values
  T(x,0,t)=200 , T(x,1,t)=0 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=0
  The plate takes 904 seconds to reach steady-state temperature with tolerence 0.50
fx Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

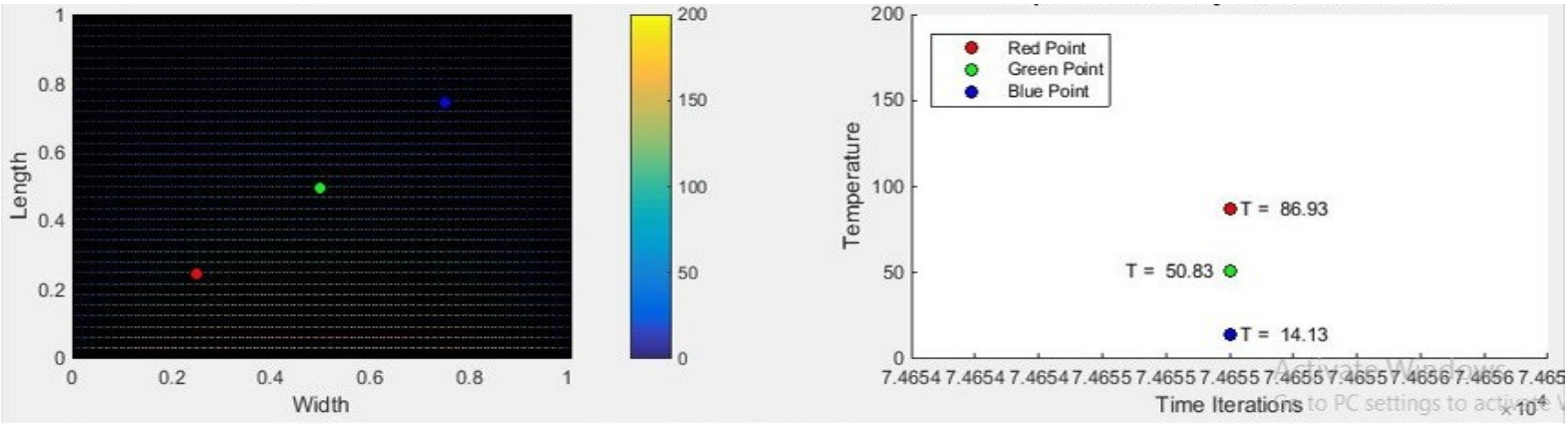
This trail was conducted using Runge-Kutta method, the initial condition and the boundary conditions was same as trial A

It took 904 seconds to reach Steady-State also same as the trial A.

But it took more time in running the code because of the complexity of Runge-Kutta method in comparison with Euler method.

5. Different number of nodes in x and y directions

Trial A




```
Lx= 1;                          % plate width (m)
Ly= 1;                          % plate length (m)
Nx=200;                          % nodes in x direction
Ny=200;                          % nodes in y direction

tolerence_ss=0.000001;             % tolerence for steady state section
```

Comment:

Because of the limited resources of the computer I have, just I ran the code to solve for Steady-State only and compare here.

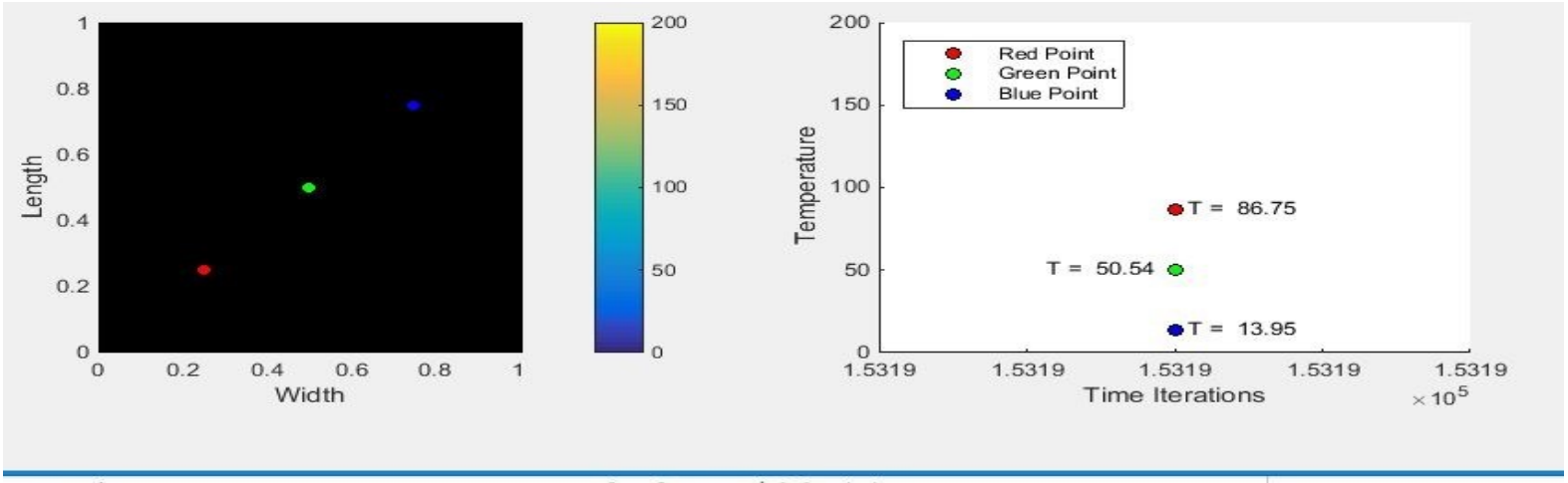
This trail was conducted using Euler method, the initial condition was 0 and the boundary conditions was 200, 0, 0, 0.

Number of nodes was 200 and It took about 15 minutes to solve

But the accuracy was improved which is clear in the temperature of the center point.

Its optimum value is 50 deg and here it's 50.83 deg with error 1.66% in comparison with 53.84 in the previous trials with 40 nodes only and error = 7.68%.

Trial B




```
Lx= 1;                                  % plate width (m)
Ly= 1;                                  % plate length (m)
Nx=300;                                  % nodes in x direction
Ny=300;                                  % nodes in y direction

tolerence_ss=0.000001;                     % tolerence for steady state section
```

Comment:

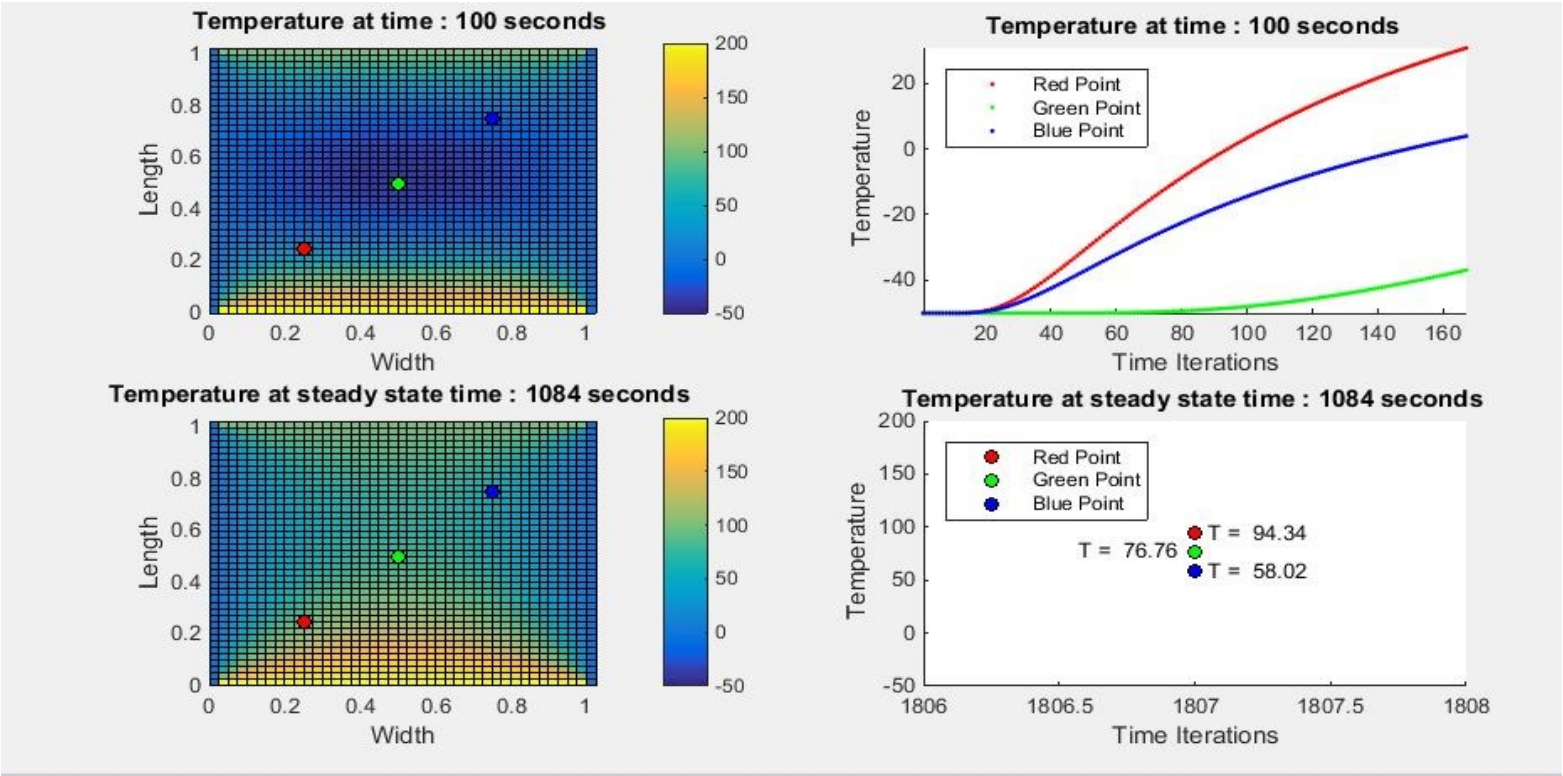
This was the highest accuracy I could reach.

Number of nodes was 300 and it took about 45 minutes to solve

The accuracy here was better than 200 nodes with error = 1.08%

6. More results with different conditions

Trial A



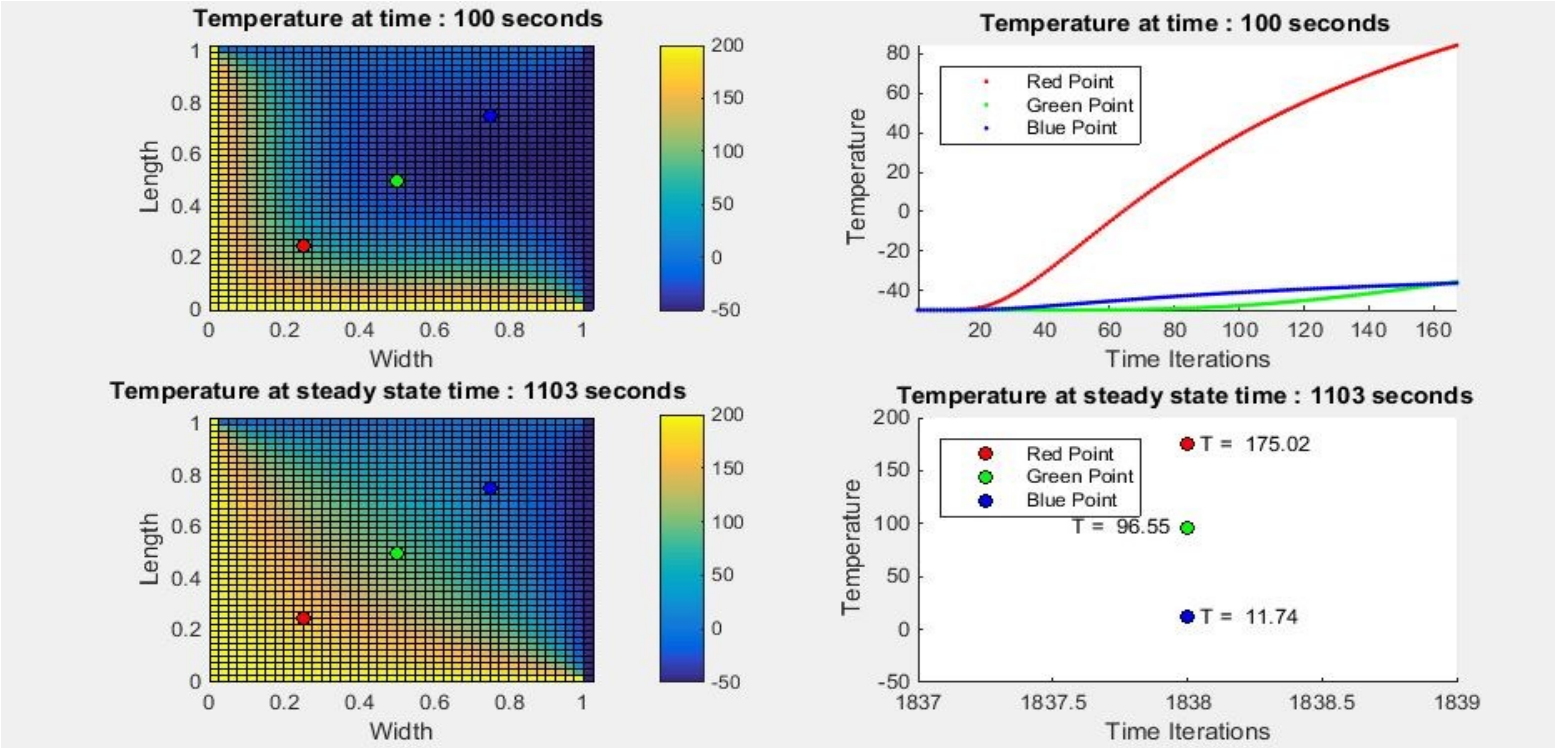

```
Command Window
  This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Silver
  The solution is  based on "Dirichlet Boundry Conditions" with initial values
  T(x,0,t)=200 , T(x,1,t)=100 , T(0,y,t)=0 , T(1,y,t)=0 , T(x,y,0)=-50
  The plate takes 1084 seconds to reach steady-state temperature with tolerence 0.50
fx Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

In this trail the boundary conditions was 200, 100, 0 and 0 and initial condition -50 as shown in the command window.

It took 1084 seconds to reach Steady-State.

Trial B



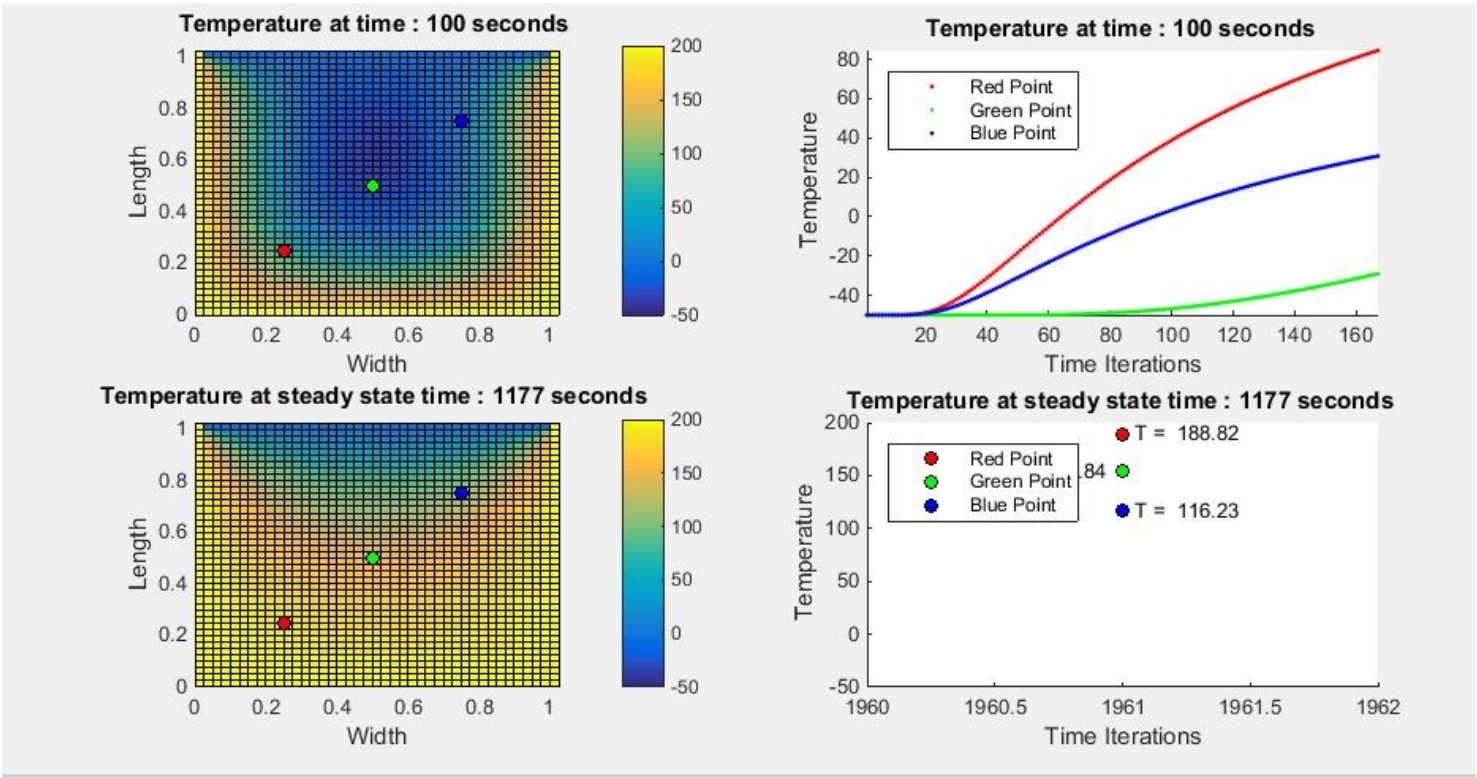

```
Command Window
  This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Silver
  The solution is  based on "Dirichlet Boundry Conditions" with initial values
  T(x,0,t)=200 , T(x,1,t)=0 , T(0,y,t)=200 , T(1,y,t)=-50 , T(x,y,0)=-50
  The plate takes 1103 seconds to reach steady-state temperature with tolerence 0.50
fx Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

In this trail the boundary conditions was 200, 0, 200 and -50 and initial condition -50 as shown in the command window.

It took 1103 seconds to reach Steady-State.

Trial C




```
Command Window
  This is the solution of the heat equation through out a plate of diamensions 1 X 1 of material Silver
  The solution is  based on "Dirichlet Boundry Conditions" with initial values
  T(x,0,t)=200 , T(x,1,t)=0 , T(0,y,t)=200 , T(1,y,t)=200 , T(x,y,0)=-50
  The plate takes 1177 seconds to reach steady-state temperature with tolerence 0.50
fx Now, Simulation is running with final time 100 seconds and step 0.60 second
```


Comment:

In this trail the boundary conditions was 200, 0, 200 and 200and initial condition -50 as shown in the command window.

It took 1177 seconds to reach Steady-State.

Conclusion:

Finite difference is a very powerful tool to solve ordinary deferential equations and partial deferential equations. It can be used to solve many initial value problems like our problem here and many others such as wave equation and Laplace equation.

In this project, I benefited a lot about solving complex problems and simulating it and generally programing in MATLAB. It was a great experience in planning for a code, designing, debugging and fixing its errors.

References

1. “FINITE-DIFFERENCE SOLUTION TO THE 2-D HEAT EQUATION slides 1-14”. Web. 27 Dec. 2015.

2. “MATLAB code for solving Laplace's equation using the Jacobi method”. YouTube Video. 20 Dec. 2015.

3. “Stability of the Implicit Solution of 1D Heat Equation” http://www.cs.berkeley.edu/~demmel/cs267/lecture17/lecture17.html . Web. 22 Dec. 2015.

Appendix I : " The Code "

```
% This code is designed to solve the heat equation in a 2D plate.
% Using fixed boundary conditions "Dirichlet Conditions" and initial
% temperature in all nodes, It can solve until reach steady state with
% tolerance value selected below.
% After solution, graphical simulation appears to show you how the heat
% diffuses throughout the plate within time interval selected below

clear; close all; clc;


 %------ 1- Inputs section -------

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 % Please select your material, enter your parameters and your initial
   conditions %
 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % %    -- Aluminum --    % ** Uncomment if needed **
 % name=('Aluminium');    % Material name
 % conductivity = 204.3;  % thermal conductivity (j/m.C.sec)
 % spacific_heat = 910;   % specific heat (j/kg.C)
 % denisty = 2700.0;  % density (kg/m^3)

 % %   -- Copper --   % ** Uncomment if needed **
 % name=('Copper');   % Material name
 % conductivity = 401;    % thermal conductivity (W/m.K)
 % spacific_heat = 390;   % specific heat (J/kg K)
 % denisty = 8940;    % density (kg/m^3)

 %   -- Silver --
 name=('Silver');     % Material name
 conductivity = 629;  % thermal conductivity (W/m.K)
 spacific_heat = 233; % specific heat (J/kg K)
 denisty = 10490;     % density (kg/m^3)

 % %  -- Custom Material --  % ** Uncomment and enter your values if
                                needed **
 % name=('Custom Material');  % Material name
 % conductivity =    ;    % thermal conductivity (W/m.K)
 % spacific_heat =    ;   % specific heat (J/kg K)
 % denisty =  ;   % density (kg/m^3)

 %--------------------------------------

 Lx= 1;          % plate width (m)
```

```
44.  Ly= 1;          % plate length (m)
45.  Nx=40;               % nodes in x direction
46.  Ny=40;               % nodes in y direction
47.
48.  T_initial= 0 ;       % Initial temperature in all nodes ( the whole
   plate )
49.  T_east   = 0 ;       % temperature on the upper side ( at y=0
   "Dirichlet Conditions" )
50.  T_west   = 0 ;       % temperature on the lower side ( at y=Ly
   "Dirichlet Conditions" )
51.  T_north  = 0 ;       % temperature on the left  side ( at x=0
   "Dirichlet Conditions" )
52.  T_south  = 200 ;       % temperature on the right side ( at x=Lx
   "Dirichlet Conditions" )
53.
54.  t_end=100 ;        % final time for visual simulation (sec)
55.  dt=0.7 ;       % time step (1 sec)
56.
57.  tolerence = 0.5;       % tolerence for numerical simulation (0.5 deg
   Celesius)
58.  tolerence_ss=0.001;    % tolerence for steady state section (0.1 deg
   Celesius)
59.
60.  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61.  %%%%%% From here, You don't need to modify %%%%%%%%
62.  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63.
64.
65.  % ---------------------------------------
66.  % -------- 2- Constants Section -------
67.  % ---------------------------------------
68.
69.
70.  k=1;               % iteration counter
71.  err_SS_max(k)=1;              % initial error
72.  err_SS_min(k)=1;              % initial error
73.  dx=Lx/Nx;              % delta x
74.  dy=Ly/Ny;              % delta y
75.  n_time=round(t_end/dt);               % number of iterrations for time
76.  alpha = conductivity/(spacific_heat*denisty);      % alpha (1/sec)
77.  T_max=max([T_east T_west T_north T_south T_initial]);      % Max T to
   set axes limits in plotting
78.  T_min=min([T_east T_west T_north T_south T_initial]);      % Min T to
   set axes limits in plotting
79.  Solution_type=questdlg('Which method you want to solve the time
   derivative with ?','Question','Euler','2nd order Runge-Kutte','Euler');
   % solve with 2nd order Runge Kutte in time or 2 to solve with Euler
80.
```

```
81.  if dt<= 1/(2*alpha*((1/dx^2)+(1/dy^2)))         % test the stability
   condition
82.  else
83.       fprintf('Error, the stability condition is not met\nPlease return
   to "Inputs Section" and choose a "dt" smaller than %f
   \n',1/(2*alpha*((1/dx^2)+(1/dy^2))))
84.       return
85.  end
86.  message=msgbox('Your computer is now solving the problem, Please
   wait..... ');    % Busy message
87.  % ----- Initial Conditions for finite difference section ---
88.  T=zeros(Nx+2,Ny+2,75000);       % set max iterations 75,000 due to
   memory limitations (T variable takes maximum 1GB in memory)
89.  T(:,1,:)=T_south;
90.  T(:,Ny+1,:)=T_north;
91.  T(:,Ny+2,:)=T_north;        % Redundant, it has no effect in
   calculations but is required in plotting section
92.  T(Nx+1,:,:)=T_east;
93.  T(Nx+2,:,:)=T_east;         % Redundant, it has no effect in
   calculations but is required in plotting section
94.  T(1,:,:)=T_west;
95.  T(:,:,1)=T_initial;
96.  % ------- Initial Conditions for steady state section -------
97.  Tss=zeros(Nx+2,Ny+2);    Tss2=zeros(Nx+2,Ny+2);
98.  Tss(:,1)=T_south;    Tss2(:,1)=T_south;
99.  Tss(:,Ny+1)=T_north;     Tss2(:,Ny+1)=T_north;
100. Tss(:,Ny+2)=T_north;     Tss2(:,Ny+2)=T_north;     % Redundant, it has
   no effect in calculations but is required in plotting section
101. Tss(Nx+1,:)=T_east;  Tss2(Nx+1,:)=T_east;
102. Tss(Nx+2,:)=T_east;  Tss2(Nx+2,:)=T_east;      % Redundant, it has no
   effect in calculations but is required in plotting section
103. Tss(1,:)=T_west;     Tss2(1,:)=T_west;
104.
105.
106. %----------------------------------------
107. %--------- 3- Steady-State section -----
108. %----------------------------------------
109.
110.
111.    while err_SS_max(k)>=tolerence_ss || err_SS_min(k)>=tolerence_ss
112.
113.     for i=2:Nx            % looping
114.     for j=2:Ny
115.     Tss2(i,j)=0.25*(Tss(i+1,j)+Tss(i,j+1)+Tss(i-1,j)+Tss(i,j-1));
116.     end
117.     end
118.     k=k+1;            % update k
119.     err_SS_max(k)=abs(max(max(Tss2-Tss)));        % calculate error
```

```
120.     err_SS_min(k)=abs(min(min(Tss2-Tss)));        % calculate error
121.     Tss=Tss2;             % update T
122.     end
123.
124.
125. %-------------------------------------------------------------------
126. %----- 4- Finite difference section (Using 2nd order Runge Kutte or
                                                        Euler in time) -----
127. %-------------------------------------------------------------------
128.
129. k=1;
130. switch Solution_type
131.     case '2nd order Runge-Kutte'
132.     err_R_k_max(k)=100;        % initial error
133.     err_R_k_min(k)=100;        % initial error
134.     while err_R_k_max(k)>=tolerence || err_R_k_min(k)>=tolerence
135.   for i=2:Nx
136.     for j=2:Ny
137.     k1=alpha*(((T(i-1,j,k)-2*T(i,j,k)+T(i+1,j,k))/dx^2)+((T(i,j-1,k)-
   2*T(i,j,k)+T(i,j+1,k))/dy^2));
138.     Tk=T(:,:,k)+k1*dt;
139.     k2=alpha*(((Tk(i-1,j)-2*Tk(i,j)+Tk(i+1,j))/dx^2)+((Tk(i,j-1)-
   2*Tk(i,j)+Tk(i,j+1))/dy^2));
140.     T(i,j,k+1) =T(i,j,k)+(dt/2)*(k1+k2);
141.     end
142.   end
143.   k=k+1;
144.   err_R_k_max(k)=abs(max(max(T(:,:,k)-Tss)));    %calculate error
145.   err_R_k_min(k)=abs(min(min(T(:,:,k)-Tss)));    %calculate error
146.   if round(err_R_k_max(k),5)==round(err_R_k_max(k-1),5) &&
   err_R_k_max(k)~= 0  % Test solution convergence
147.    errordlg('The solution is not converging, Please choose a larger
   tolerence','Tolerence Error');
148.    close(message)
149.    return
150.   end
151.   if round(err_R_k_min(k),5)==round(err_R_k_min(k-1),5) &&
   err_R_k_min(k)~= 0  % Test solution convergence
152.    errordlg('The solution is not converging, Please choose a larger
   tolerence','Tolerence Error');
153.    close(message)
154.    return
155.   end
156.  end
157.
158.     case'Euler'
159.     err_E_max(k)=100;        % initial error
160.     err_E_min(k)=100;        % initial error
```

```
161.      while err_E_max(k)>=tolerence || err_E_min(k)>=tolerence
162.    for i=2:Nx
163.      for j=2:Ny
164.      T(i,j,k+1) =T(i,j,k)+dt*alpha*(((T(i-1,j,k)-
   2*T(i,j,k)+T(i+1,j,k))/dx^2)+((T(i,j-1,k)-2*T(i,j,k)+T(i,j+1,k))/dy^2));
165.      end
166.    end
167.    k=k+1;
168.    err_E_max(k)=abs(max(max(T(:,:,k)-Tss)));    %calculate error
169.    err_E_min(k)=abs(min(min(T(:,:,k)-Tss)));    %calculate error
170.    if round(err_E_max(k),5)==round(err_E_max(k-1),5) && err_E_max(k)~= 0
   % Test solution convergence
171.     errordlg('The solution is not converging, Please choose a larger
   tolerence','Tolerence Error');
172.     close(message)
173.     return
174.    end
175.    if round(err_E_min(k),5)==round(err_E_min(k-1),5) && err_E_min(k)~= 0
   % Test solution convergence
176.     errordlg('The solution is not converging, Please choose a larger
   tolerence','Tolerence Error');
177.     close(message)
178.     return
179.    end
180.   end
181.
182.      case []
183.      close(message)
184.      msgbox('Error, Please re-run the code and choose Euler or 2nd order
   Runge-Kutte to continue the solution')
185.      return
186. end
187. T=T(:,:,1:k);              % delete the unused assigned zero layers
188. SStime=k*dt;           % steady state time
189. close(message)             % close the busy message
190.
191. %-----------------------------------------------
192. %--------- 5- Printed results section --------
193. %-----------------------------------------------
194.
195. fprintf('This is the solution of the heat equation through out a plate
   of diamensions %i X %i of material %s \n',Lx,Ly,name);
196. fprintf('The solution is  based on "Dirichlet Boundry Conditions" with
   initial values \n')
197. fprintf('T(x,0,t)=%i , T(x,%i,t)=%i , T(0,y,t)=%i , T(%i,y,t)=%i ,
   T(x,y,0)=%i \n',T_south,Ly,T_north,T_west,Lx,T_east,T_initial)
198. fprintf('The plate takes %i seconds to reach steady-state temperature
   with tolerence %0.2f \n',round(SStime),tolerence);
```

```
199. fprintf('Now, Simulation is running with final time %i seconds and step
   %0.2f second \n',t_end,dt)
200.
201.
202. %---------------------------------
203. %------ 6- Plotting section ------
204. %---------------------------------
205.
206. x=zeros(1,Nx+2);y=zeros(1,Ny+2);    %Generate the plate
207. for i = 1:Nx+2
208. x(i) =(i-1)*dx;
209. end
210. for i = 1:Ny+2
211. y(i) =(i-1)*dy;
212. end
213.
214. % %    -- Constant plot ----
215.
216. subplot(2,2,3)
217. hold on
218. title(sprintf('Temperature at steady state time : %i seconds
   ',round(SStime)))
219. surf(x,y,Tss)
220. plot3(  Lx/4,  Ly/4,T_max,'ko','markerfacecolor','r') % plot red point
221. plot3(  Lx/2,  Ly/2,T_max,'ko','markerfacecolor','g') % plot green
   point
222. plot3(3*Lx/4,3*Ly/4,T_max,'ko','markerfacecolor','b') % plot blue point
223. plot3(  Lx/4,  Ly/4,T_min,'ko','markerfacecolor','r') % plot red point
224. plot3(  Lx/2,  Ly/2,T_min,'ko','markerfacecolor','g') % plot green
   point
225. plot3(3*Lx/4,3*Ly/4,T_min,'ko','markerfacecolor','b') % plot blue point
226. cb=colorbar;
227. caxis([T_min T_max]);
228. view(90,-90);
229. xlim([0 Lx+dx]); xlabel('Length');
230. ylim([0 Ly+dy]); ylabel('Width');
231. zlim([T_min T_max]); zlabel('Temprature');
232. drawnow
233. hold off
234.
235.  subplot(2,2,4)
236. hold on
237. title(sprintf('Temperature at steady state time : %i seconds
   ',round(SStime)))
238. scatter(k,Tss(floor(Nx/4),floor(Ny/4)),'ko','markerfacecolor','r');
239. val=(sprintf('  T =  %0.2f   ',Tss(floor(Nx/4),floor(Ny/4))));
240. text(k,Tss(floor(Nx/4),floor(Ny/4)),val,'HorizontalAlignment','Left');
241. scatter(k,Tss(floor(Nx/2),floor(Ny/2)),'ko','markerfacecolor','g');
```

```
val=(sprintf('  T =  %0.2f   ',Tss(floor(Nx/2),floor(Ny/2))));
text(k,Tss(floor(Nx/2),floor(Ny/2)),val,'HorizontalAlignment','right');
scatter(k,Tss(floor(3*Nx/4),floor(3*Ny/4)),'ko','markerfacecolor','b');
val=(sprintf('  T =  %0.2f   ',Tss(floor(3*Nx/4),floor(3*Ny/4))));
text(k,Tss(floor(3*Nx/4),floor(3*Ny/4)),val,'HorizontalAlignment','Left
');
axis tight; xlabel('Time Iterations');
ylim([T_min T_max]); ylabel('Temperature');
legend('Red Point','Green Point ','Blue Point ','Location','northwest')
drawnow
hold off

%     ------ Animated plot ----

for j=1:n_time
subplot(2,2,1)
surf(x,y,T(:,:,j))
hold on
title(sprintf('Temperature at time : %i seconds ',round(j*dt)))
plot3(  Lx/4,  Ly/4,T_max,'ko','markerfacecolor','r') % plot red point
plot3(  Lx/2,  Ly/2,T_max,'ko','markerfacecolor','g') % plot green point
plot3(3*Lx/4,3*Ly/4,T_max,'ko','markerfacecolor','b') % plot blue point
plot3(  Lx/4,  Ly/4,T_min,'ko','markerfacecolor','r') % plot red point
plot3(  Lx/2,  Ly/2,T_min,'ko','markerfacecolor','g') % plot green point
plot3(3*Lx/4,3*Ly/4,T_min,'ko','markerfacecolor','b') % plot blue point
cb=colorbar;
caxis([T_min T_max]);
view(90,-90);
xlim([0 Lx+dx]); xlabel('Length');
ylim([0 Ly+dy]); ylabel('Width');
zlim([T_min T_max]); zlabel('Temprature');
drawnow
hold off

subplot(2,2,2)
hold on
title(sprintf('Temperature at time : %i seconds ',round(j*dt)))
scatter(j,T(floor((Nx+2)/4),floor((Ny+2)/4),j),'r.');
scatter(j,T(ceil((Nx+2)/2),ceil((Ny+2)/2),j),'g.');
scatter(j,T(ceil(3*(Nx+2)/4),ceil(3*(Ny+2)/4),j),'b.');
axis tight; xlabel('Time Iterations');
axis tight; ylabel('Temperature');
legend('Red Point','Green Point ','Blue Point ','Location','northwest')
drawnow
hold off

end
```