Advanced Computational Fluid Dynamics (AM6513)

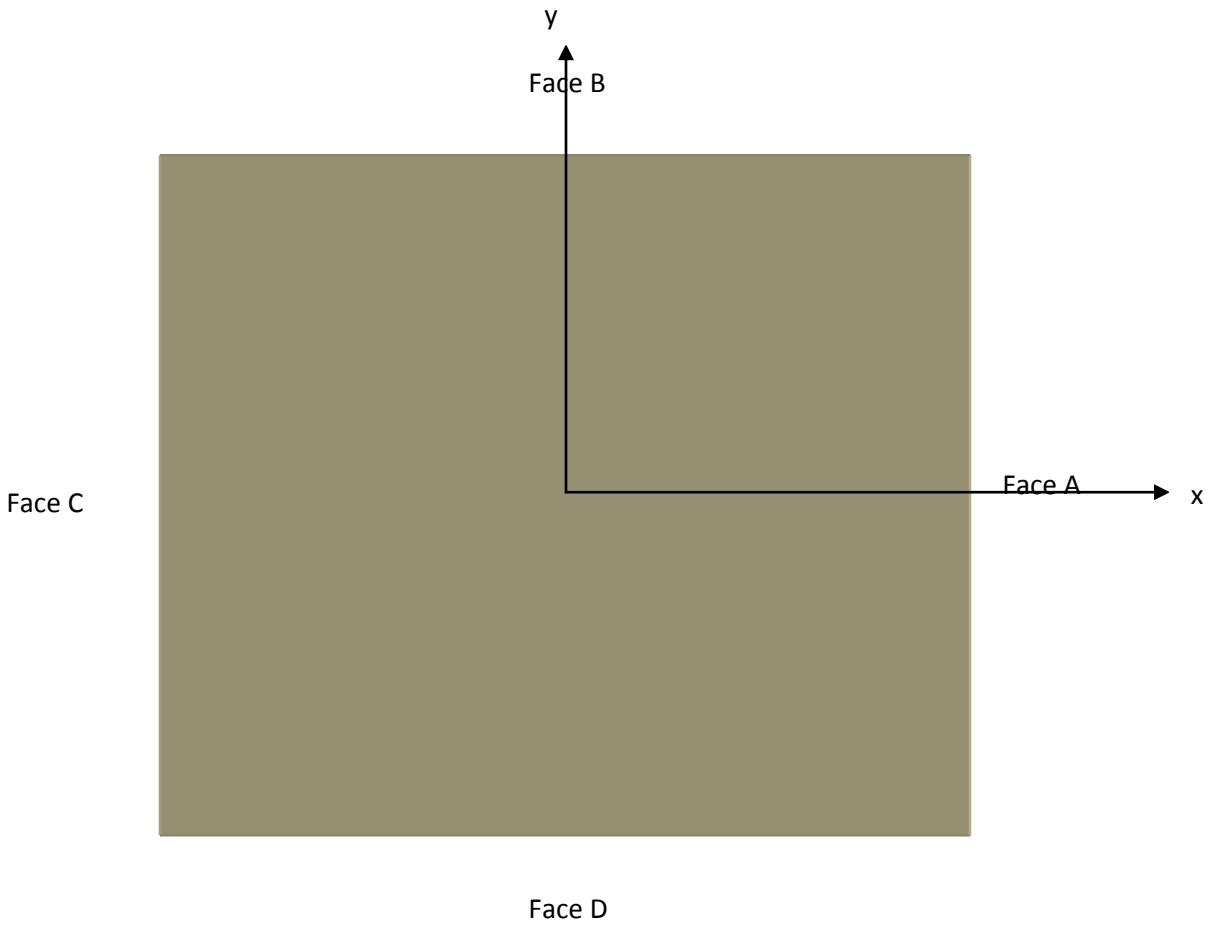
Report on Assignment 2

Kumar Saurabh (MA14M004)

Problem Description:

To determine the temperature distribution for various values of Biot Number

- Bi = 0.01
- Bi = 0.1
- Bi = 1.0
- Bi = 10.0
- Bi =100



Governing Equation:

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

Under Steady State:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = -\frac{\emptyset}{k}$$

Converting to the non dimension form:

$$x^* = \frac{x}{L}$$
$$y^* = \frac{y}{L}$$
$$\theta = \frac{T - T_f}{\frac{\emptyset L^2}{k}}$$

$$\boldsymbol{\frac{\partial^2 \theta}{\partial x^{*2}} + \frac{\partial^2 \theta}{\partial y^{*2}} = -1}$$

Boundary condition:

$$Bi = \frac{h * L}{k}$$

Face A:
$$-k\frac{\partial T}{\partial x} = h(T - T_f)$$
$$\boldsymbol{\frac{\partial \theta}{\partial x^*} + Bi\theta = 0}$$

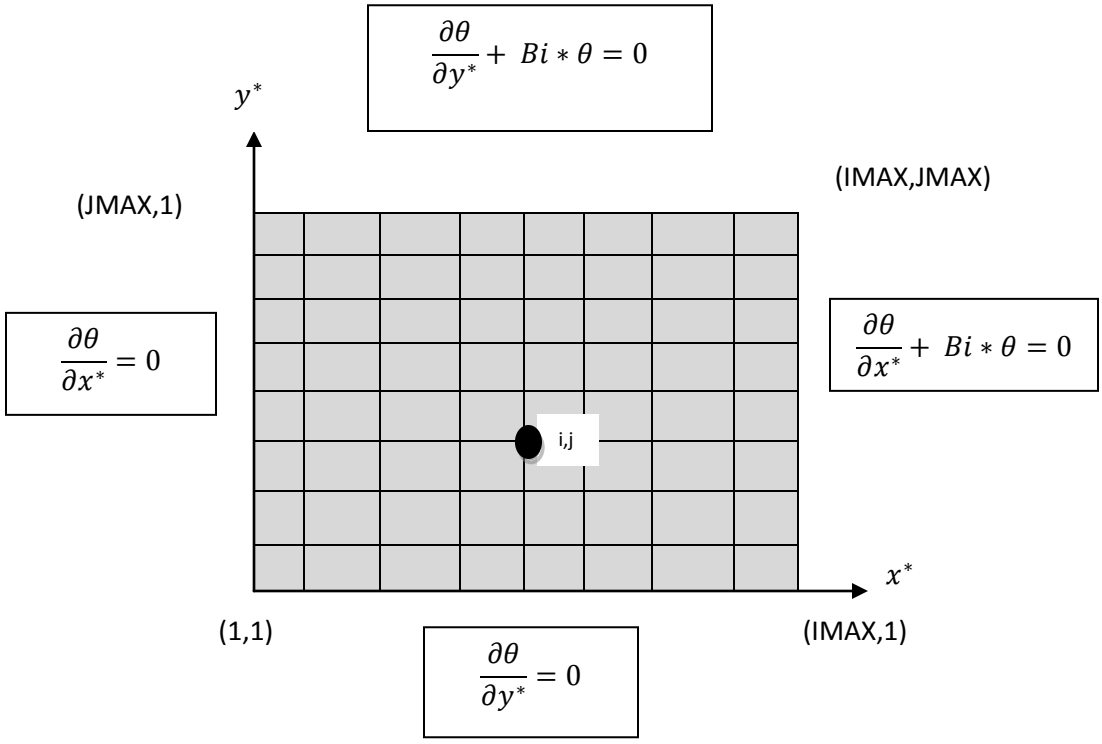
Face B:
$$-k\frac{\partial T}{\partial y} = h(T - T_f)$$
$$\boldsymbol{\frac{\partial \theta}{\partial y^*} + Bi\theta = 0}$$

Face C:
$$k\frac{\partial T}{\partial x} = h(T - T_f)$$
$$\boldsymbol{-\frac{\partial \theta}{\partial x^*} + Bi\theta = 0}$$

Face D:
$$k\frac{\partial T}{\partial x} = h(T - T_f)$$
$$\boldsymbol{-\frac{\partial \theta}{\partial x^*} + Bi\theta = 0}$$

Numerical Formulation:

Since there is symmetry, so we solve only for the $(1/4)^{th}$ of the slab. Now the boundary condition changes for the symmetrical side. The resultant Boundary conditions are shown in the figure:



Assumption:

Number of grid points along x^* and y^* are same i.e.

$$\Delta x = \Delta y$$

At any interior node (i,j):

$$\frac{\theta_{i+1,j} - 2 * \theta_{i,j} + \theta_{i-1,j}}{\Delta x^2} + \frac{\theta_{i,j-1} - 2 * \theta_{i,j} + \theta_{i,j+1}}{\Delta y^2} = -1$$

$$\boldsymbol{-4\theta_{i,j} + \theta_{i+1,j} + \theta_{i-1,j} + \theta_{i,j-1} + \theta_{i,j+1} = -\Delta x^2}$$

Along the line y = 1 (except at the corner points):

- j= 1
- Discretizing the boundary condition by CDS:

$$\frac{\partial \theta}{\partial y^*} = 0$$
$$\frac{\theta_{i,0} - \theta_{i,2}}{2 * \Delta y} = 0$$
$$\theta_{i,0} = \theta_{i,2}$$

- Discretizing the governing equation:

$$\boldsymbol{-4\theta_{i,1} + \theta_{i+1,1} + \theta_{i-1,1} + 2 * \theta_{i,2} = -\Delta x^2}$$

Along the line x = 1 (except at the corner points):

- i= 1
- Discretizing the boundary condition by CDS:

$$\frac{\partial \theta}{\partial x^*} = 0$$
$$\frac{\theta_{0,j} - \theta_{2,j}}{2 * \Delta x} = 0$$
$$\theta_{0,j} = \theta_{2,j}$$

- Discretizing the governing equation:

$$\boldsymbol{-4\theta_{1,j} + 2 * \theta_{2,j} + \theta_{1,j-1} + \theta_{1,j+1} = -\Delta x^2}$$

Along the line y = JMAX (except at the corner points):

- j = JMAX
- Discretizing the boundary condition with CDS:

$$\frac{\partial \theta}{\partial y^*} + Bi * \theta = 0$$
$$\frac{\theta_{i,JMAX+1} - \theta_{i,JMAX-1}}{2 * \Delta y} + Bi * \theta_{i,JMAX} = 0$$
$$\theta_{i,JMAX+1} = \theta_{i,JMAX-1} - 2 * Bi * \theta_{i,JMAX} * \Delta x$$

- Discretizing the governing equation:

$$\boldsymbol{-(4 + 2 * Bi * \Delta x)\theta_{i,JMAX} + \theta_{i-1,JMAX} + \theta_{i+1,JMAX} + 2 * \theta_{i,JMAX-1} = -\Delta x^2}$$

Along the line x = IMAX (except at the corner points):

- i = IMAX
- Discretizing the boundary condition with CDS:

$$\frac{\partial \theta}{\partial x^*} + Bi * \theta = 0$$
$$\frac{\theta_{IMAX+1,j} - \theta_{IMAX-1,j}}{2 * \Delta x} + Bi * \theta_{IMAX,j} = 0$$
$$\theta_{IMAX+1,j} = \theta_{IMAX-1,j} - 2 * Bi * \theta_{IMAX,j} * \Delta x$$

- Discretizing the governing equation:

$$-(\mathbf{4} + \mathbf{2} * \boldsymbol{Bi} * \Delta \boldsymbol{x})\boldsymbol{\theta}_{\boldsymbol{IMAX,j}} + \mathbf{2} * \boldsymbol{\theta}_{\boldsymbol{IMAX-1,j}} + \boldsymbol{\theta}_{\boldsymbol{IMAX,j+1}} + \boldsymbol{\theta}_{\boldsymbol{IMAX,j-1}} = -\Delta \boldsymbol{x}^{\mathbf{2}}$$

At the corner point (1,1):

- i = 1
- j = 1
- Imposing the boundary condition of both sides:

$$\theta_{0,1} = \theta_{2,1}$$
$$\theta_{1,0} = \theta_{1,2}$$

- Discretizing the governing equation:

$$-\mathbf{4}\boldsymbol{\theta}_{\mathbf{1,1}} + \mathbf{2} * \boldsymbol{\theta}_{\mathbf{2,1}} + \mathbf{2} * \boldsymbol{\theta}_{\mathbf{1,2}} = -\Delta \boldsymbol{x}^{\mathbf{2}}$$

At the corner point (1,JMAX):

- i = 1
- j = JMAX
- Imposing the boundary condition of both sides:

$$\theta_{0,JMAX} = \theta_{2,JMAX}$$

$$\theta_{1,JMAX+1} = \theta_{1,JMAX-1} - 2 * Bi * \theta_{1,JMAX} * \Delta x$$

- Discretizing the governing equation:

$$-(\mathbf{4} + \mathbf{2} * \boldsymbol{Bi} * \Delta \boldsymbol{x})\boldsymbol{\theta}_{\mathbf{1},\boldsymbol{JMAX}} + \mathbf{2} * \boldsymbol{\theta}_{\mathbf{2},\boldsymbol{JMAX}} + \mathbf{2} * \boldsymbol{\theta}_{\mathbf{1},\boldsymbol{JMAX-1}} = -\Delta \boldsymbol{x}^{\mathbf{2}}$$

At the corner point (IMAX,1):

- i = IMAX
- j = 1
- Imposing the boundary condition of both sides:

$$\theta_{IMAX,0} = \theta_{IMAX,2}$$

$$\theta_{IMAX+1,1} = \theta_{IMAX-1,1} - 2 * Bi * \theta_{IMAX,1} * \Delta x$$

- Discretizing the governing equation:

$$-(\mathbf{4} + \mathbf{2} * \boldsymbol{Bi} * \Delta \boldsymbol{x})\boldsymbol{\theta_{IMAX,1}} + \mathbf{2} * \boldsymbol{\theta_{IMAX,2}} + \mathbf{2} * \boldsymbol{\theta_{IMAX-1,1}} = -\Delta \boldsymbol{x^2}$$

At the corner point (IMAX,JMAX):

- i = IMAX
- j = 1
- Imposing the boundary condition of both sides:

$$\theta_{IMAX,JMAX+1} = \theta_{IMAX,JMAX-1} - 2 * Bi * \theta_{IMAX,JMAX} * \Delta x$$

$$\theta_{IMAX+1,JMAX} = \theta_{IMAX-1,JMAX} - 2 * Bi * \theta_{IMAX,JMAX} * \Delta x$$

- Discretizing the governing equation:

$$-(\mathbf{4} + \mathbf{4} * \boldsymbol{Bi} * \Delta \boldsymbol{x})\boldsymbol{\theta_{IMAX,JMAX}} + \mathbf{2} * \boldsymbol{\theta_{IMAX,JMAX-1}} + \mathbf{2} * \boldsymbol{\theta_{IMAX-1,JMAX}} = -\Delta \boldsymbol{x^2}$$

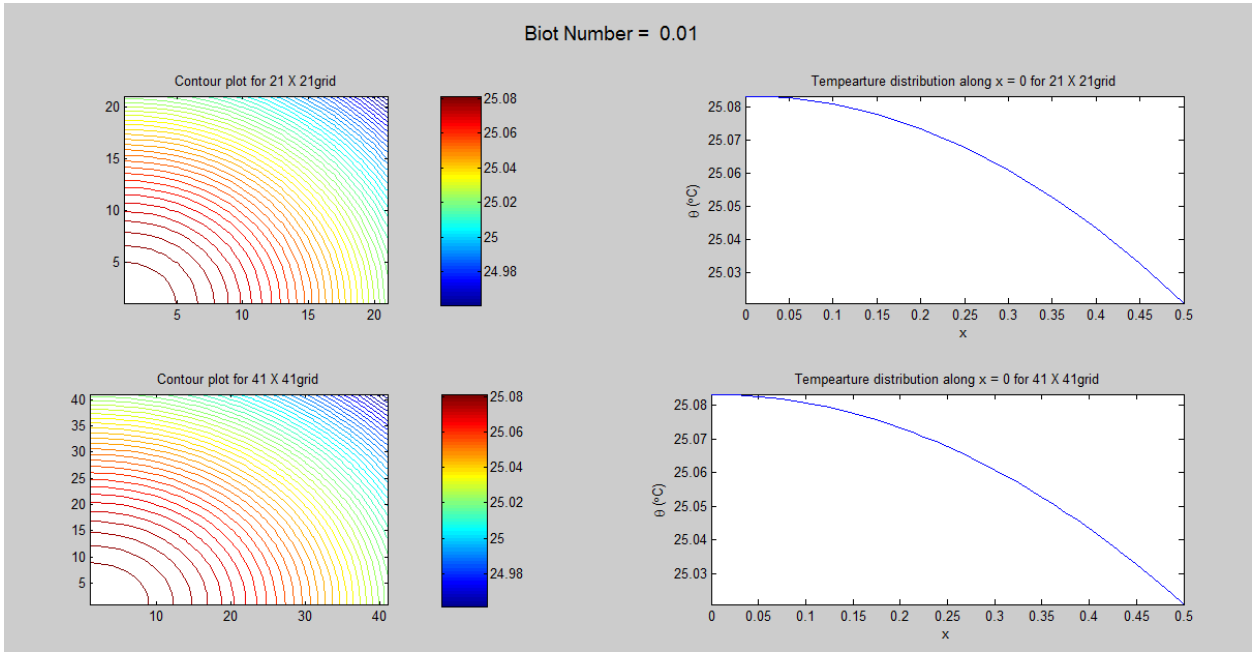
Algorithm:

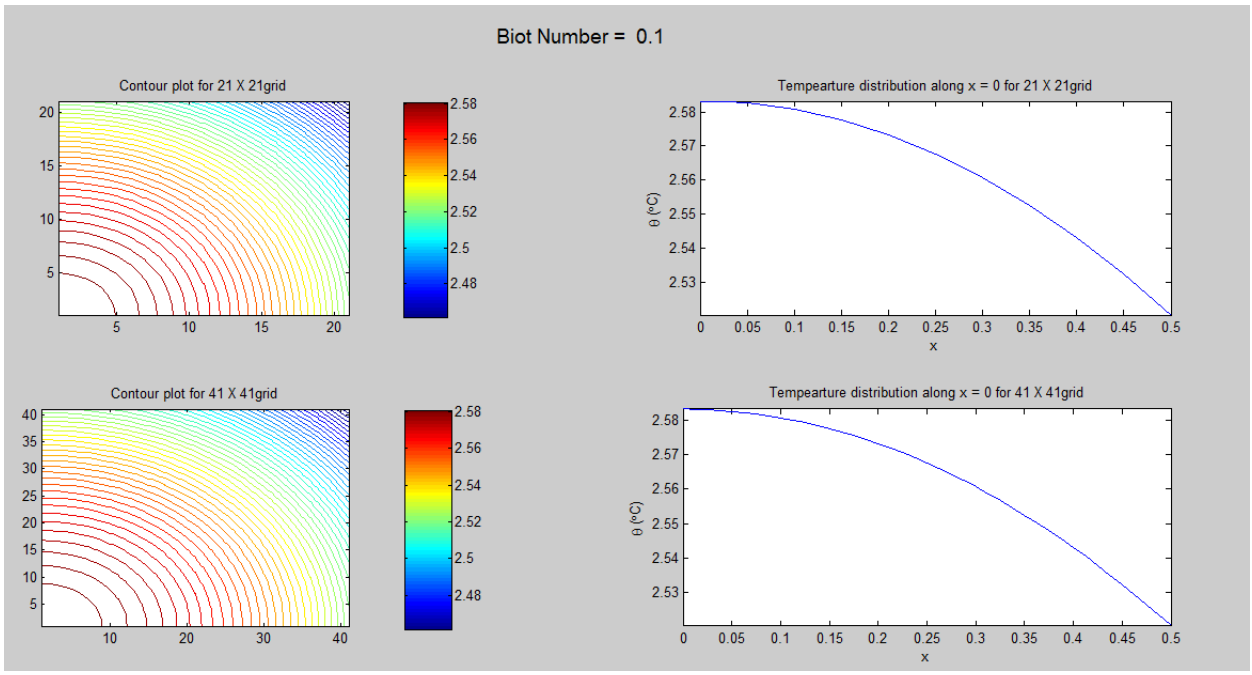
1. Read the data from the input file.
2. Calculate the number of grid points.
3. Form the coefficient matrices for each grid point.
4. Solve the system of linear equation using Gauss Seidel Method.
5. Plot the results.

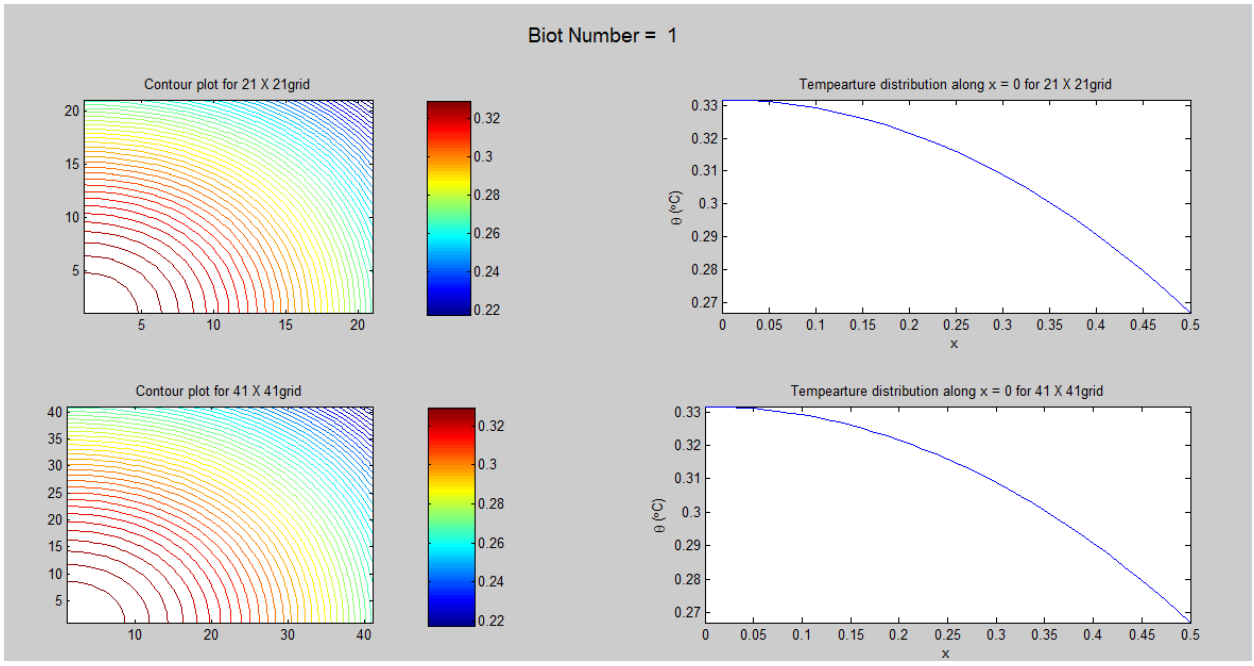
Input File:

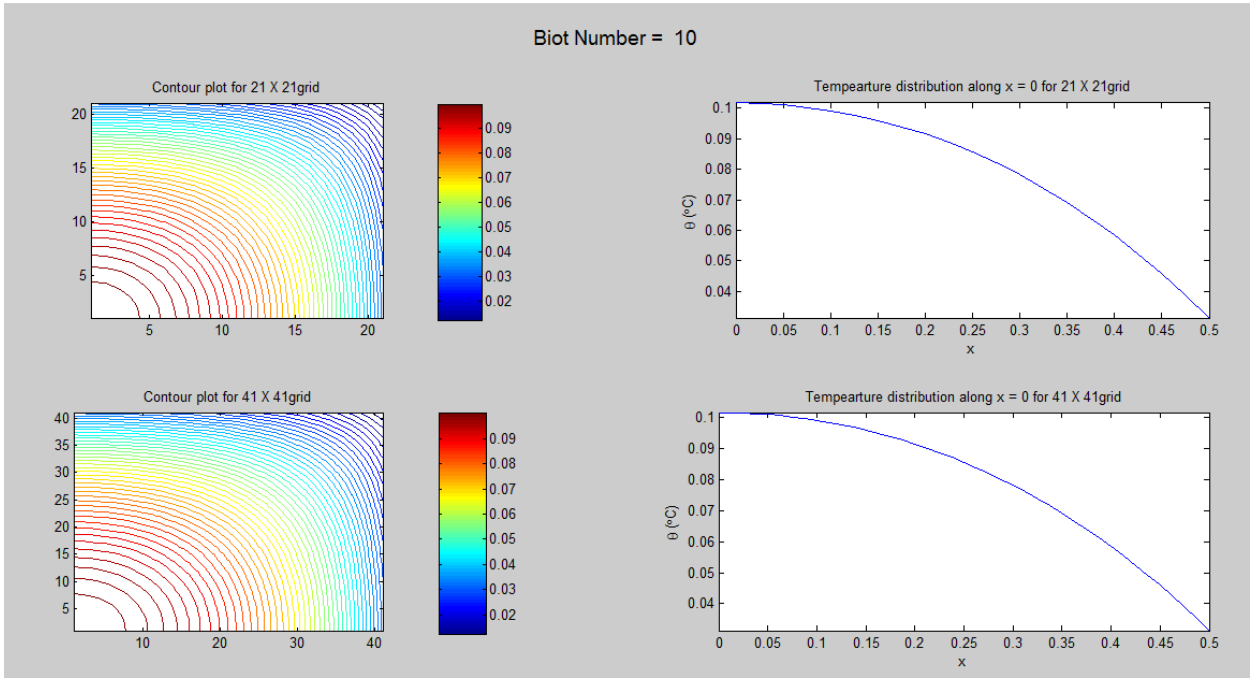
	A	B	C
1	***Number of grid points***	***Biot Number***	
2	21	0.01	
3	41	0.1	
4		1	
5		10	
6		100	
7			

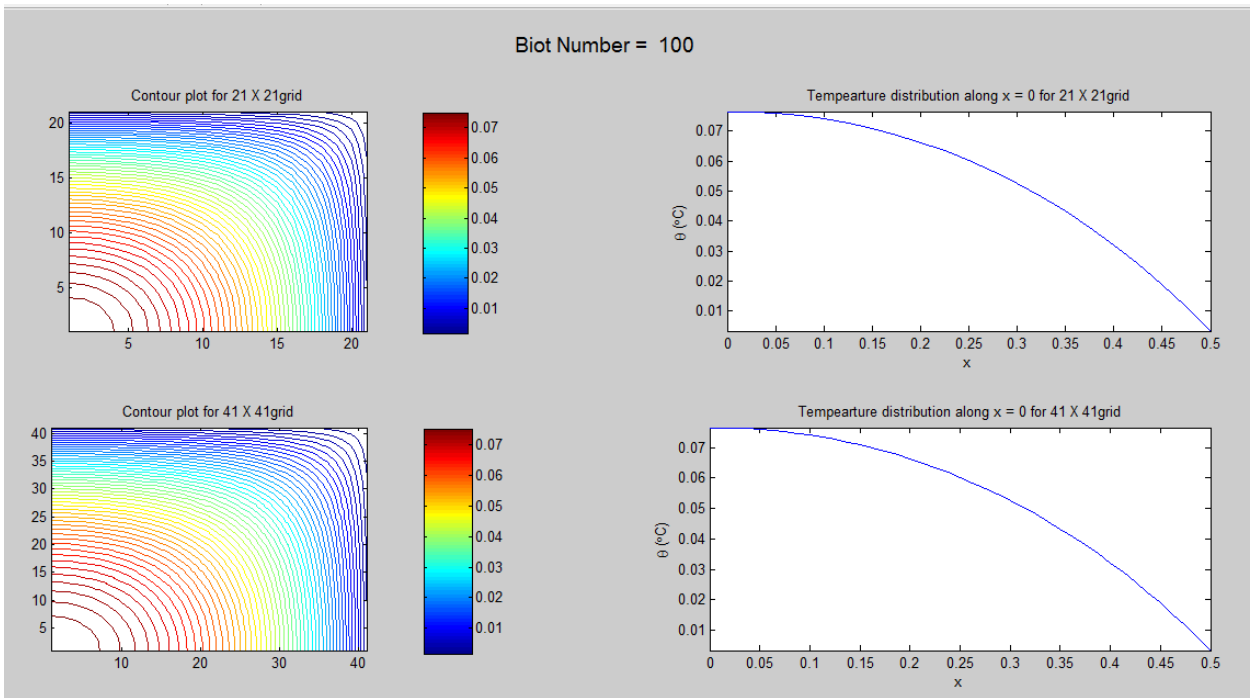
Results:




Biot Number = 0.1
Contour plot for 21 X 21grid
Tempearture distribution along x = 0 for 21 X 21grid
θ (°C)
x
Contour plot for 41 X 41grid
Tempearture distribution along x = 0 for 41 X 41grid
θ (°C)
x


Biot Number = 1
Contour plot for 21 X 21grid
Tempearture distribution along x = 0 for 21 X 21grid
θ (°C)
x
Contour plot for 41 X 41grid
Tempearture distribution along x = 0 for 41 X 41grid
θ (°C)
x


Biot Number = 10
Contour plot for 21 X 21grid
Tempearture distribution along x = 0 for 21 X 21grid
θ (°C)
x
Contour plot for 41 X 41grid
Tempearture distribution along x = 0 for 41 X 41grid
θ (°C)
x


Biot Number = 100
Contour plot for 21 X 21grid
Tempearture distribution along x = 0 for 21 X 21grid
θ (°C)
x
Contour plot for 41 X 41grid
Tempearture distribution along x = 0 for 41 X 41grid
θ (°C)
x

Output File:

	A	B	C	D	E	F	G	H	I
1	*Thetha_max obtained from Gauss Siedel Method*		Bi = 0.01	Bi = 0.1	Bi = 1	Bi = 10	Bi = 100		
2		21 X 21 Grid	25.08331836	2.583156	0.331634	0.101714	0.076565		
3		41 X 41 Grid	25.08329546	2.583135	0.331623	0.101726	0.07659		
4									
5	*Thetha_max obtained from Direct Method*		Bi = 0.01	Bi = 0.1	Bi = 1	Bi = 10	Bi = 100		
6		21 X 21 Grid	25.08333846	2.583154	0.331634	0.101714	0.076565		
7		41 X 41 Grid	25.08331902	2.583136	0.331623	0.101726	0.07659		
8									
9	*Calculation of error in the solution of Gauss Siedel Method*		Bi = 0.01	Bi = 0.1	Bi = 1	Bi = 10	Bi = 100		
10		21 X 21 Grid	8.01241E-07	5.43E-07	1.28E-07	4.57E-08	1.53E-08		
11		41 X 41 Grid	9.3922E-07	5.35E-08	9.2E-09	1.27E-08	4.95E-09		
12									

Appendix:

MA14M004_mainfile.m

```
%% Individual Details
% Author: Kumar Saurabh
% Roll No. MA14M004
% Main File

close all;
clear;
clc;
%% Reading data from Input file %%
Biot = xlsread('inputMA14M004_assign2.xlsx','B2:B6');
partition = xlsread('inputMA14M004_assign2.xlsx','A2:A3');

%% Computation
V = 1;
Max = zeros(2,5);
Max1 = zeros(2,5);

for k = 1:5

    Bi = Biot(k);
    for p = 1:2
        N = partition(p);
        deltax = 0.5/(N - 1);
        deltay = 0.5/(N - 1);

        B = zeros(N*N,N*N);
        C = zeros(N*N,1);
```

```
n = 1;
%% Computation of coeffecient matrix %%
for j = 1:N
    for i = 1:N
        if (i == 1)

            if(j == 1)
                B(n,((j - 1)*N + i+1)) = 2;
                B(n,(j - 1)*N + i) = -4;
                B(n,((j*N) + i)) = 2;
            elseif (j == N)
                B(n,((j-1)*N + i)) = -(4 + 2*Bi*deltax);
                B(n,((j - 2)*N + i)) = 2;
                B(n,((j - 1)*N+i+1)) = 2;
            else
                B(n,((j - 1) * N + i)) = -4;
                B(n,((j - 2) * N + i)) = 1;
                B(n,((j)*N + i)) = 1;
                B(n,((j - 1) * N + i + 1)) = 2;
            end
        elseif(i == N)
            if(j == 1)
                B(n,((j-1)*N + i)) = -(4 + 2*Bi*deltax);
                B(n,((j - 1)*N + i - 1)) = 2;
                B(n,((j)*N + i)) = 2;

            elseif(j == N)
                B(n,((j - 1)*N + i)) = - (4 + (4*Bi*deltax));
                B(n,((j - 1)*N + i - 1)) = 2;
                B(n,((j - 2)*N + i)) = 2;
            else
                B(n,((j - 1)*N + i)) = -(4 + (2*Bi*deltax));
                B(n,((j - 1)*N + i - 1)) = 2;
                B(n,((j - 2)*N + i)) = 1;
                B(n,((j)*N + i)) = 1;
            end
        elseif(j == 1)
            B(n,((j - 1)*N + i)) = -4;
            B(n,((j - 1)*N + i - 1)) = 1;
            B(n,((j - 1)*N + i + 1)) = 1;
            B(n,((j)*N + i)) = 2;
        elseif(j == N)
            B(n,((j - 1)*N + i)) = -(4 + 2*Bi*deltay);
            B(n,((j - 1)*N + i - 1)) = 1;
            B(n,((j - 1)*N + i + 1)) = 1;
            B(n,((j - 2)*N + i)) = 2;
        else
            B(n,((j - 1)*N + i)) = -4;
            B(n,((j - 1)*N + i - 1)) = 1;
            B(n,((j - 1)*N + i + 1)) = 1;
            B(n,((j)*N + i)) = 1;
            B(n,((j - 2)*N + i)) = 1;
        end
        C(n) = -(deltax)^2;
        n = n + 1;
```

```
            end
        end

        %% Solution of the system of Equation
        X = MA14M004_Gauss(B,C);  % Solution by Gauss Siedel Method
        X1 = B\C; % Solution by Direct Method

        %% Updating the temperature to corresponding temperature of the slab
        D = zeros(N,N);
        D1 = zeros(N,N);
        l = 1;
        for j = 1:N
            for i = 1:N
                D(i,j) = X(l);
                D1(i,j) = X1(l);
                l = l  + 1;
            end
        end

        %% Calculation of maximum temperature
        Max(p,k) = max(max(D)); % from Gauss Siedel Method
        Max1(p,k) = max(max(D1)); %from Direct Method

        %% Plotting of figure
        figure(V);
        subplot(2,2,p*2 - 1);
        contour(D,50);
        colorbar;
        title(['Contour plot for ',num2str(N), ' X ',num2str(N),'grid']);
        subplot(2,2,p*2);
        length = 0:deltax:0.5;
        plot(length,D(:,1));
        axis tight;
        xlabel('x');
        ylabel(['{\theta} (\circC)']);
        title(['Tempearture distribution along x = 0 for ',num2str(N), ' X
',num2str(N),'grid']);



    end
    suptitle(['Biot Number =  ',num2str(Bi)]);
    V = V+1;
end

%% Writing the final result to Excel file
xlswrite('outputMA14M004_asign2.xlsx',Max,'C2:G3');
xlswrite('outputMA14M004_asign2.xlsx',Max1,'C6:G7');
```

MA14M004_Gauss.m

```
%% Individual Details
% Author: Kumar Saurabh
% Roll No. MA14M004
% Gauss siedel Method to solve system of linear equation

%% Function :
function [ X ] = MA14M004_Gauss( A, B )

len = size(B,1);
error = 1999;
X1 = zeros(len,1);
X = zeros(len,1);
n = 0;
N = sqrt(len);
relax = 1.99; % overrelaxation for fast convergence

%% Traditional Gauss Siedel %%
% while(error > 10^(-6))
%     for j = 1:len
%         sum = 0;
%         for k = 1:len
%             if j ~= k
%                 sum = sum + A(j,k)*X(k);
%             end
%         end
%         X(j) =  X1(j) + relax*((B(j) - sum)/A(j,j) - X1(j));
%     end
%     alpha = zeros(len,1);
%     for i = 1:len
%             alpha(i) = abs((X(i) - X1(i))/X(i));
%     end
%     error = max(alpha);
%
%     disp(error);
%     clear X1;
%     X1 = X;
%     n = n + 1;
%
% end

%% Fast Gauss Siedel specific to this case %%
while(error > 10^(-6))
    for k = 1:len
        row = floor((k-1)/N) + 1;
        column = mod((k - 1),N) + 1;
        index = (row - 1)*N + column;
        left = index - 1;
        right = index + 1;
        top = index + N;
        bottom = index - N;

        if (row == 1)
            if (column == 1)
                sum = A(k,top)*X(top) + A(k,right)*X(right);
            elseif (column == N)
```

```
                sum = A(k,top)*X(top) + A(k,left)*X(left);
            else
                sum = A(k,top)*X(top) + A(k,left)*X(left) +
A(k,right)*X(right);
            end
        elseif (row == N)
            if (column == 1)
                sum = A(k,bottom)*X(bottom) + A(k,right)*X(right);
            elseif (column == N)
                sum = A(k,bottom)*X(bottom) + A(k,left)*X(left);
            else
                sum = A(k,bottom)*X(bottom) + A(k,left)* X(left) +
A(k,right)*X(right);
            end
        else
            if(column == 1)
                sum = A(k,bottom)*X(bottom) + A(k,right)*X(right) +
A(k,top)*X(top);
            elseif (column == N)
                sum = A(k,bottom)*X(bottom) + A(k,left)*X(left) +
A(k,top)*X(top);
            else
                sum = A(k,bottom)*X(bottom) + A(k,left)*X(left) +
A(k,top)*X(top) + A(k,right)*X(right);
            end
        end
        X(k) =  X1(k) + relax*((B(k) - sum)/A(k,k) - X1(k));
    end

    % Calculation of error
    error = 0;
    for i = 1:len
        error = error + abs((X(i) - X1(i))/X(i));
    end


    %disp(error);
    clear X1;
    X1 = X;
    n = n + 1;

end



%disp(n);
end
```