Solving the steady and unsteady 2D heat conduction problem

INTRODUCTION :

The project is based on the solving a single equation in different iterations using different methods. As per the different methods equations are solved by their respective boundary and initial conditions.

OBJECTIVE :

The main objective of this program is to understand the temperature distribution over a plate along the axis. The concept used in this projects are Explicit , Implicit , Jacobi , Gauss-Siedal , Gauss-Siedal with successive over relaxation and Crank nicolson methods. These methods are commonly used in many cfd simulations to solve complex equations.

PROCESS :

In this projects i used the two methods to solve this problem.

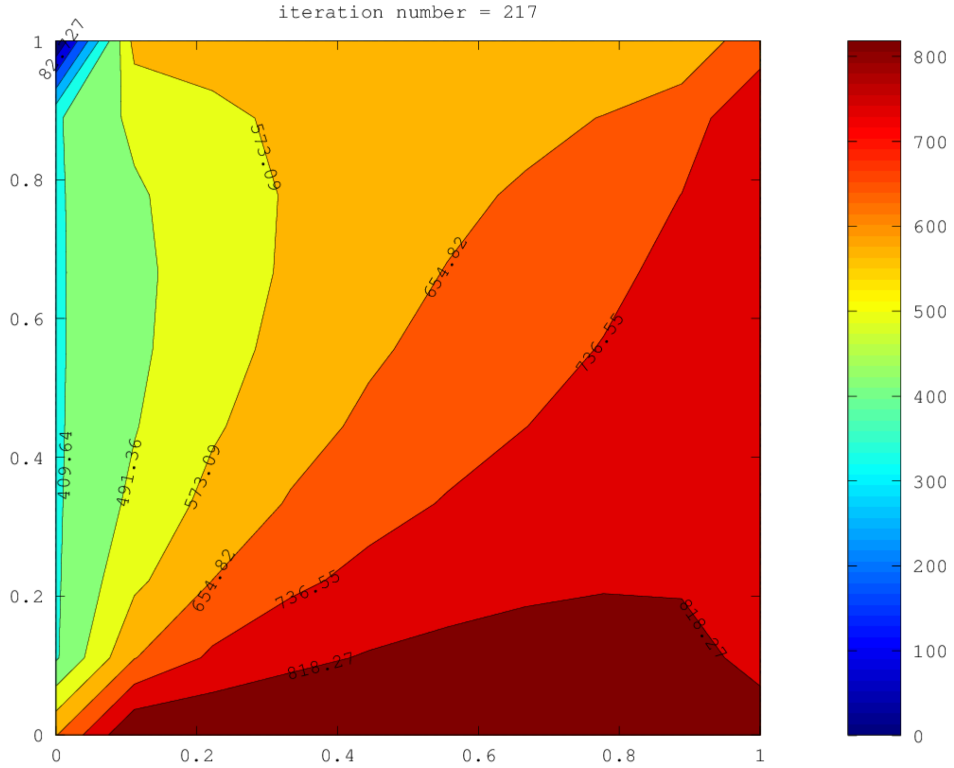
- Steady state solver .
- Unsteady state solver.

Both these methods used the above mentioned methods to solve the equations and the time period for all the methods are also noted .

Steady state solver :

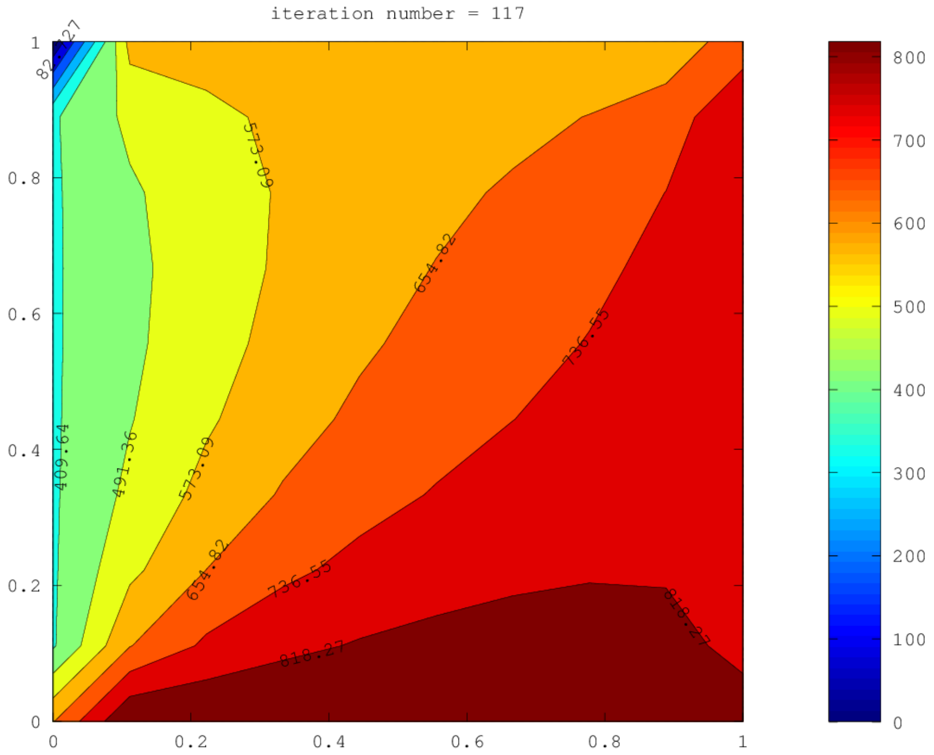
- **Jacobi - iteration:**

  217 iterations , iteration time = 0.46276 seconds.



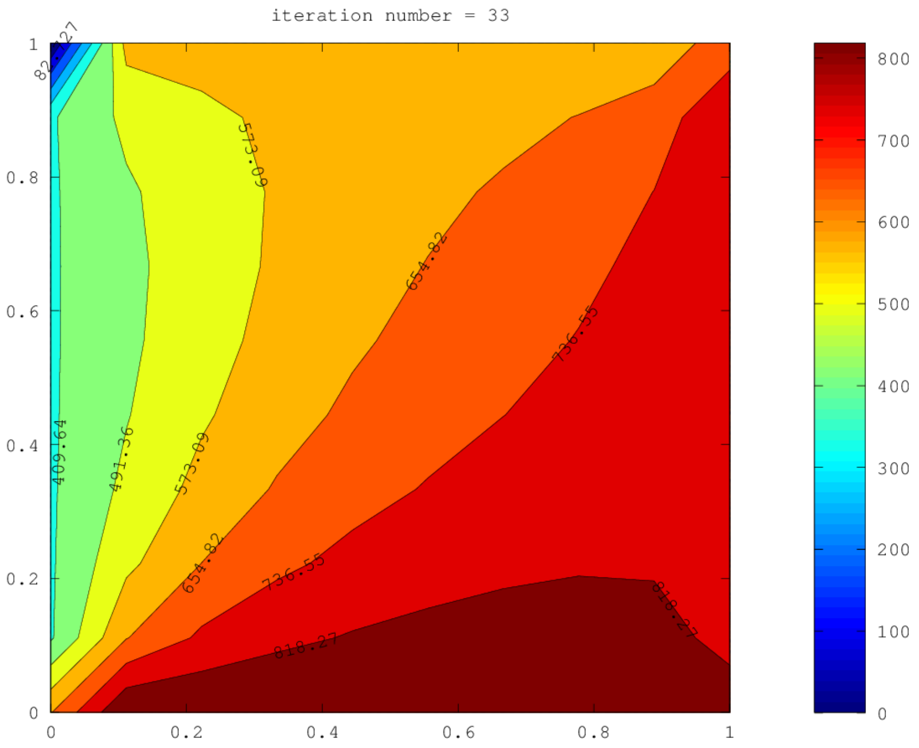
- **Gauss-Seidal iteration :**

117 iterations , iteration time = 0.21765 seconds.



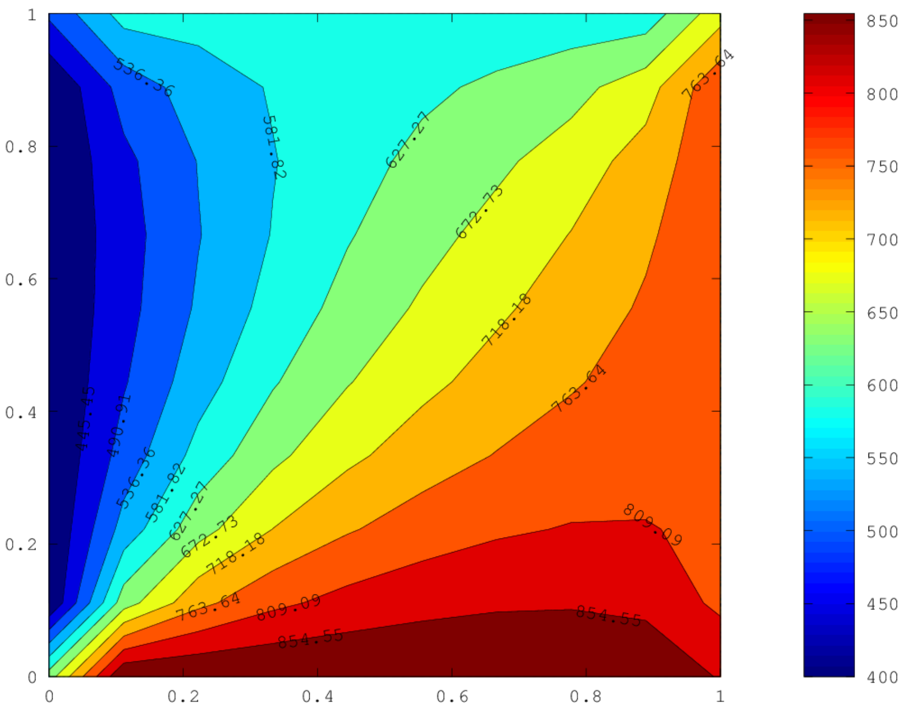
- **Successive over relaxtion :**

  33 iterations , iteration time = 0.11369 seconds.

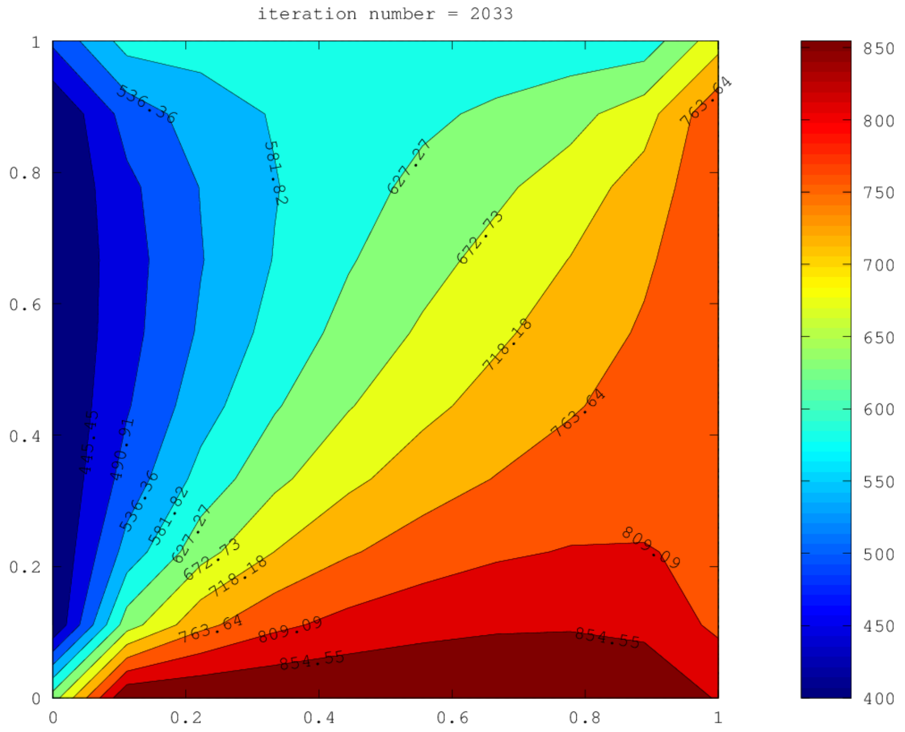


Unsteady state solver :

- **unsteady Explicit :**

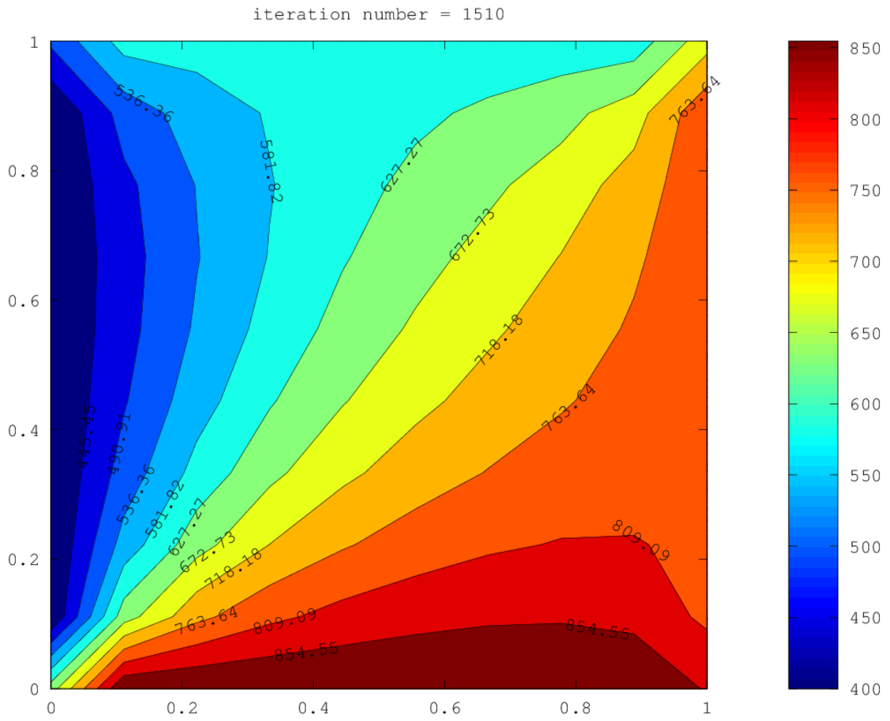


- **Unsteady implicit Jacobi:**
- 2033 iterations , iteration time = 4.3583 seconds.



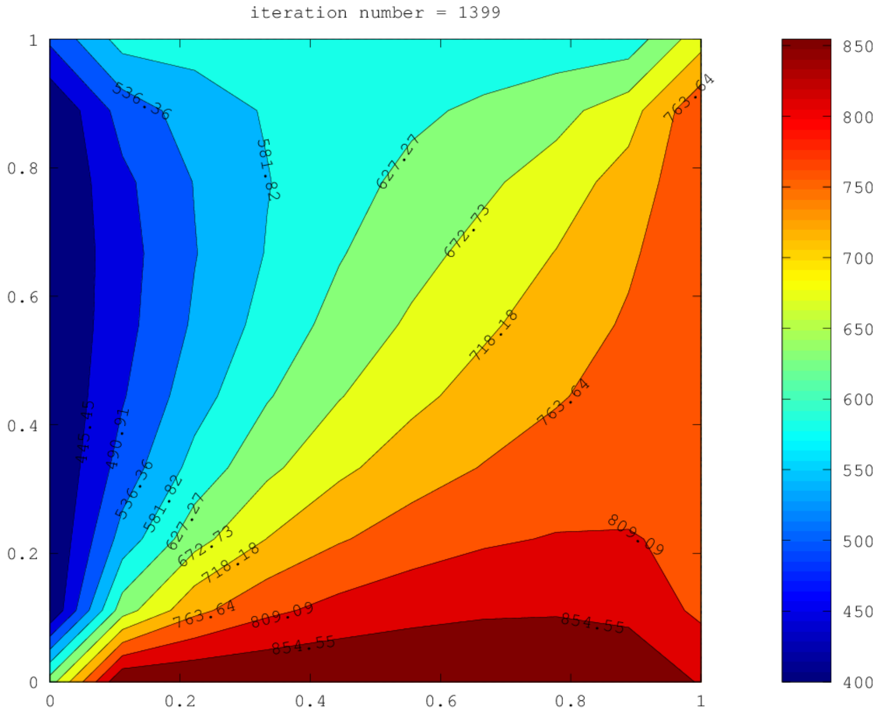
- **Unsteady implicit Gauss seidal :**

1510 iterations , iteration time = 3.4568 seconds.



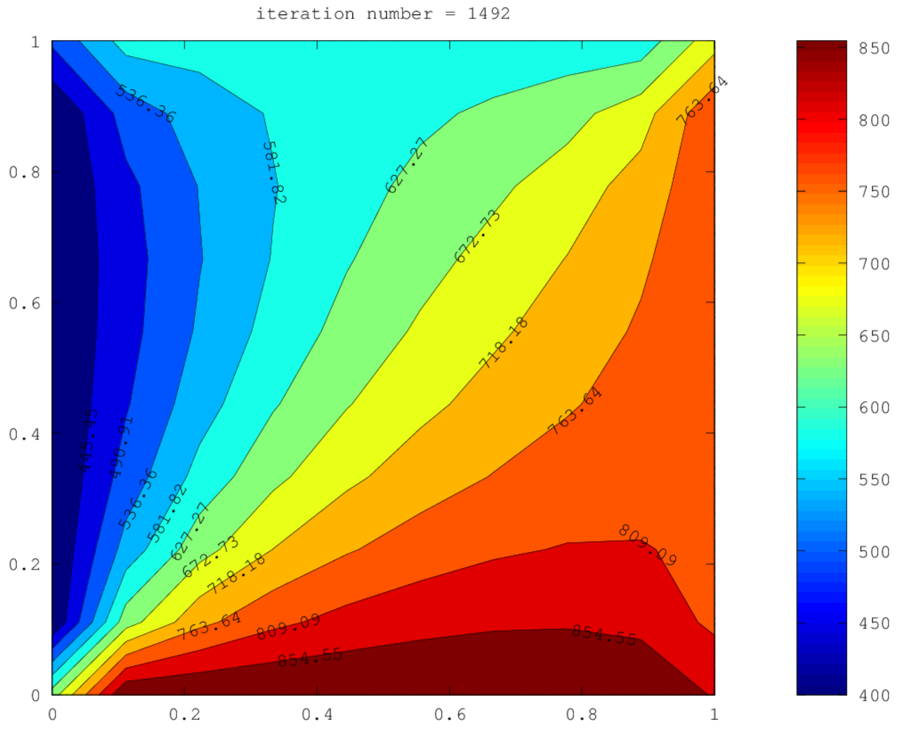
- **Unsteady implicit Gauss seidal with over relaxation :**

  1399 iterations , iteration time = 3.6511 seconds.



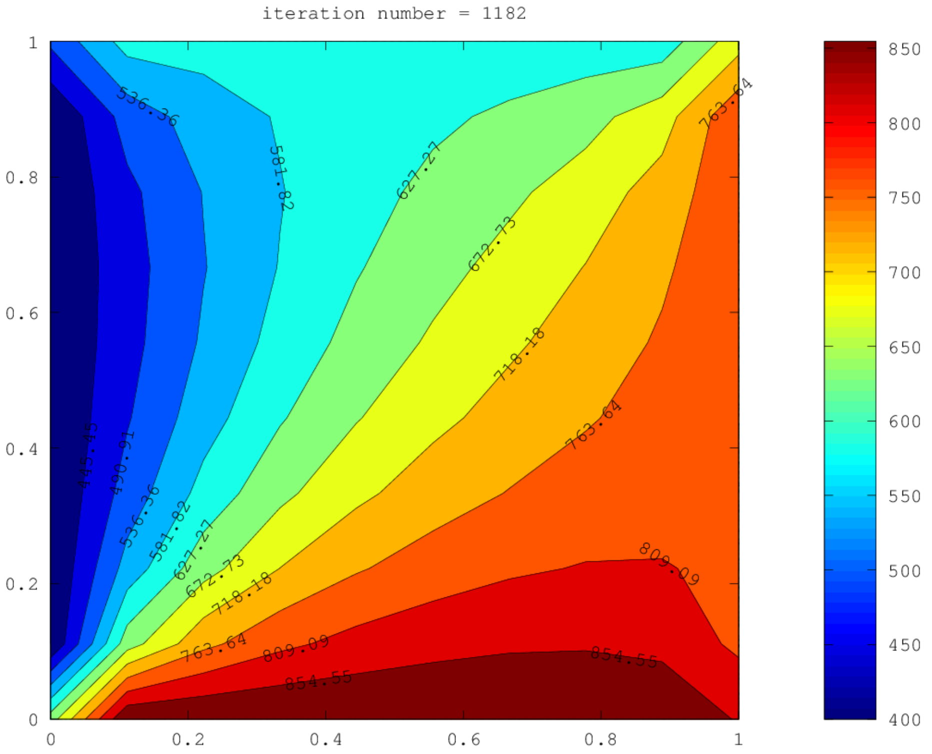
- **Unsteady implicit Crank Nicolson :**

1492 iterations , iteration time = 5.2180 seconds.



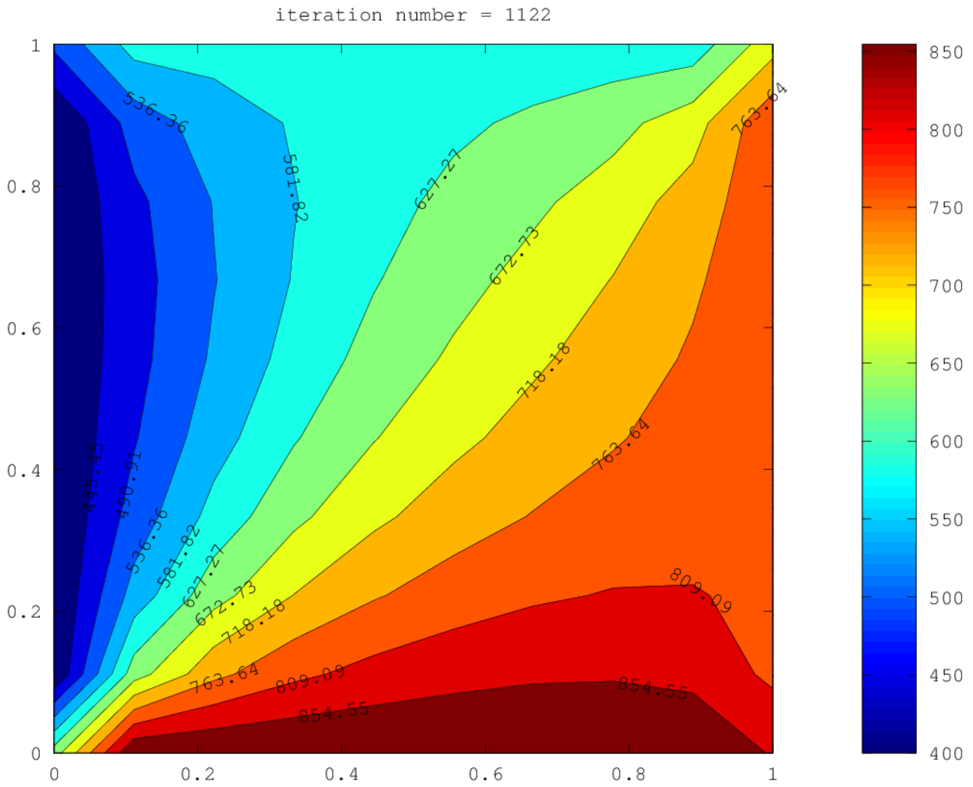
- **Unsteady implicit Crank Nicolson with Gauss Seidal :**

1182 iterations , iteration time = 4.0552 seconds.



- **Unsteady implicit Crank Nicolson SOR :**

1122 iterations , iteration time = 4.3534 seconds.



So these are the results provided by solving the equations by using different methods . We can notice mostly all the graphs are same this is because we are solving the same equation by different methods and the only difference thate we notice is the iteration number and the iteration time . These two vary because one method is always better than the another and the results are obatained .

PROGRAMS :

The above mentioned results are provided because of the programs which are given below .

These programs uses different kind of methods to solve the equation and to reduce the iteration as much as possible .

- **Steady state solver :**

In this method we are provided with error which is always within the tolerance criteria , So eventually most of the iterations will be given quickly and most probably the SOR method provides the result in much lesser time than any methods.

- **Unsteady state solver :**

In this method the time marching solution method is used . The dt value is kept minimum because whenever the dt value goes maximum the results will blow up . In implicit methods both time marching and convergence loop.

Now the programs for Steady and Unsteady solvers are given below respectievely.

Steady state solver :

```
clear all
close all
clc

% Now initializing the given condition

L = 1;
nx = 10;
ny = 10;
x = linspace(0,L,nx);
y = linspace(0,L,ny);
dx = x(2)-x(1);
dy = y(2)-y(1);
error = 9e9;
tol = 1e-4;



% Defining boundary condition
% As plotting the condition the graph appears in inverse position, so the bottom
and the top values are changed according to that.

% T_L = Temperature at left side
% T_T = Temperature at Top
% T_R = Temperature at Right
% T_B = Temperature at Bottom



T_L = 400;
T_T = 900;
T_R = 800;
T_B = 600;
T_matrix = ones(nx,ny);

% Now providing the temperature for the boundaries.
T_matrix(1,2:end-1) = T_T;
T_matrix(nx,2:end-1) = T_B;
T_matrix(2:end-1,1) = T_L;
T_matrix(2:end-1,ny) = T_R;

% Now providing the temperatues ate the each corner of the boundaries by taking
the average value temperature.

T_matrix(1,1) = (T_T + T_L)/2;
T_matrix(nx,ny) = (T_R + T_B)/2;
T_matrix(1,ny) = (T_T + T_R)/2;
T_Mmatrix(nx,1) = (T_L + T_B)/2;
```

```
iterative_solver = 3

graphics_toolkit("gnuplot")


Told = T_matrix;

k1 = (dx^2)/(2*(dx^2+dy^2));
k2 = (dy^2)/(2*(dx^2+dy^2));


% Iteration_solver 1 = jacobi_iteration
% 217 iteration
% Iteration time is = 0.46276 seconds.


if iterative_solver ==1
    jacobi_iteration = 1;
tic
while (error > tol)
   for i = 2:nx-1
    for j = 2:ny-1
 T_matrix(i,j) = k1*(Told(i,j-1)+Told(i,j+1))+k2*(Told(i-1,j)+Told(i+1,j));
 end
end

  error = max(max(abs(Told-T_matrix)));
  Told = T_matrix;
  jacobi_iteration = jacobi_iteration + 1;
end
time = toc;

% Now providing the color status for the output
  contourf(x,y,T_matrix);
  [a b] = contourf(x,y,T_matrix);
  clabel(a,b);
  colorbar

  title_text = sprintf('iteration number = %d' , jacobi_iteration);
  title(title_text)
  file_text = sprintf("S-jacobi_%d.png", iterative_solver);
  saveas(gca,file_text);
end

% Iteration_solver 2 = Gauss_seidal_iteration
% 117 iteration
% Iteration time is = 0.21765 seconds.




if iterative_solver ==2
   GS_iteration = 2;
tic
while (error > tol)
   for i = 2:nx-1
    for j = 2:ny-1
  T_matrix(i,j) = k1*(T_matrix(i,j-1)+Told(i,j+1))+k2*(T_matrix(i-
1,j)+Told(i+1,j));
 end
end

 error = max(max(abs(Told-T_matrix)));
 Told = T_matrix;
 GS_iteration = GS_iteration + 1;
end
time = toc;

% Now providing the color status for the output
 contourf(x,y,T_matrix);
 [a b] = contourf(x,y,T_matrix);
 clabel(a,b);
 colorbar

 title_text = sprintf('iteration number = %d' , GS_iteration);
 title(title_text)
 file_text = sprintf("S-Gauss_%d.png", iterative_solver);
 saveas(gca,file_text);
end

% Iteration_solver 3 = successive over relaxation
% 33 iteration
% Iteration time is = 0.11369 seconds.



if iterative_solver ==3;
   SOR_iteration = 3;
   opt_omega = 2/(1+sin(pi*dx));

tic
while (error > tol)
 for i =2:nx-1;
  for j = 2:ny-1;
  T_matrix(i,j) = opt_omega*(k1*(T_matrix(i,j-1)+T_matrix(i,j+1))+k2*(T_matrix(i-
1,j)+T_matrix(i+1,j)))+(1-opt_omega)*Told(i,j);
 end
end

error = max(max(abs(Told-T_matrix)));
Told = T_matrix;
SOR_iteration = SOR_iteration + 1;
end
 time = toc;

 % Now providing the color status for the output
 contourf(x,y,T_matrix);
 [a b] = contourf(x,y,T_matrix);
 clabel(a,b);
 colorbar;
 title_text = sprintf('iteration number = %d' , SOR_iteration);
 title(title_text)
 file_text = sprintf("S-SOR_%d.png", iterative_solver);
 saveas(gca,file_text);

 end
```

Unsteady state solver :

```
clear all
close all
clc

% Now intializing the given condition

L = 1;
nx = 10;
ny = 10;
x = linspace(0,L,nx);
y = linspace(0,L,ny);
dx = x(2)-x(1);
dy = y(2)-y(1);
error = 9e9;
tol = 1e-4;


% Defining boundary condition

% T_L = Temperature at left side
% T_T = Temperature at Top
% T_R = Temperature at Right
% T_B = Temperature at Bottom

T_L = 400;
T_T = 900;
T_R = 800;
T_B = 600;
T_matrix = ones(nx,ny);

% Now providing the temperature for the boundaries.
T_matrix(1,2:end-1) = T_T;
T_matrix(nx,2:end-1) = T_B;
T_matrix(2:end-1,1) = T_L;
T_matrix(2:end-1,ny) = T_R;


% Now providing the temperatues ate the each corner of the boundaries by taking
the average value temperature.

T_matrix(1,1) = (T_T + T_L)/2;
T_matrix(nx,ny) = (T_R + T_B)/2;
T_matrix(1,ny) = (T_T + T_R)/2;
T_matrix(nx,1) = (T_L + T_B)/2;


iterative_solver = 7

graphics_toolkit("gnuplot")
```

```
Told = T_matrix;

% Now providing the time step.
dt = 20;

% Now providing the thermal diffusivity.
alpha = 1e-4;

k4 = alpha*dt/(dx^2);
k5 = alpha*dt/(dy^2);

% Iteration_solver 1 = unsteady_explicit

if iterative_solver ==1
tic
for k = 1:200
 for i = 2:nx-1
  for j = 2:ny-1
T_matrix(i,j) = Told(i,j)+k4*(Told(i+1,j)-2*Told(i,j)+Told(i-1,j))+k5*(Told(i,j-
1)-2*Told(i,j)+Told(i,j+1));
 end
end

 Told = T_matrix;
end
time = toc

 % Now providing the color status for the output
  contourf(x,y,T_matrix);
  [a b] = contourf(x,y,T_matrix);
  clabel(a,b);
  colorbar

  file_text = sprintf("unsteady_explicit_%d.png", iterative_solver);
  saveas(gca,file_text);

end

% Iteration_solver 2 = unsteady_implicit_Jacobi.
% 2033 iterations.
% Iteration time is = 4.3583 seconds.
term1 = (1+2*k4+2*k5)^-1;
term2 = k4*term1;
term3 = k5*term1;


if iterative_solver ==2
tic

previous_dt = Told;
jacobi_iteration = 1;

 for k = 1:200

 while (error > tol)

  for i = 2:nx-1
   for j = 2:ny-1
   H = (Told(i-1,j)+Told(i+1,j));
   V = (Told(i,j-1)+Told(i,j+1));

T_matrix(i,j) = (previous_dt(i,j)*term1)+(term2*H)+(term3*V);
 end
end
  error = max(max(abs(Told-T_matrix)));
  Told = T_matrix;
  jacobi_iteration = jacobi_iteration + 1;
end

 previous_dt = T_matrix;
 error = 9e9;
end

time = toc;
  contourf(x,y,T_matrix);
  [a b] = contourf(x,y,T_matrix);
  clabel(a,b);
  colorbar
  title_text = sprintf('iteration number = %d', jacobi_iteration);
  title(title_text)
  file_text = sprintf("unsteady_implicit_Jacobi_%d.png" , iterative_solver);
  saveas(gca,file_text);
 end


 % Iteration_solver 3 = unsteady_implicit_Gauss_seidal.
 % 1510 iterations.
 % Iteration time is = 3.4568 seconds.

 term1 = (1+2*k4+2*k5)^-1;
term2 = k4*term1;
term3 = k5*term1;

if iterative_solver ==3
tic


 previous_dt = Told;
 GS_iteration = 1;

 for k = 1:200

 while (error > tol)
  for i = 2:nx-1
   for j = 2:ny-1

   H = (T_matrix(i-1,j)+Told(i+1,j));
   V = (T_matrix(i,j-1)+Told(i,j+1));

T_matrix(i,j) = (previous_dt(i,j)*term1)+(term2*H)+(term3*V);
 end
end

 error = max(max(abs(Told-T_matrix)));
 Told = T_matrix;
 GS_iteration = GS_iteration + 1;
end
```

```
 previous_dt = T_matrix;
 error = 9e9;
end
time = toc
  contourf(x,y,T_matrix);
  [a b] = contourf(x,y,T_matrix);
  clabel(a,b);
  colorbar
  title_text = sprintf('iteration number = %d', GS_iteration);
  title(title_text)
  file_text = sprintf("unsteady_implicit_Gauss_seidal_%d.png" , iterative_solver);
  saveas(gca,file_text);
end

% Iteration_solver 4 = unsteady_implicit_Gauss_seidal_with_over_relaxation
% 1399 iterations.
% iteration time is = 3.6511 seconds.


 term1 = (1+2*k4+2*k5)^-1;
term2 = k4*term1;
term3 = k5*term1;

if iterative_solver ==4
tic
 opt_omega = 2/(1+sin(pi*dx));

 previous_dt = Told;
 GS_iteration_SOR = 1;

  for k = 1:200

  while (error > tol)
   for i = 2:nx-1
    for j = 2:ny-1

   H = (T_matrix(i-1,j)+Told(i+1,j));
   V = (T_matrix(i,j-1)+Told(i,j+1));

T_matrix(i,j) = opt_omega*((previous_dt(i,j)*term1)+(term2*H)+(term3*V))+(1-
opt_omega)*previous_dt(i,j);
 end
end
 error = max(max(abs(Told-T_matrix)));
 Told = T_matrix;
 GS_iteration_SOR = GS_iteration_SOR + 1;
end
 previous_dt = T_matrix;
 error = 9e9;
end
time = toc;
  contourf(x,y,T_matrix);
  [a b] = contourf(x,y,T_matrix);
  clabel(a,b);
  colorbar
  title_text = sprintf('iteration number = %d', GS_iteration_SOR);
  title(title_text)
  file_text = sprintf("unsteady_implicit_Gauss_seidal_with_over_relaxation_%d.png"
, iterative_solver);
  saveas(gca,file_text);
 end


% Iteration_solver 5 = unsteady_implicit_crank_nicolson
% Providing the condition dx = dy so therefore  k4 = k5
% 1492 iterations.
% iteration time is = 5.2180 seconds.


term1 = (1-2*k4)/(1+2*k4);
term2 = k4/(2*(1+2*k4));

if iterative_solver ==5
tic

 previous_dt = Told;
 CN_iteration = 1;
for k = 1:200

 while (error > tol)
  for i = 2:nx-1
   for j = 2:ny-1

 H = (previous_dt(i+1,j)+previous_dt(i-1,j)+previous_dt(i,j+1)+previous_dt(i,j-
1));
 V = (Told(i+1,j)+Told(i-1,j)+Told(i,j+1)+Told(i,j-1));

 T_matrix(i,j) = (previous_dt(i,j)*term1)+(term2*H)+(term2*V);
 end
end

 error = max(max(abs(Told-T_matrix)));
 Told = T_matrix;
 CN_iteration = CN_iteration + 1;
end
 previous_dt = T_matrix;
 error = 9e9;
end
time = toc;
 contourf(x,y,T_matrix);
  [a b] = contourf(x,y,T_matrix);
  clabel(a,b);
  colorbar
  title_text = sprintf('iteration number = %d', CN_iteration);
  title(title_text)
  file_text = sprintf("unsteady_implicit_crank_nicolson_%d.png" ,
iterative_solver);
  saveas(gca,file_text);
 end

 % Iteration_solver 6 = unsteady_implicit_crank_nicolson_gauss_seidal.
 % Here the same is as k4 = k5
 %1182 iterations.
 % iteration time is = 4.0552 seconds.
 term1 = (1-2*k4)/(1+2*k4);
 term2 = k4/(2*(1+2*k4));

 if iterative_solver ==6
 tic
  previous_dt = Told;
  CNG_iteration = 1;

  for k = 1:200

   while (error > tol)
    for i = 2:nx-1
     for j = 2:ny-1

  H = (previous_dt(i+1,j)+previous_dt(i-1,j)+previous_dt(i,j+1)+previous_dt(i,j-
1));
 V = (Told(i+1,j)+T_matrix(i-1,j)+Told(i,j+1)+T_matrix(i,j-1));
  T_matrix(i,j) = (previous_dt(i,j)*term1)+(term2*H)+(term2*V);
 end
end

 error = max(max(abs(Told-T_matrix)));
 Told = T_matrix;
 CNG_iteration = CNG_iteration + 1;
end

 previous_dt = T_matrix;
 error = 9e9;
end
time = toc;
 contourf(x,y,T_matrix);
  [a b] = contourf(x,y,T_matrix);
  clabel(a,b);
  colorbar
  title_text = sprintf('iteration number = %d', CNG_iteration);
  title(title_text)
  file_text = sprintf("unsteady_implicit_crank_nicolson_gauss_seidal_%d.png" ,
iterative_solver);
  saveas(gca,file_text);
 end


 % Iteration_solver 7 = unsteady_implicit_crank_nicolson_SOR.
 % 1122 iterations.
 % iteration time is = 4.3534 seconds.


 term1 = (1-2*k4)/(1+2*k4);
 term2 = k4/(2*(1+2*k4));

 if iterative_solver ==7
 tic

  previous_dt = Told;
  CNS_iteration = 1;

  for k = 1:200
  opt_omega = 2/(1+sin(pi*dx));
  while (error > tol)
   for i = 2:nx-1
    for j = 2:ny-1

   H = (previous_dt(i+1,j)+previous_dt(i-1,j)+previous_dt(i,j+1)+previous_dt(i,j-
1));
   V = (Told(i+1,j)+T_matrix(i-1,j)+Told(i,j+1)+T_matrix(i,j-1));

  T_matrix(i,j) = opt_omega*((previous_dt(i,j)*term1)+(term2*H)+(term2*V))+(1-
opt_omega)*previous_dt(i,j);
 end
end
  error = max(max(abs(Told-T_matrix)));
  Told = T_matrix;
  CNS_iteration = CNS_iteration + 1;
 end

 previous_dt = T_matrix;
 error = 9e9;
end
time = toc;
  contourf(x,y,T_matrix);
  [a b] = contourf(x,y,T_matrix);
  clabel(a,b);
  colorbar
  title_text = sprintf('iteration number = %d', CNS_iteration);
  title(title_text)
  file_text = sprintf("unsteady_implicit_crank_nicolson_SOR_%d.png" ,
iterative_solver);
  saveas(gca,file_text);
 end
```