AIM: To solve the steady and unsteady state 2D heat conduction equation by using the point iterative techniques.

OBJECTIVE:

- To solve steady and unsteady state 2D heat conduction equation by explicit an implicit method.
- Solving equations by implicit method iterations techniques such as Jacobian, Gauss-Seidel & Successive Over Relaxation(SOR).
- The solution to meet the absolute error criteria of 1e-4 which is the acurracy of the solution.

Explicit method:

Explicit methods calculate the state of the system at a later time from the state of the system at the current time without the need to solve algebraic equations.

Implicit methods:

Implicit methods attempt to find a solution to the nonlinear system of equations iteratively by considering the current state of the system as well as its previous time state.

Iterative methods:

In computational mathematics, an iterative method is a mathematical procedure that uses an initial value to generate a sequence of improving approximate solutions for a class of problems, in which the n-th approximation is derived from the previous ones.

Types of iterative techniques:

1. Jacobbi method:

In jacobbi method we use the calculated value of the previous iteration in order to computed the current value.

2. Gauss seidal method:

In gauss seidal method we use the latest updated value in order to computed the value in the current state.

3. Sucessive over relaxation (SOR):

This type of iterative solver is used to improve the convergence rate by including a factor called relaxation factor.

Heat conduction equation in 3D space with change in time period is defined as,

`(∂T)/(∂t) = alpha . ((∂^2T)/(∂x^2)+(∂^2T)/(∂y^2)+(∂^2T)/(∂z^2))`

Heat conduction equation in 2D space with change in time period is defined as,

`(∂T)/(∂t) = alpha . ((∂^2T)/(∂x^2)+(∂^2T)/(∂y^2))`

where, alpha (α) = Thermal Diffusivity.

Steady state 2D Heat Conduction Equation:

At Steady state fluid properties does not change with respect to time.

i.e. `(∂T)/(∂t)=0`

The Equation for Steady State,

`((∂^2T)/(∂x^2)+(∂^2T)/(∂y^2))=0`

Discretization (Central Differencing) for Steady State:

`(T(i-1,j)-2*T(i,j)+T(i+1,j))/(Δx^2)+(T(i,j-1)-2*T(i,j)+T(i,j+1))/(Δy^2)=0`

Let,

Tp = T(i,j)

Tl = T(i-1,j)

Tr = T(i+1,j)

Tt = T(i,j-1)

Tb = T(i,j+1)

Solving the above equation for T(i,j),

`Tp=(1/k)((Tl+Tr)/(Δx^2))+(1/k)((Tt+Tb)/(Δy^2))`

where, `k = (2(Δx^2+Δy^2))/(Δx^2*Δy^2)`

Steady State Jacobian iterative solver:

`T(jacobi)=(Tp)^(n+1)=(1/k)((Tl+Tr)/(Δx^2))^(n)+(1/k)((Tt+Tb)/(Δy^2))^(n)`

Steady State Gauss Seidal iterative solver:

`T(GS)=(Tp)^(n+1)=(1/k)((Tl+Tr)/(Δx^2))^(n+1)+(1/k)((Tt+Tb)/(Δy^2))^(n+1)`

Steady State Gauss Seidal iterative solver:

`(Tp)^(n+1)= (Tp)^(n)*(1-w)+w*(T(GS)orT(jacobi))`

GIVEN DATA:

Top Boundary temperature, Tt = 600 K.

Bottom Boundary temperature, Tb = 900 K.

Left Boundary temperature, Tl = 400 K.

Right Boundary temperature, Tr = 800 K.

Domain length on x-axis = 1m.

Domain length on y-axis = 1m.

Thermal diffusivity, α = 1.1

Relaxation factor, w = 1.1

Number of node at x and y axis = 10

Timestep = 1e-3.

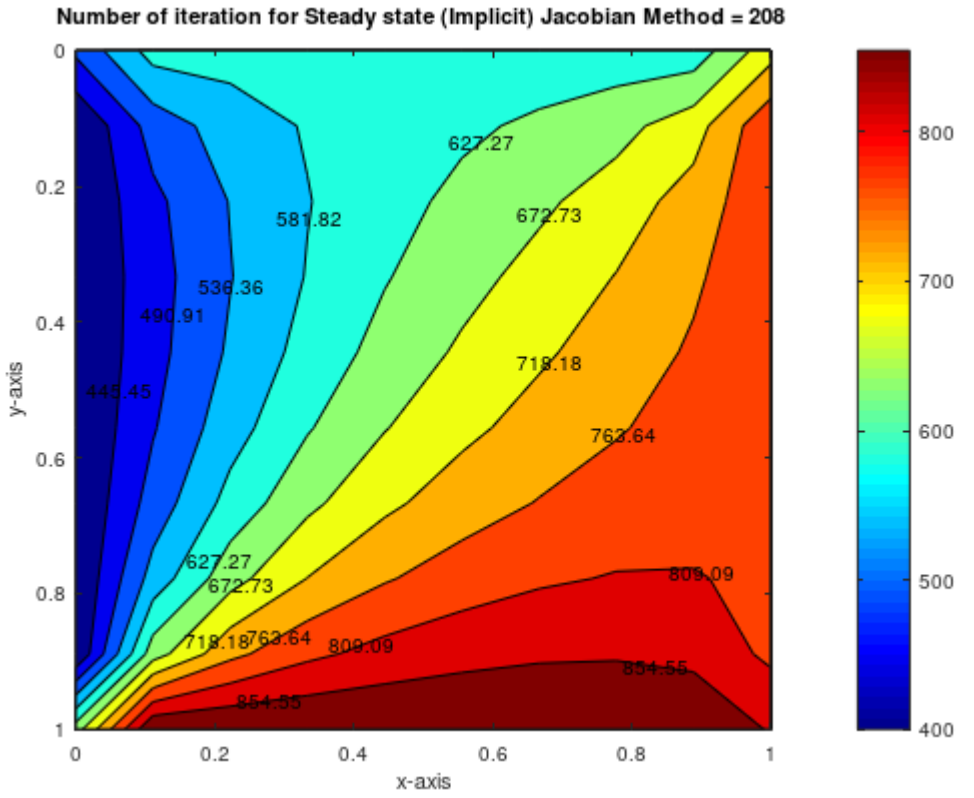
MATLAB CODE FOR STEADY STATE IMPLICIT METHOD:

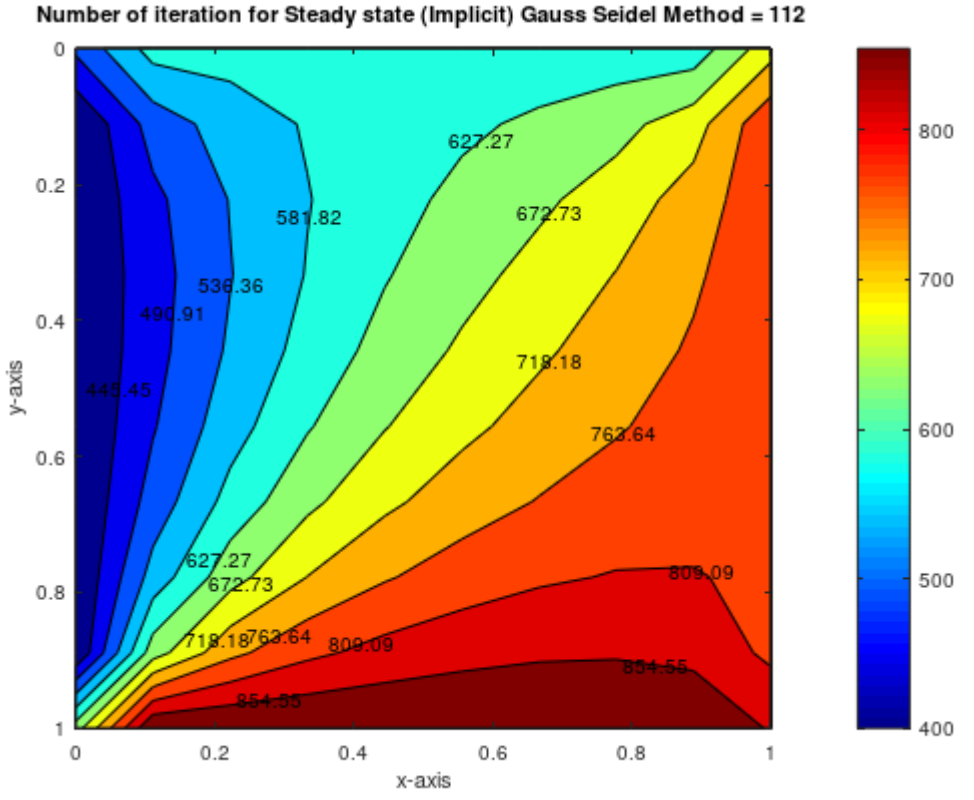
```
% Steady state (Implicit) 2-D HEAT CONDUCTION.
% ITERATIVE METHOD - JACOBIAN, GAUSS SEIDAL, SUCCESSIVE OVER RELAXATION(SOR).

close all
clear all
clc

% Inputs
nx = 10;
ny = 10;

Tx = linspace(0,1,nx);
Ty = linspace(0,1,ny);

% Defining Grid Size
dx = 1/(nx-1);
dy = 1/(ny-1);

% Boundary Condition
T = 300*ones(nx,ny);
T(1,:) = Tt = 600;
T(nx,:) = Tb = 900;
T(:,1) = Tl = 400;
T(:,ny) = Tr = 800;

T(1,1) = (Tt + Tl)/2;
T(1,ny) = (Tt + Tr)/2;
T(nx,1) = (Tb + Tl)/2;
T(nx,ny) = (Tb + Tr)/2;

Told = T;
error = 9e9;
tol = 1e-4;
k = (2*(dx^2+dy^2))/(dx^2*dy^2);
w = 1.1;
itaration_solver = input('Enter the value 1. Jacobian 2. Gauss seidel 3. SOR : ');

% Jacobian Method
```

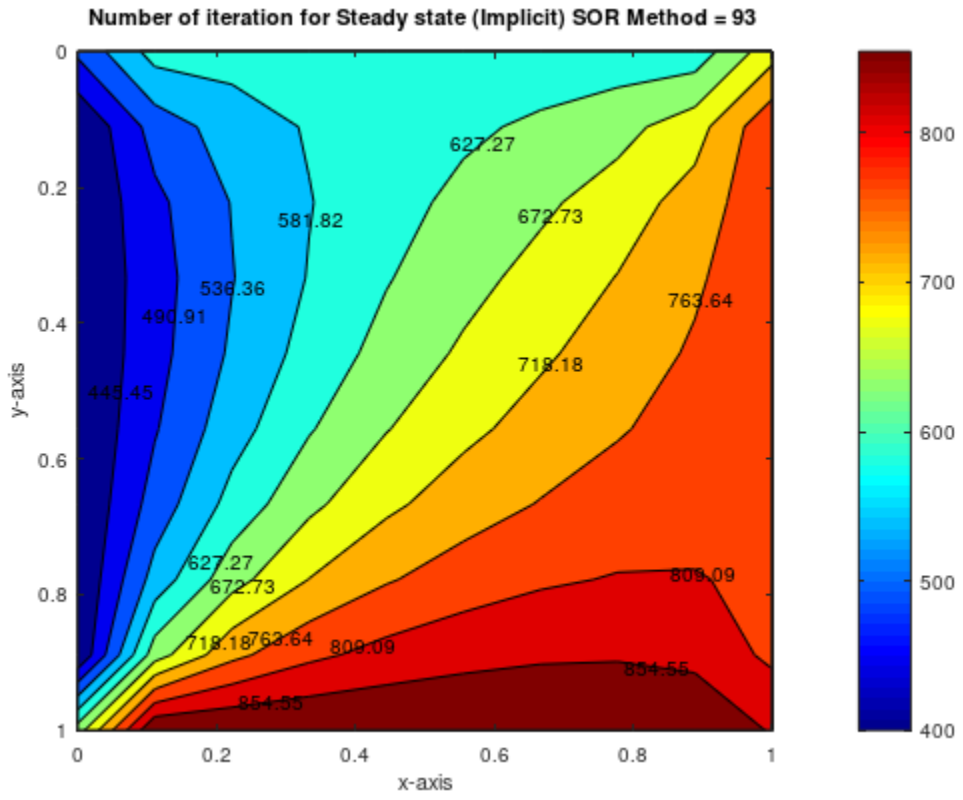
```
if itaration_solver == 1
  jacobian_iter = 1;
  while (error>tol)
    for i = 2:nx-1
      for j = 2:ny-1
        T(i,j) = (1/k)*(((Told(i-1,j)+Told(i+1,j))/dx^2) + ((Told(i,j-
1)+Told(i,j+1))/dy^2));
      endfor
    endfor
    error = max(max(abs(Told-T)));
    Told = T;
    jacobian_iter = jacobian_iter+1;
    contourf(Tx,Ty,T,'ShowText','on')
    colorbar
    colormap(jet)
    set(gca, 'ydir','reverse')
    xlabel('x-axis')
    ylabel('y-axis')
    title(sprintf('Number of iteration for Steady state (Implicit) Jacobian Method = %d
',jacobian_iter));
    pause(0.0003)
  endwhile

% Gauss Seidel Method
elseif itaration_solver == 2
    gaussseidel_iter = 1;
  while (error>tol)
    for i = 2:nx-1
      for j = 2:ny-1
        T(i,j) = (1/k)*(((T(i-1,j)+T(i+1,j))/dx^2) + ((T(i,j-1)+T(i,j+1))/dy^2));
      endfor
    endfor
    error = max(max(abs(Told-T)));
    Told = T;
    gaussseidel_iter = gaussseidel_iter+1;
    contourf(Tx,Ty,T,'ShowText','on')
    colorbar
    colormap(jet)
    set(gca, 'ydir','reverse')
    xlabel('x-axis')
    ylabel('y-axis')
    title(sprintf('Number of iteration for Steady state (Implicit) Gauss Seidel Method
= %d ',gaussseidel_iter));
    pause(0.0003)
  endwhile

% SOR Method
elseif itaration_solver == 3
      sor_iter = 1;
  while (error>tol)
    for i = 2:nx-1
      for j = 2:ny-1
        T(i,j) = (1-w)*Told(i,j) + w*(1/k)*(((T(i-1,j)+T(i+1,j))/dx^2) + ((T(i,j-
1)+T(i,j+1))/dy^2));
      endfor
    endfor
    error = max(max(abs(Told-T)));
    Told = T;
    sor_iter = sor_iter+1;
    contourf(Tx,Ty,T,'ShowText','on')
    colorbar
    colormap(jet)
    set(gca, 'ydir','reverse')
    xlabel('x-axis')
    ylabel('y-axis')
    title(sprintf('Number of iteration for Steady state (Implicit) SOR Method = %d
',sor_iter));
```

```
    pause(0.0003)
  endwhile
endif
```

OUTPUT :







Un-Steady state 2D Heat Conduction Equation:

At Un-Steady state fluid properties change with respect to time.

i.e. `(∂T)/(∂t)!=0`

The Equation for Un-Steady State,

`(∂T)/(∂t) = alpha . ((∂^2T)/(∂x^2)+(∂^2T)/(∂y^2))`

Discretization (Central Differencing) for Un-Steady State:

`(Tp^(n+1)-Tp^(n))/(Δt)-α*((Tl-2*Tp+Tr)/(Δx^2)+(Tt-2*Tp+Tb)/(Δy^2))=0`

Let,

Tp = T(i,j)

Tl = T(i-1,j)

Tr = T(i+1,j)

Tt = T(i,j-1)

Tb = T(i,j+1)

Solving the above equation explicitly,

`Tp^(n+1)=(1-2*k1-2*k2)*Tp^(n)+k1*(Tl+Tr)^n+k2*(Tt+Tb)^n`

where,

`k1=α*(Δt)/(Δx^2)`

`k2=α*(Δt)/(Δy^2`

Stability criteria for 2D heat conduction explicitly,

`k1+k2<=0.5`

Solving the above equation implicitly,

`Tp^(n+1)=(1/k)*(Tp^(n)+k1*(Tl+Tr)^(n+1)+k2*(Tt+Tb)^(n+1))`

where,

`k=1+2*k1+2*k2`

`k1=α*(Δt)/(Δx^2)`

`k2=α*(Δt)/(Δy^2`

Un-Steady State Jacobian iterative solver:

`T(jacobi)=Tp^(n+1)=(1/k)*(Tp^(n)+k1*(Tl+Tr)^(n)+k2*(Tt+Tb)^(n))`

Un-Steady State Gauss Seidal iterative solver:

`T(GS)=Tp^(n+1)=(1/k)*(Tp^(n)+k1*(Tl^(n)+Tr^(n+1))+k2*(Tt^(n)+Tb^(n+1)))`

Un-Steady State Gauss Seidal iterative solver:

`(Tp)^(n+1)= (Tp)^(n)*(1-w)+w*(T(GS)orT(jacobi))`

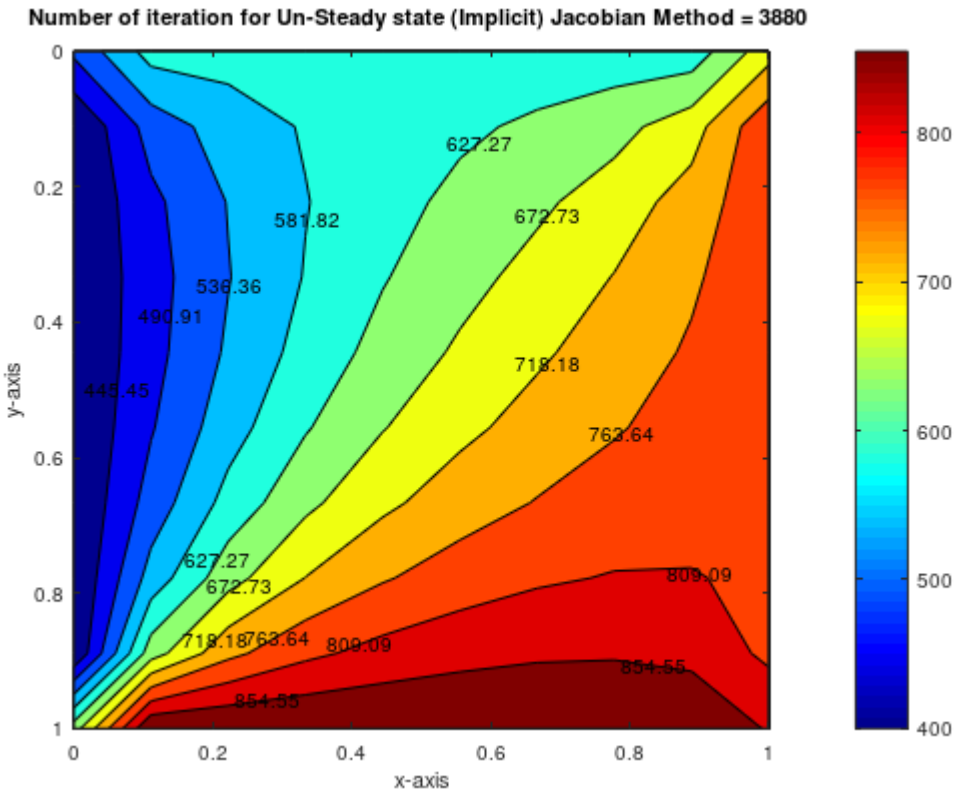
MATLAB CODE FOR UN-STEADY STATE IMPLICIT METHOD:

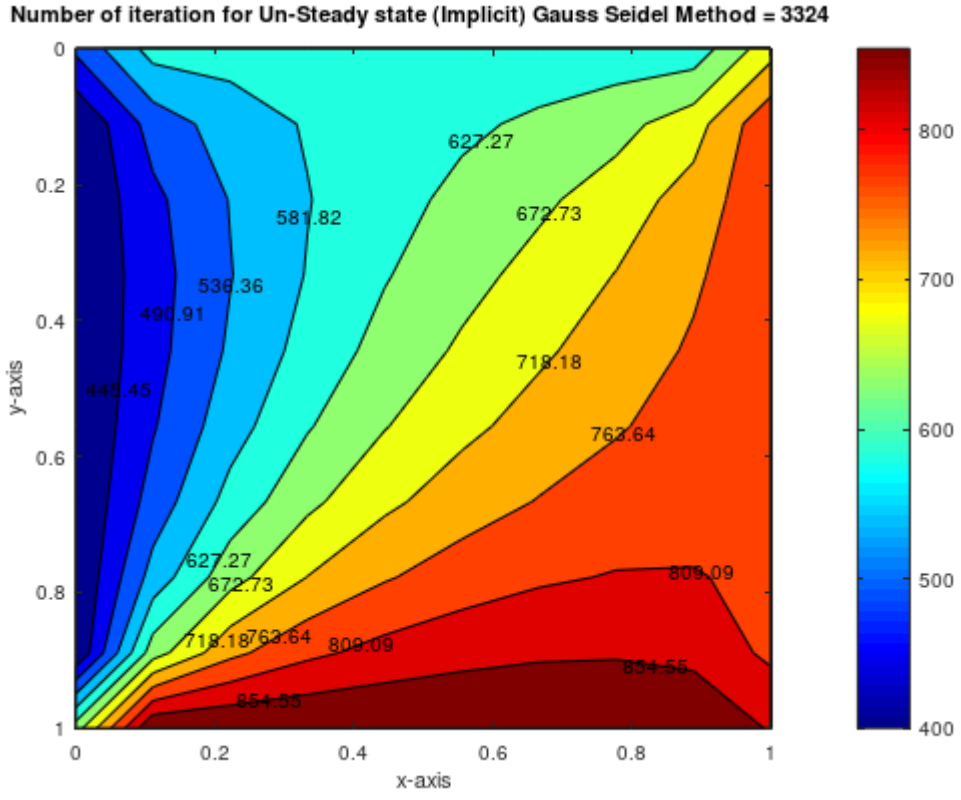
```
% Un-Steady state (Implicit) 2-D HEAT CONDUCTION.
% ITERATIVE METHOD - JACOBIAN, GAUSS SEIDAL, SUCCESSIVE OVER RELAXATION(SOR).

close all
clear all
clc

% Inputs
nx = 10;
ny = 10;
nt = 1400;

Tx = linspace(0,1,nx);
Ty = linspace(0,1,ny);

% Defining Grid Size
dx = 1/(nx-1);
dy = 1/(ny-1);

% Boundary Condition
T = 300*ones(nx,ny);
```

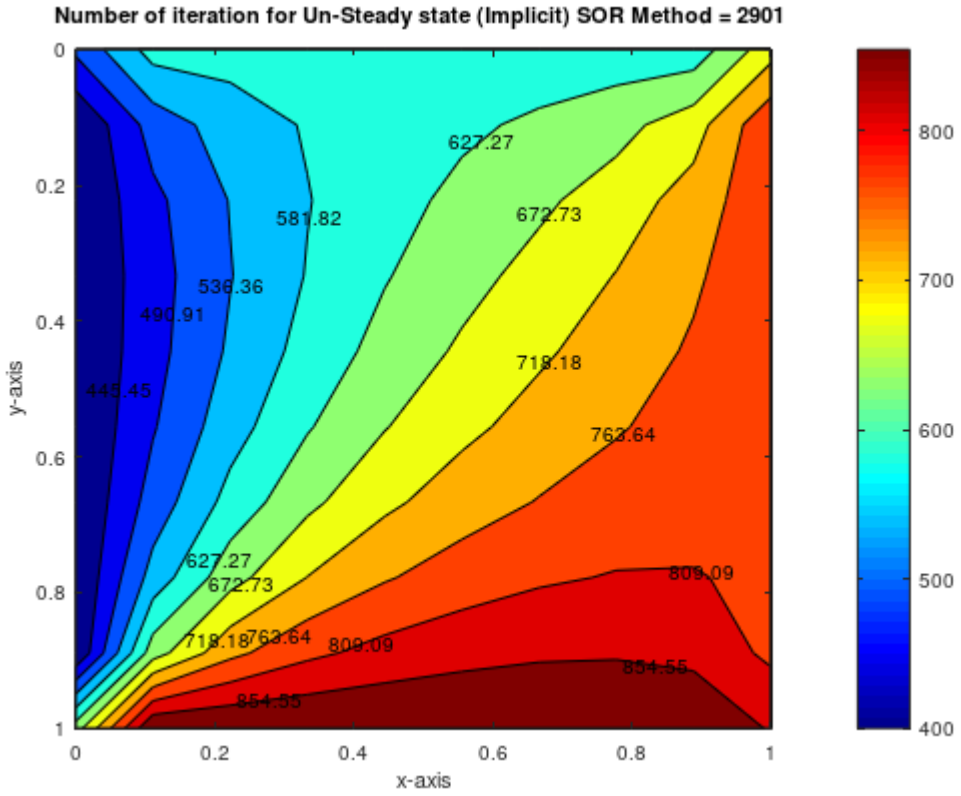
```
T(1,:) = Tt = 600;
T(nx,:) = Tb = 900;
T(:,1) = Tl = 400;
T(:,ny) = Tr = 800;

T(1,1) = (Tt + Tl)/2;
T(1,ny) = (Tt + Tr)/2;
T(nx,1) = (Tb + Tl)/2;
T(nx,ny) = (Tb + Tr)/2;

Told = T;
Tp = T;
error = 9e9;
tol = 1e-4;
alpha = 1.1;
dt = 1e-3;
k1 = (alpha*dt)/(dx^2);
k2 = (alpha*dt)/(dy^2);
w = 1.1;
itaration_solver = input('Enter the value 1. Jacobian 2. Gauss seidel 3. SOR : ');

% Jacobian Method
if itaration_solver == 1
  jacobian_iter = 1;
 for k = 1:nt
   error = 9e9;
  while (error>tol)
    for i = 2:nx-1
      for j = 2:ny-1
        Term1 = 1/(1+2*k1+2*k2);
        Term2 = k1*Term1;
        Term3 = k2*Term1;
        H = Told(i-1,j)+Told(i+1,j);
        V = Told(i,j-1)+Told(i,j+1);
        T(i,j) = (Term1*Tp(i,j))+Term2*H+Term3*V;
      endfor
    endfor
    error = max(max(abs(Told-T)));
    Told = T;
    jacobian_iter = jacobian_iter+1;
    contourf(Tx,Ty,T,'ShowText','on')
    colorbar
    colormap(jet)
    set(gca, 'ydir','reverse')
    xlabel('x-axis')
    ylabel('y-axis')
    title(sprintf('Number of iteration for Un-Steady state (Implicit) Jacobian Method =
%d ',jacobian_iter));
    pause(0.0003)
  endwhile
  Tp = Told;
 endfor

% Gauss Seidel Method
elseif itaration_solver == 2
  gaussseidel_iter = 1;
 for k = 1:nt
   error = 9e9;
  while (error>tol)
    for i = 2:nx-1
      for j = 2:ny-1
        Term1 = 1/(1+2*k1+2*k2);
        Term2 = k1*Term1;
        Term3 = k2*Term1;
        H = T(i-1,j)+T(i+1,j);
        V = T(i,j-1)+T(i,j+1);
        T(i,j) = (Term1*Tp(i,j))+Term2*H+Term3*V;
```

```
      endfor
    endfor
    error = max(max(abs(Told-T)));
    Told = T;
    gaussseidel_iter = gaussseidel_iter+1;
    contourf(Tx,Ty,T,'ShowText','on')
    colorbar
    colormap(jet)
    set(gca, 'ydir','reverse')
    xlabel('x-axis')
    ylabel('y-axis')
    title(sprintf('Number of iteration for Un-Steady state (Implicit) Gauss Seidel
Method = %d ',gaussseidel_iter));
    pause(0.0003)
  endwhile
  Tp = Told;
 endfor

% SOR Method
elseif itaration_solver == 3
  sor_iter = 1;
 for k = 1:nt
   error = 9e9;
  while (error>tol)
    for i = 2:nx-1
      for j = 2:ny-1
        Term1 = 1/(1+2*k1+2*k2);
        Term2 = k1*Term1;
        Term3 = k2*Term1;
        H = T(i-1,j)+T(i+1,j);
        V = T(i,j-1)+T(i,j+1);
        T(i,j) = (1-w)*Told(i,j) + w*((Term1*Tp(i,j))+(Term2*H+Term3*V));
      endfor
    endfor
    error = max(max(abs(Told-T)));
    Told = T;
    sor_iter = sor_iter+1;
    contourf(Tx,Ty,T,'ShowText','on')
    colorbar
    colormap(jet)
    set(gca, 'ydir','reverse')
    xlabel('x-axis')
    ylabel('y-axis')
    title(sprintf('Number of iteration for Un-Steady state (Implicit) SOR Method = %d
',sor_iter));
    pause(0.0003)
  endwhile
  Tp = Told;
 endfor
endif
```

OUTPUT :


Number of iteration for Un-Steady state (Implicit) Jacobian Method = 3880
y-axis
x-axis
627.27
581.82
672.73
536.36
490.91
718.18
445.45
763.64
809.09
854.55


Number of iteration for Un-Steady state (Implicit) Gauss Seidel Method = 3324
y-axis
x-axis
627.27
581.82
672.73
536.36
490.91
718.18
445.45
763.64
809.09
854.55

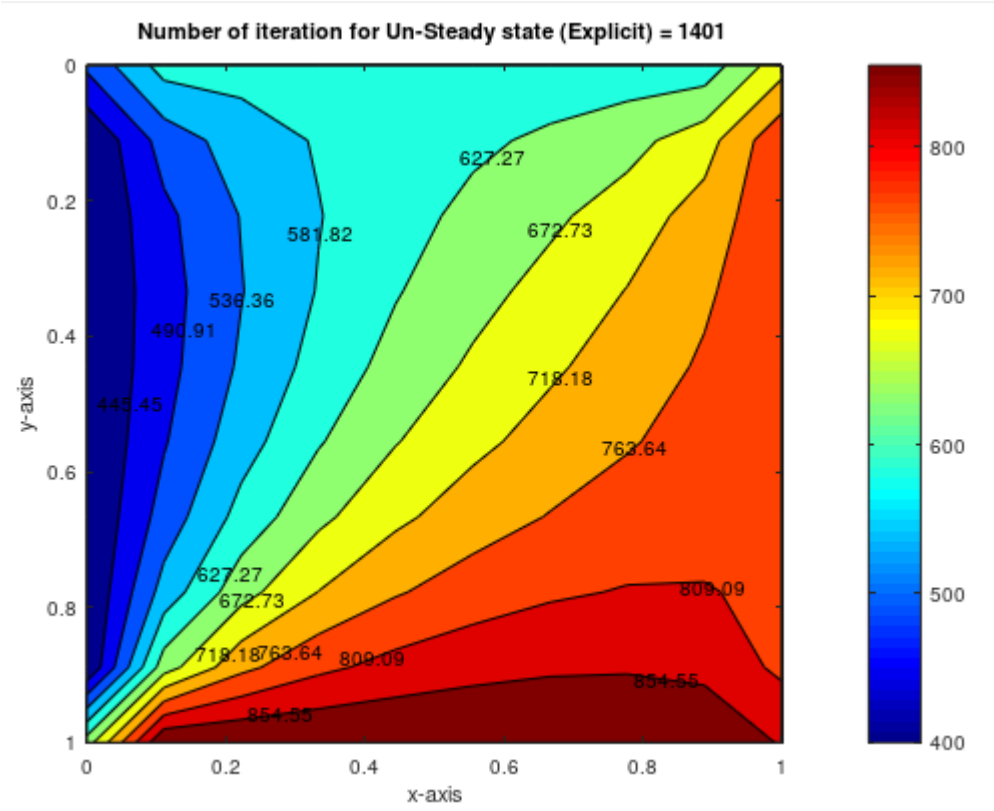


MATLAB CODE FOR UN-STEADY STATE EXPLICIT METHOD:

```
% Un-Steady state (Explicit) 2-D HEAT CONDUCTION.

close all
clear all
clc

% Inputs
nx = 10;
ny = 10;
nt = 1400;

Tx = linspace(0,1,nx);
Ty = linspace(0,1,ny);

% Defining Grid Size
dx = 1/(nx-1);
dy = 1/(ny-1);

% Boundary Condition
T = 300*ones(nx,ny);
T(1,:) = Tt = 600;
T(nx,:) = Tb = 900;
T(:,1) = Tl = 400;
T(:,ny) = Tr = 800;

T(1,1) = (Tt + Tl)/2;
T(1,ny) = (Tt + Tr)/2;
T(nx,1) = (Tb + Tl)/2;
T(nx,ny) = (Tb + Tr)/2;

Told = T;
error = 9e9;
tol = 1e-4;
alpha = 1.1;
```

```
dt = 1e-3;
k1 = (alpha*dt)/(dx^2);
k2 = (alpha*dt)/(dy^2);

% Un-Steady state - Explicit Method
explicit_iter = 1;
for k = 1:nt
  for i = 2:nx-1
      for j = 2:ny-1
        Term1 = Told(i-1,j)-2*Told(i,j)+Told(i+1,j);
        Term2 = Told(i,j-1)-2*Told(i,j)+Told(i,j+1);
        T(i,j) = Told(i,j)+k1*Term1+k2*Term2;
      endfor
    endfor
    error = max(max(abs(Told-T)));
    Told = T;
    explicit_iter = explicit_iter+1;
    contourf(Tx,Ty,T,'ShowText','on')
    colorbar
    colormap(jet)
    set(gca, 'ydir','reverse')
    xlabel('x-axis')
    ylabel('y-axis')
    title(sprintf('Number of iteration for Un-Steady state (Explicit) = %d
',explicit_iter));
    pause(0.0003)
endfor
```

OUTPUT :



Number of Iteration:

	Steady State Implicit Method Number of Iteration
Jacobian	208
Gauss seidal	112
Successive over relaxation	93
	Un-Steady State Implicit Method Number of Iteration
Jacobian	3880
Gauss seidal	3324
Successive over relaxation	2901
	Un-Steady State Explicit Method Number of Iteration
	1401

FROM THE OUTPUT:

- It is clear from the above results, The steady state condition requires comparetively less number of iteration and less computational time when compare to that of un-steady state system in order to converge a result.
- Jocobian technique requires more number of iteration then the gauss seidal and then succesive over relaxation technique in implicit method of both steady and unsteady state. **Jacobian > Gauss seidal > Successive over relaxation.**
- The successsive over relaxation technique has minimum iteration as relaxation factor, w = 1.1 which is over relaxed which helps to converge a result in a faster rate.
- Comparing the solution of unsteady state explicit and implicit method, Explicit method requires very less number of iteration and computational time to converge a result than implicit.

CONCLUSION:

Solution for the steady and unsteady state 2D heat conduction equation by using the point iterative techniques can be observed. and the output is explained based on number of iteration and computational time for the Analysis.