Solving the steady and unsteady 2D heat conduction problem

Objective: To solve the 2D heat conduction equation using a Transient solver and a Steady state solver using Iterative techniques (Jacobi,Gauss Seidel,SOR).

Given Data:

1. Assume that the domain is a unit square.

2. Assume nx = ny [Number of points along the x direction is equal to the number of points along the y direction]

3. Boundary conditions for steady and transient case

   Left:400K

   Top:600K

   Right:800K

   Bottom:900K

4. Absolute error criteria is 1e-4

Solution:

Steady State Equation:

$$\frac{d^2T}{dx^2} + \frac{d^2T}{dy^2} = 0$$

$$Tp = (1/K) * [(Tl + Tr)/\Delta x^2 + (Tt + Tb)/\Delta y^2]$$

Where,

$$K = 2 * [\Delta x^2 + \Delta y^2]/[\Delta x^2 * \Delta y^2]$$

Jacobi Method:

In the Jacobi Method, old values are used for successive iterations.

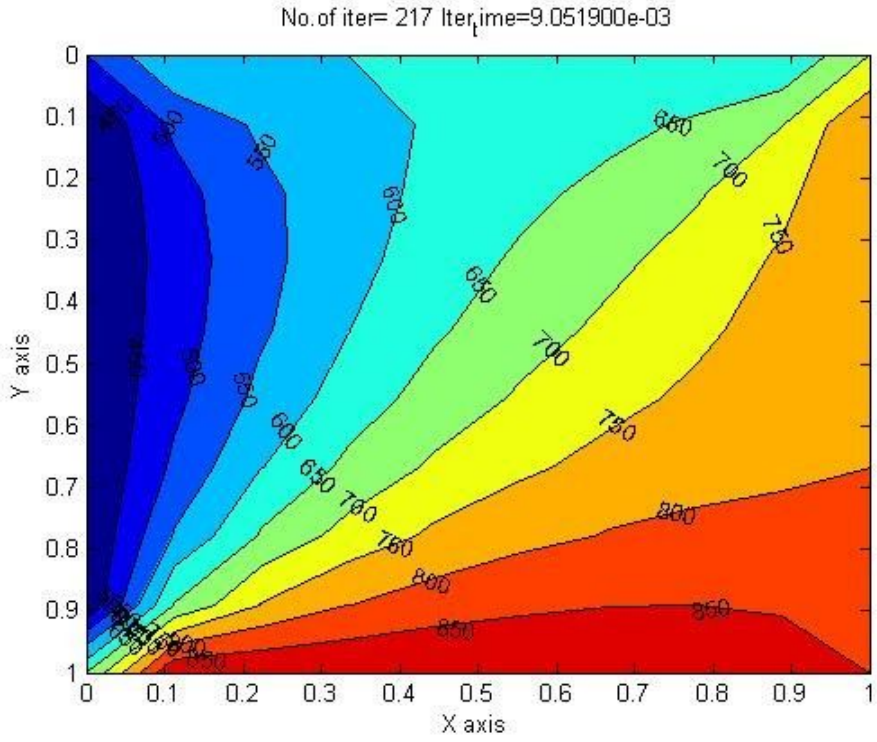
$$T1 = (1/K) * ((Told(i-1,j) + Told(i+1,j))/(dx^2));$$
$$T2 = (1/K) * ((Told(i,j-1) + Told(i,j+1))/(dy^2));$$
$$T(i,j) = T1 + T2;$$

Program:

```
clear all
close all
clc

%steady state eqn
%length of the domain is 1 (Unit square)
%Tl=400,Tr=800,Tt=600,Tb=900

%Number of nodes along x&y
nx=10;
ny=10;

%Grid spacing
x=linspace(0,1,nx);
y=linspace(0,1,ny);
[X,Y]=meshgrid(x,y); %converts a vector into matrix
dx=x(2)-x(1);
dy=y(2)-y(1);

%BCs
T=ones(nx,ny);
T(:,1)=400;   %T at left
T(:,10)=800;  %T at right
T(1,:)=600;   %T at top
T(10,:)=900;  %T at bottom
T(1,1)= 500;  %[T(:,1)+T(1,:)]/2
T(1,10)= 700; %[T(1,:)+T(:,10)]/2
T(10,1)=650;  %[T(:,1)+T(10,:)]/2
T(10,10)=850; %[T(10,:)+T(:,10)]/2

Told=T;
K=2*[(dx^2+dy^2)/(dx^2*dy^2)];
tol=1e-4;
error=9e9;
tic
jacobi_solver=1;
if jacobi_solver==1
    while error>tol
        %space loop
         for i=2:nx-1
         for j=2:ny-1
```

```
        T1=(1/K)*((Told(i-1,j)+Told(i+1,j))/(dx^2));
        T2=(1/K)*((Told(i,j-1)+Told(i,j+1))/(dy^2));
        T(i,j)=T1+T2;  %Tp=(1/K)*[((T+Tr)/(dx^2))+((Tt+Tb)/(dy^2))]
        end
        end
        error_vector=max(abs(Told-T));
        error=max(error_vector); %error single element
        Told=T;
        jacobi_solver=jacobi_solver+1;
    end
end
toc
time_taken=toc;

figure(1)
contourf(x,y,T);
[x,y]=contourf(x,y,T);
clabel(x,y);        %to label the points over the contour
xlabel('X axis');
ylabel('Y axis');
set(gca,'YDIR','reverse');
title_name=sprintf('No.of iter= %d Iter_time=%d',jacobi_solver,time_taken);
title(title_name)
```

Output:



For the input given for tol=1e-4, error=9e9, the solution converges at 217 iterations using the jacobi method and the iteration time=9.051900e-03.

Gauss Seidel Method:

In Gauss Seidel Method, old values are replaced by the newer values and used for successive iterations. The value of K remains the same as in the Jacobi method.

$$T1 = (1/K) * ((T(i-1,j) + T(i+1,j))/(dx^2))$$

$$T2 = (1/K) * ((T(i,j-1) + T(i,j+1))/(dy^2))$$

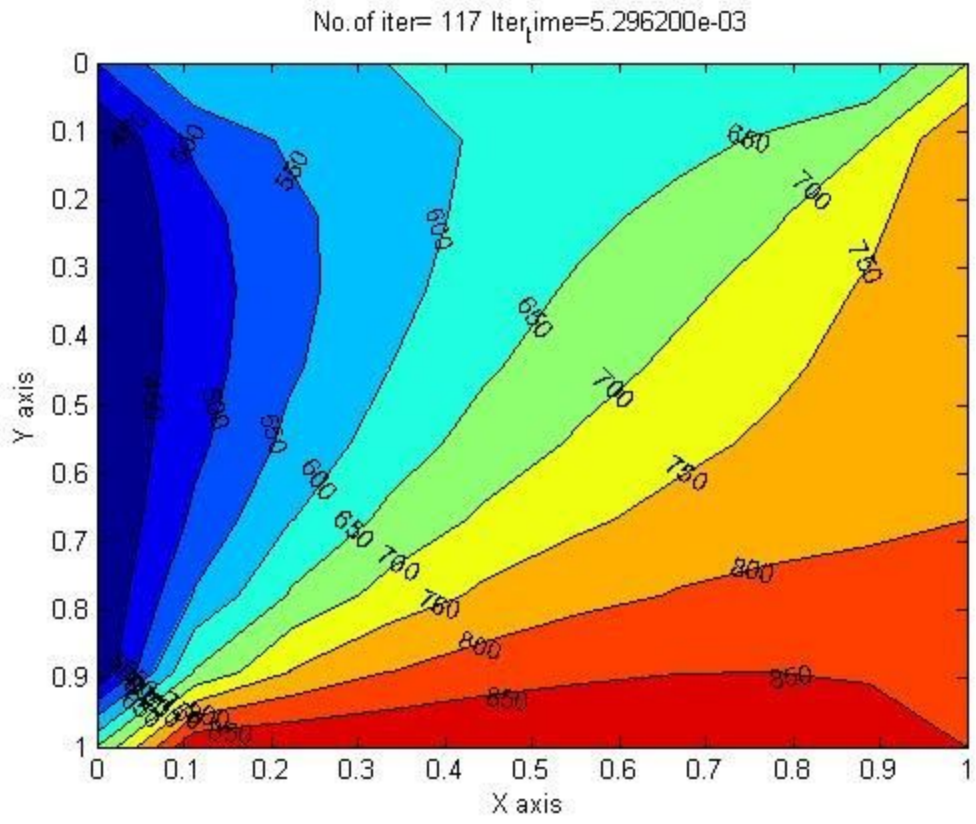
$$T(i,j) = T1 + T2;$$

Program:

```
clear all
close all
clc

%steady state eqn
%length of the domain is 1 (Unit square)
%Tl=400,Tr=800,Tt=600,Tb=900

%Number of nodes along x&y
nx=10;
ny=10;

%Grid spacing
x=linspace(0,1,nx);
y=linspace(0,1,ny);
[X,Y]=meshgrid(x,y);
dx=x(2)-x(1);
dy=y(2)-y(1);

%BCs
T=ones(nx,ny);
T(:,1)=400;   %T at left
```

```
T(:,10)=800;  %T at right
T(1,:)=600;   %T at top
T(10,:)=900;  %T at bottom
T(1,1)= 500;  %[T(:,1)+T(1,:)]/2
T(1,10)= 700; %[T(1,:)+T(:,10)]/2
T(10,1)=650;  %[T(:,1)+T(10,:)]/2
T(10,10)=850; %[T(10,:)+T(:,10)]/2

Told=T;
K=2*[(dx^2+dy^2)/(dx^2*dy^2)];
tol=1e-4;
error=9e9;
tic
GS_solver=1;
if GS_solver==1
   while error>tol
       for i=2:nx-1
       for j=2:ny-1
       T1=(1/K)*((T(i-1,j)+T(i+1,j))/(dx^2));
       T2=(1/K)*((T(i,j-1)+T(i,j+1))/(dy^2));
       T(i,j)=T1+T2; %Tp=(1/K)*[((T+Tr)/(dx^2))+((Tt+Tb)/(dy^2))]
       end
       end
       error_vector=max(abs(Told-T));
       error=max(error_vector); %error single element
       Told=T;
       GS_solver=GS_solver+1;
   end
end
time_taken=toc;

figure(1)
contourf(x,y,T);
[x,y]=contourf(x,y,T);
clabel(x,y);          %to label the points over the contour
xlabel('X axis');
ylabel('Y axis');
set(gca,'YDIR','reverse');
title_name=sprintf('No.of iter= %d Iter_time=%d',GS_solver,time_taken);
title(title_name);
```

Output:



For the input given for tol=1e-4, error=9e9, the solution converges at 117 iterations using the Gauss Seidel method and the iteration time=5.296e-03. The number of iterations is less than the jacobi method (217) which means that the solution converges faster.

Successive Over Relaxation Method:

It is a combination of both Jacobi and Gauss Seidel methods.

$$T1 = (1/K) * ((T(i-1,j) + T(i+1,j))/(dx^2))$$

$$T2 = (1/K) * ((T(i,j-1) + T(i,j+1))/(dy^2))$$

$$Tgs = T1 + T2;$$

$$T(i,j) = [Told(i,j) * (1-G)] + G * (Tgs);$$

Here, Omega(G) is a relaxation factor. If G>1, over relaxation and if G<1, under relaxation. Tgs refers to the Temperature calculated using the Gauss Seidel method.

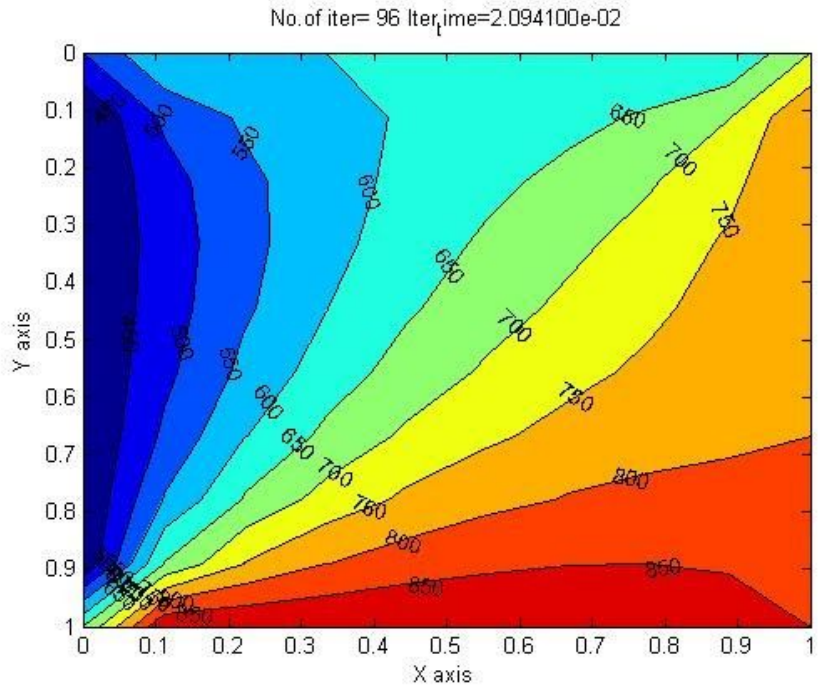
Program:

```
clear all
close all
clc

%steady state eqn
%length of the domain is 1 (Unit square)
%Tl=400,Tr=800,Tt=600,Tb=900

%Number of nodes along x&y
nx=10;
ny=10;

%Grid spacing
x=linspace(0,1,nx);
y=linspace(0,1,ny);
[X,Y]=meshgrid(x,y);
dx=x(2)-x(1);
dy=y(2)-y(1);

%BCs
T=ones(nx,ny);
```

```
T(:,1)=400;   %T at left
T(:,10)=800;  %T at right
T(1,:)=600;   %T at top
T(10,:)=900;  %T at bottom
T(1,1)= 500;  %[T(:,1)+T(1,:)]/2
T(1,10)= 700; %[T(1,:)+T(:,10)]/2
T(10,1)=650;  %[T(:,1)+T(10,:)]/2
T(10,10)=850; %[T(10,:)+T(:,10)]/2

Told=T;
K=2*[(dx^2+dy^2)/(dx^2*dy^2)];
tol=1e-4;
error=9e9;
tic
SOR_solver=1;
G=1.7; %alpha
if SOR_solver==1
   while error>tol
       for i=2:nx-1
       for j=2:ny-1
       T1=(1/K)*((T(i-1,j)+T(i+1,j))/(dx^2));
       T2=(1/K)*((T(i,j-1)+T(i,j+1))/(dy^2));
       Tgs=T1+T2; %Tp=(1/K)*[((T+Tr)/(dx^2))+((Tt+Tb)/(dy^2))]
       T(i,j)=[Told(i,j)*(1-G)]+G*(Tgs);
       end
       end
       error_vector=max(abs(Told-T));
       error=max(error_vector); %error single element
       Told=T;
       SOR_solver=SOR_solver+1;
   end
end
time_taken=toc;

figure(1)
contourf(x,y,T);
[x,y]=contourf(x,y,T);
clabel(x,y);          %to label the points over the contour
xlabel('X axis');
ylabel('Y axis');
set(gca,'YDIR','reverse');
title_name=sprintf('No.of iter= %d Iter_time=%d',SOR_solver,time_taken);
title(title_name);
```

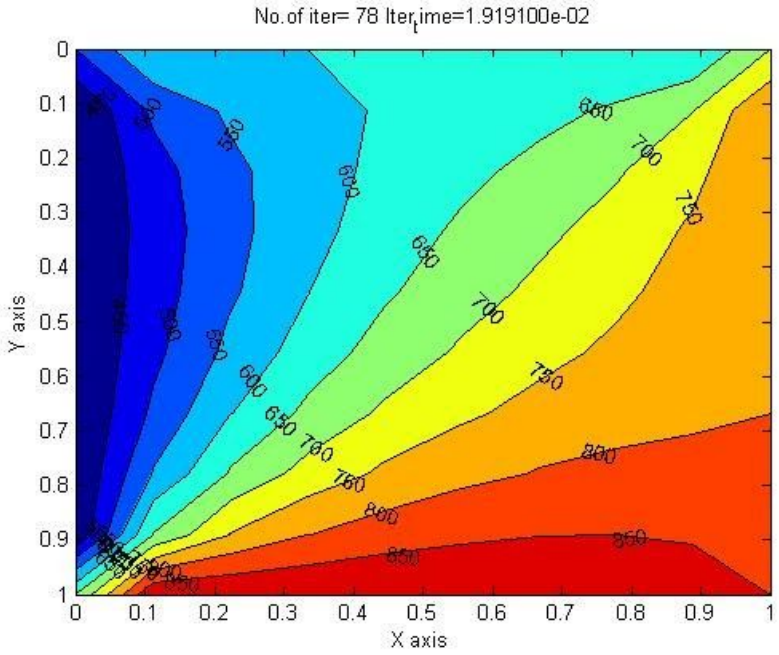
Output:

For Alpha(G)=1.1

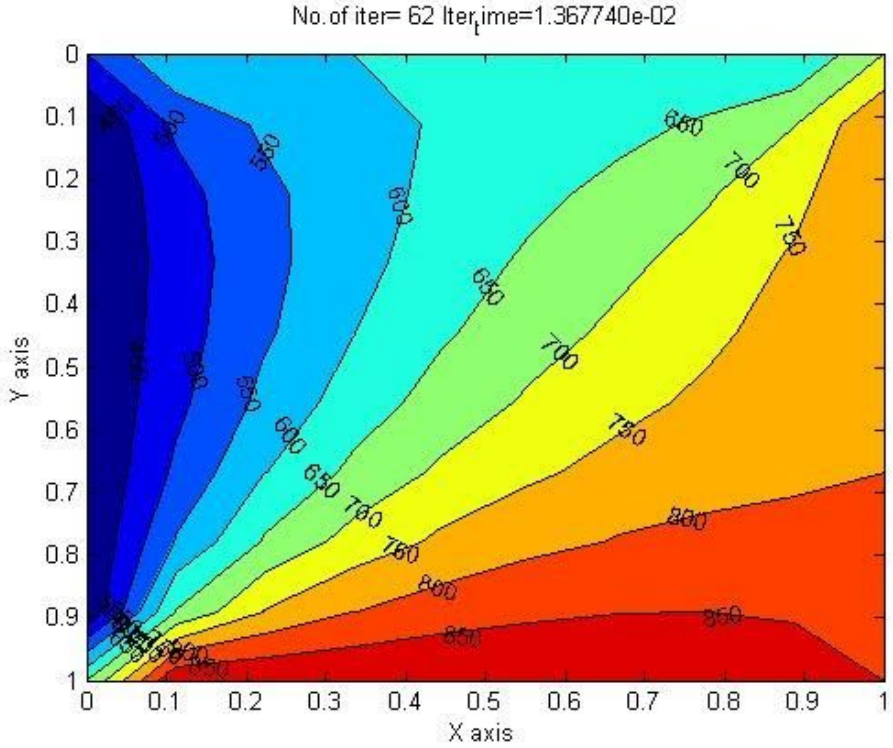


For the input given for tol=1e-4, error=9e9, Alpha(G) = 1.1, the solution converges at 96iterations using the SOR method and the iteration time=2.0941e-02. The selection of alpha is critical in this SOR method. Below are the few contours for different values of Alpha(G) such as 1.2,1.3,1.4,1.5 and 1.6. The solution converges at 96 number of iterations which is much lesser than the other two iterative solvers. The value decreases further to 30 at Alpha(G)=1.5 and the value shoots up to 38 iterations at 1.6. This shows that the selection of Alpha(G) is critical such that the result remains under control and does not overshoot to another reference value.

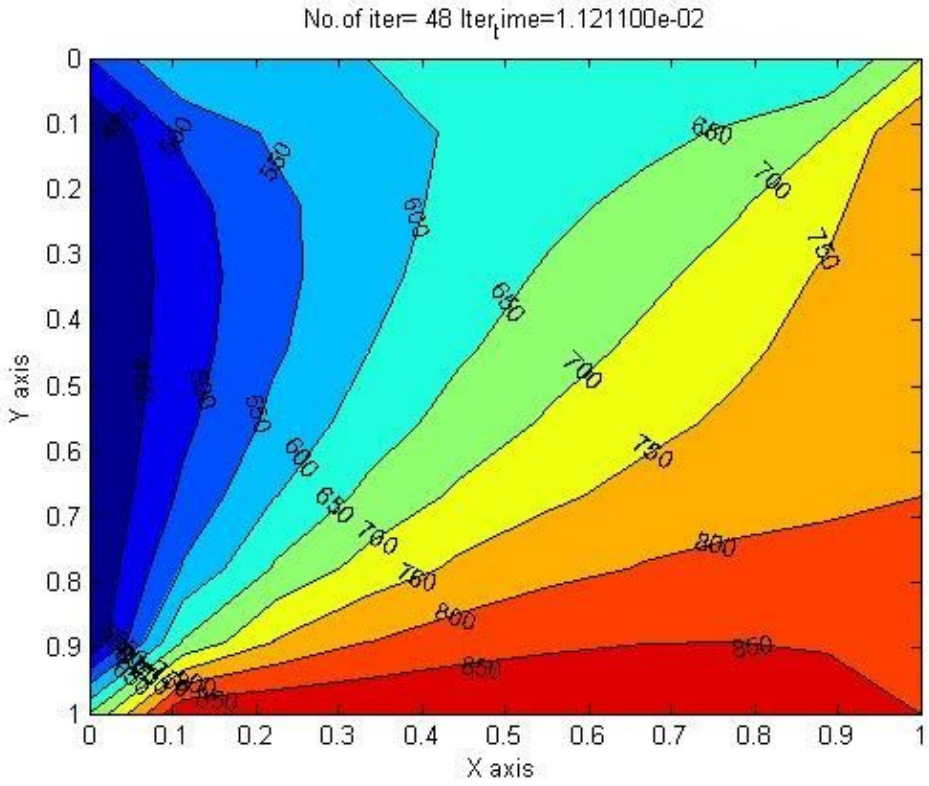
For Alpha(G)=1.2



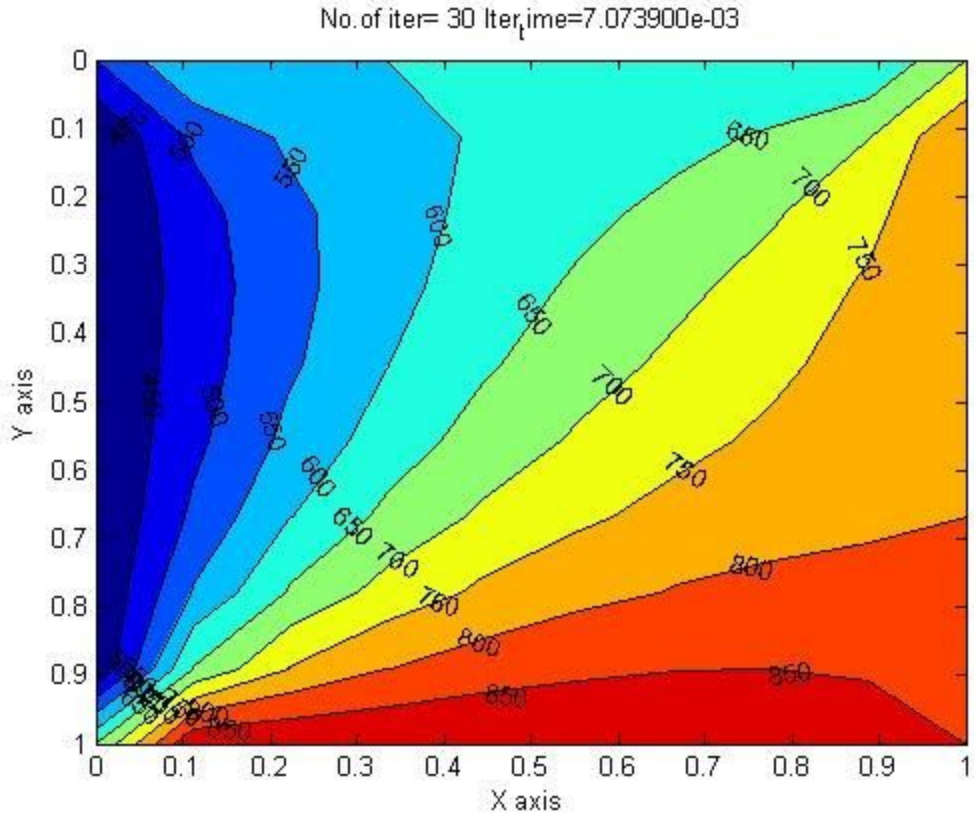
For Alpha(G)=1.3



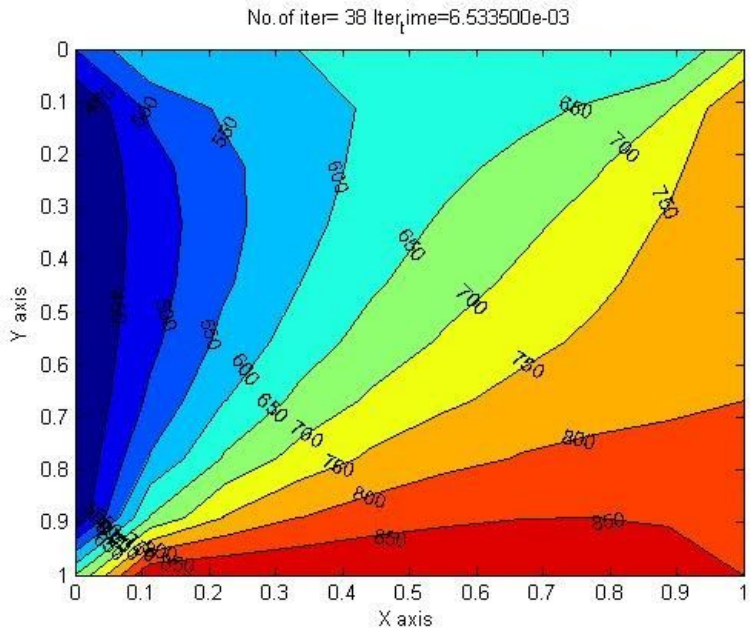
For Alpha(G)=1.4



For Alpha(G)=1.5



For Alpha(G)=1.6



As it is observed, when the Alpha value goes beyond 1.5, the number of iterations increases (from 30 to 38) which shows that the solution shoots up far away from the required point. It shows that the solution is getting unstable.

Transient / Unsteady state equation: (Explicit)

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

$$T(i,j) = Told(i,j) + [K1 * (Told(i+1,j) - 2 * Told(i,j) + Told(i-1,j))] + [K2 * (Told(i,j+1) - 2 * Told(i,j) + Told(i,j-1))]$$

Here,

$$K1 = (\alpha * dt)/(dx^2)$$
$$K2 = (\alpha * dt)/(dy^2)$$

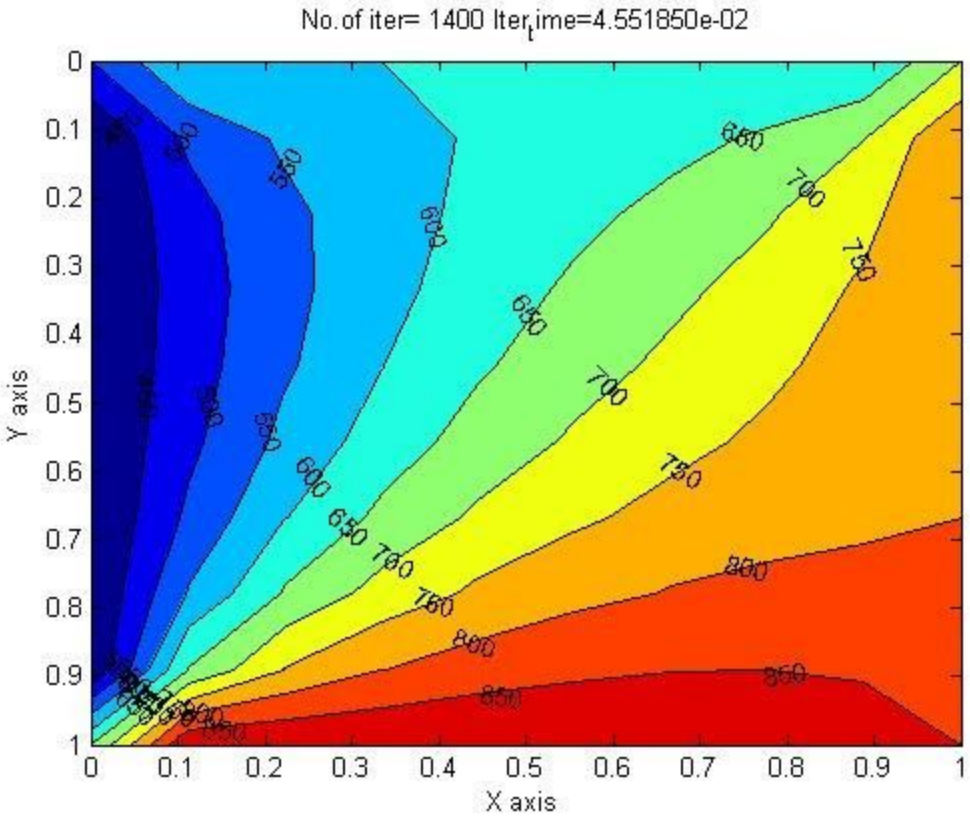
K1 and K2 are the CFL numbers along the x and y axis respectively.
The sum of two CFL numbers must not exceed 0.5 in order to get a stable solution.

Program:

```
clear all
close all
clc

%steady state eqn.
%length of the domain is 1 (Unit square)
%Tl=400,Tr=800,Tt=600,Tb=900

%Number of nodes along x&y
nx=10;
ny=10;

%Grid spacing
x=linspace(0,1,nx);
y=linspace(0,1,ny);
[X,Y]=meshgrid(x,y); %converts a vector into matrix
dx=x(2)-x(1);
dy=y(2)-y(1);

%BCs
T=ones(nx,ny);
T(:,1)=400;   %T at left
T(:,10)=800;  %T at right
T(1,:)=600;   %T at top
T(10,:)=900;  %T at bottom
T(1,1)= 500;  %[T(:,1)+T(1,:)]/2
T(1,10)= 700; %[T(1,:)+T(:,10)]/2
T(10,1)=650;  %[T(:,1)+T(10,:)]/2
T(10,10)=850; %[T(10,:)+T(:,10)]/2

Told=T;
dt=0.001;
alpha=1.4;
K1=(alpha*dt)/(dx^2);
K2=(alpha*dt)/(dy^2);

tol=1e-4;
error=9e9;
```

```
nt=1400;

tic
computation_start=0;
for k=1:nt
        while error>tol
        for i=2:nx-1
        for j=2:ny-1
            T1=Told(i,j);
            T2=K1*(Told(i+1,j)-2*Told(i,j)+Told(i-1,j));
            T3=K2*(Told(i,j+1)-2*Told(i,j)+Told(i,j-1));
            T(i,j)=T1+T2+T3;  %Tp=(1/K)*[((T+Tr)/(dx^2))+((Tt+Tb)/(dy^2))]
        end
        end
        error_vector=max(abs(Told-T));
        error=max(error_vector); %error single element
        Told=T;
        end
        computation_start=computation_start+1;
end
toc
time_taken=toc;

figure(1)
contourf(x,y,T);
[x,y]=contourf(x,y,T);
clabel(x,y);        %to label the points over the contour
xlabel('X axis');
ylabel('Y axis');
set(gca,'YDIR','reverse');
title_name=sprintf('No.of iter= %d Iter_time=%d',computation_start,time_taken);
title(title_name)
```

Output:



Transient/Unsteady state equation: (Implicit)

Jacobi Method:

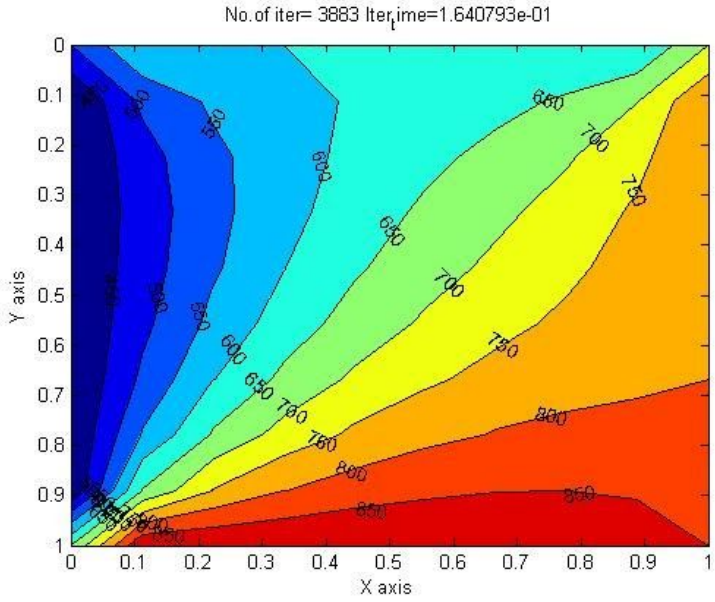
$$term1 = 1/(1 + (2 * K1) + (2 * K2));$$
$$term2 = K1 * term1;$$
$$term3 = K2 * term1;$$
$$H = (Told(i-1, j) + Told(i+1, j));$$
$$V = (Told(i, j-1) + Told(i, j+1));$$
$$T(i, j) = (T_prev(i, j) * term1) + (H * term2) + (V * term3)$$

Program:

```
clear all
close all
clc

%steady state eqn
%length of the domain is 1 (Unit square)
%Tl=400,Tr=800,Tt=600,Tb=900

%Number of nodes along x&y
nx=10;
ny=10;

%Grid spacing
x=linspace(0,1,nx);
y=linspace(0,1,ny);
[X,Y]=meshgrid(x,y); %converts a vector into matrix
dx=x(2)-x(1);
dy=y(2)-y(1);

%BCs
T=ones(nx,ny);
T(:,1)=400;   %T at left
T(:,10)=800;  %T at right
T(1,:)=600;   %T at top
T(10,:)=900;  %T at bottom
T(1,1)= 500;  %[T(:,1)+T(1,:)]/2
T(1,10)= 700; %[T(1,:)+T(:,10)]/2
T(10,1)=650;  %[T(:,1)+T(10,:)]/2
T(10,10)=850; %[T(10,:)+T(:,10)]/2

Told=T;
T_prev=Told;
dt=0.001;
alpha=1.4;
K1=(alpha*dt)/(dx^2);
K2=(alpha*dt)/(dy^2);
nt=1400;
```

```
tic
jacobi_iter=1;
if (jacobi_iter==1)
for t=1:nt
   tol=1e-4;
   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)=(T_prev(i,j)*term1)+(H*term2)+(V*term3);
       end
       end
       errori=max(abs(Told-T));
       error=max(errori); %error single element
       Told=T;
       jacobi_iter=jacobi_iter+1;
       end
      T_prev=T;
end
end
toc
time_taken=toc;

figure(1)

[x,y]=contourf(x,y,T);
clabel(x,y);       %to label the points over the contour
xlabel('X axis');
ylabel('Y axis');
set(gca,'YDIR','reverse');
title_name=sprintf('No.of iter= %d Iter_time=%d',jacobi_iter,time_taken);
title(title_name)
```

Output:



For transient conditions, the number of iterations increases drastically to 3883 using the jacobi iteration method.

Gauss Seidel Method: (Implicit Transient)

$$term1 = 1/(1 + (2 * K1) + (2 * K2));$$
$$term2 = K1 * term1;$$
$$term3 = K2 * term1;$$
$$H = (T(i-1,j) + Told(i+1,j));$$
$$V = (T(i,j-1) + Told(i,j+1));$$

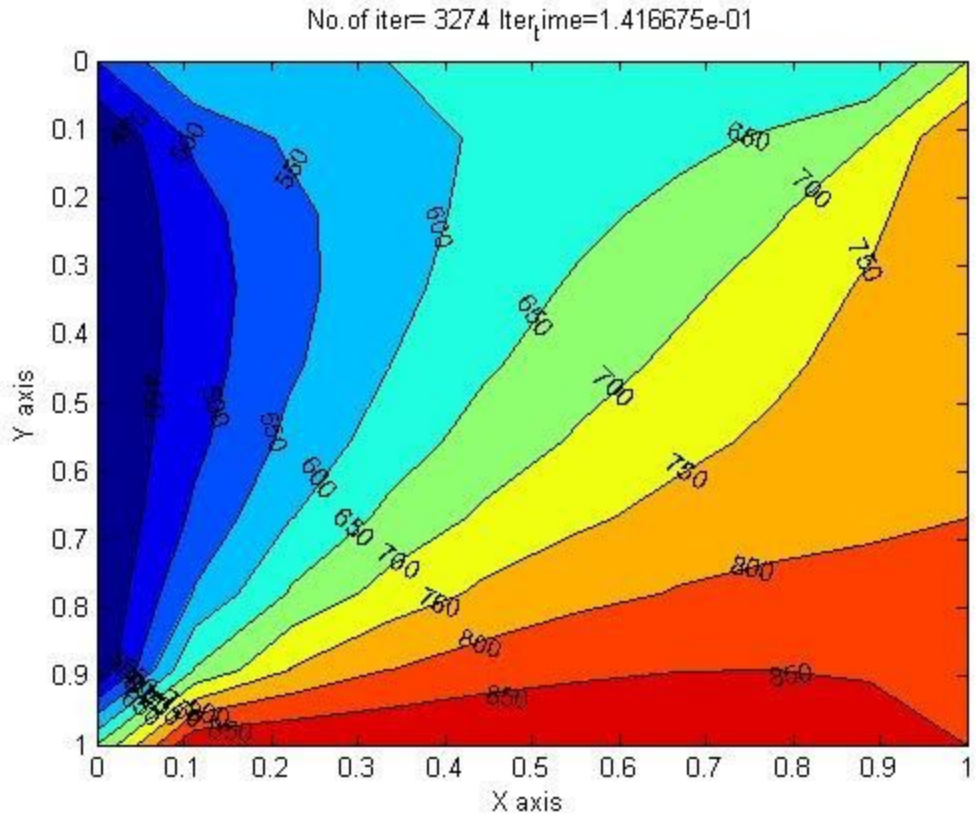
$$T(i,j) = (Tprev(i,j) * term1) + (H * term2) + (V * term3)$$

Program:

```
clear all
close all
clc

%steady state eqn
%length of the domain is 1 (Unit square)
%Tl=400,Tr=800,Tt=600,Tb=900

%Number of nodes along x&y
nx=10;
ny=10;

%Grid spacing
x=linspace(0,1,nx);
y=linspace(0,1,ny);
[X,Y]=meshgrid(x,y); %converts a vector into matrix
dx=x(2)-x(1);
dy=y(2)-y(1);

%BCs
T=ones(nx,ny);
T(:,1)=400;   %T at left
T(:,10)=800;  %T at right
T(1,:)=600;   %T at top
T(10,:)=900;  %T at bottom
T(1,1)= 500;  %[T(:,1)+T(1,:)]/2
T(1,10)= 700; %[T(1,:)+T(:,10)]/2
T(10,1)=650;  %[T(:,1)+T(10,:)]/2
T(10,10)=850; %[T(10,:)+T(:,10)]/2

Told=T;
T_prev=Told;
dt=0.001;
alpha=1.4;
K1=(alpha*dt)/(dx^2);
K2=(alpha*dt)/(dy^2);
nt=1400;
```

```
tic
GS_iter=1;
if (GS_iter==1)
for t=1:nt
   tol=1e-4;
   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)+Told(i+1,j));
          V=(T(i,j-1)+Told(i,j+1));
          T(i,j)=(T_prev(i,j)*term1)+(H*term2)+(V*term3);
       end
       end
       errori=max(abs(Told-T));
       error=max(errori); %error single element
       Told=T;
       GS_iter=GS_iter+1;
       end
      T_prev=T;
end
end
toc
time_taken=toc;

figure(1)

[x,y]=contourf(x,y,T);
clabel(x,y);        %to label the points over the contour
xlabel('X axis');
ylabel('Y axis');
set(gca,'YDIR','reverse');
title_name=sprintf('No.of iter= %d Iter_time=%d',GS_iter,time_taken);
title(title_name)
```

Output:


No.of iter= 3274 Iter_time=1.416675e-01
Y axis
X axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
600
650
700
750
800
850

Successive Over Relaxation (SOR) - (Implicit Transient)

$$term1 = 1/(1 + (2 * K1) + (2 * K2));$$
$$term2 = K1 * term1;$$
$$term3 = K2 * term1;$$
$$H = (T(i-1,j) + Told(i+1,j));$$
$$V = (T(i,j-1) + Told(i,j+1));$$

$$T(i,j) = (Tprev(i,j) * term1) + (H * term2) + (V * term3)$$

$$Tsor(i,j) = ((1 - Omega) * Told(i,j)) + (Omega * T(i,j))$$

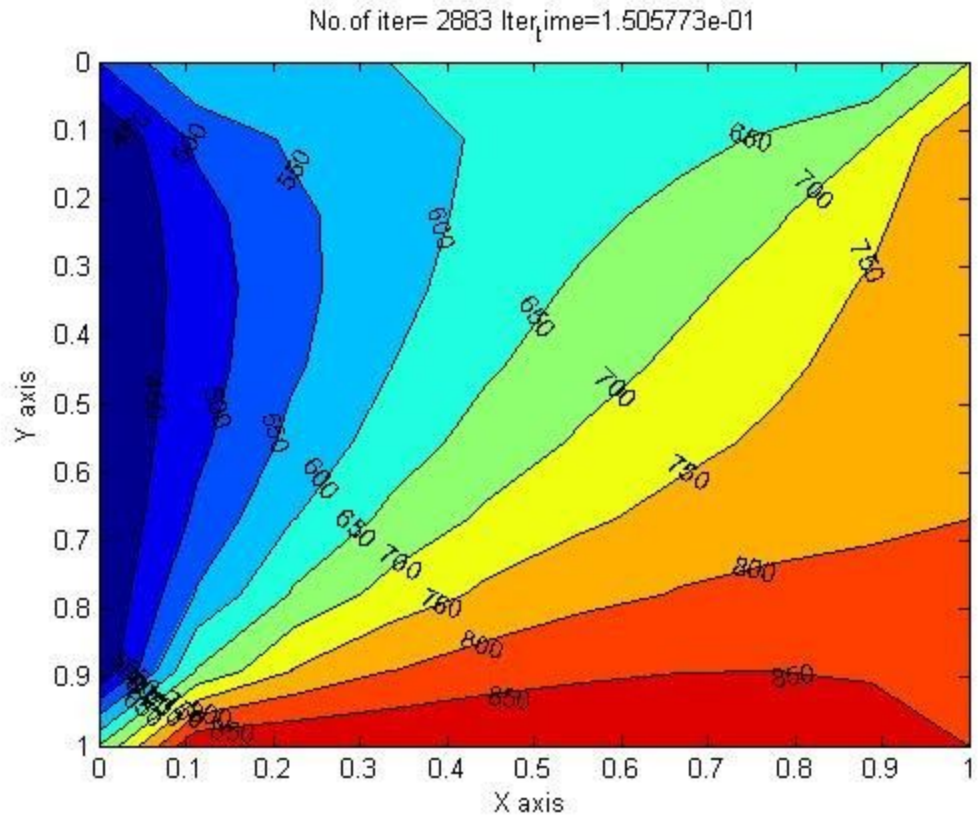
Program:

```
clear all
close all
clc

%steady state eqn
%length of the domain is 1 (Unit square)
%Tl=400,Tr=800,Tt=600,Tb=900

%Number of nodes along x&y
nx=10;
ny=10;

%Grid spacing
x=linspace(0,1,nx);
y=linspace(0,1,ny);
[X,Y]=meshgrid(x,y); %converts a vector into matrix
dx=x(2)-x(1);
dy=y(2)-y(1);
```

```
%BCs
T=ones(nx,ny);
T(:,1)=400;   %T at left
T(:,10)=800;  %T at right
T(1,:)=600;   %T at top
T(10,:)=900;  %T at bottom
T(1,1)= 500;  %[T(:,1)+T(1,:)]/2
T(1,10)= 700; %[T(1,:)+T(:,10)]/2
T(10,1)=650;  %[T(:,1)+T(10,:)]/2
T(10,10)=850; %[T(10,:)+T(:,10)]/2

Told=T;
T_prev=Told;
Tsor=T;
dt=0.001;
alpha=1.4;
Omega=1.1;
K1=(alpha*dt)/(dx^2);
K2=(alpha*dt)/(dy^2);
nt=1400;

tic
SOR_iter=1;
if (SOR_iter==1)
for t=1:nt
   tol=1e-4;
   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)+Told(i+1,j));
          V=(T(i,j-1)+Told(i,j+1));
          T(i,j)=(T_prev(i,j)*term1)+(H*term2)+(V*term3);
          Tsor(i,j)=((1-Omega)*Told(i,j))+(Omega*T(i,j));
       end
       end
       errori=max(abs(Told-T));
       error=max(errori); %error single element
       Told=Tsor;
```

```
        SOR_iter=SOR_iter+1;
        end
       T_prev=Tsor;
end
end
toc
time_taken=toc;

figure(1)

[x,y]=contourf(x,y,T);
clabel(x,y);        %to label the points over the contour
xlabel('X axis');
ylabel('Y axis');
set(gca,'YDIR','reverse');
title_name=sprintf('No.of iter= %d Iter_time=%d',SOR_iter,time_taken);
title(title_name)
```

Output:



Comparison between Explicit Transient and Implicit Transient:

The number of iterations between explicit and implicit transient schemes have a larger variations. However the time taken by the explicit is more as it does not take the neighbouring points into account for computation.

Among the Implicit schemes, SOR converges faster at 2883 iterations itself whereas both Jacobi and Gauss Siedel takes about 3000 iterations.