Steady state vs unsteady analysis

Objective: To compare the fastest steady state simulation vs fastest transient simulation.

Data: (From the previous challenge)

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:

Simulation	**Method**	**Time**	**Number of Iterations**
Steady State	Jacobi	9.051900e-03	217
	Gauss Seidel	5.296e-03	117
	SOR	2.0941e-02	96
Transient	Explicit	4.551850e-02	1400
	Jacobi	1.640793e-01	3883
	Gauss Seidel	1.416675e-01	3274
	SOR	1.505773e-01	2883

As per the previous challenge outcome(table), SOR is the fastest steady state simulation and Explicit method is the fastest transient simulation.

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,

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

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:

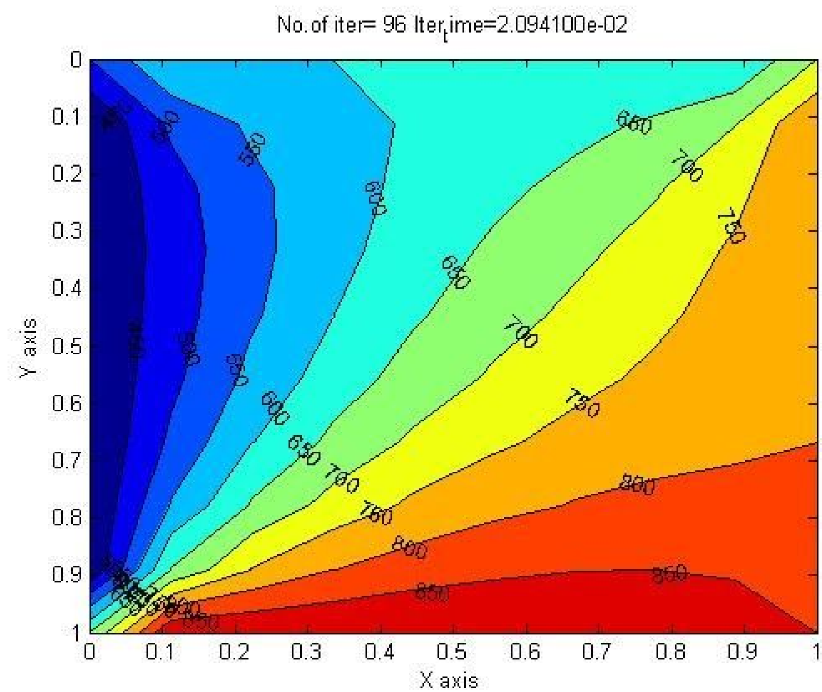
```
clear all
close all
clc

%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; %omega
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);
```

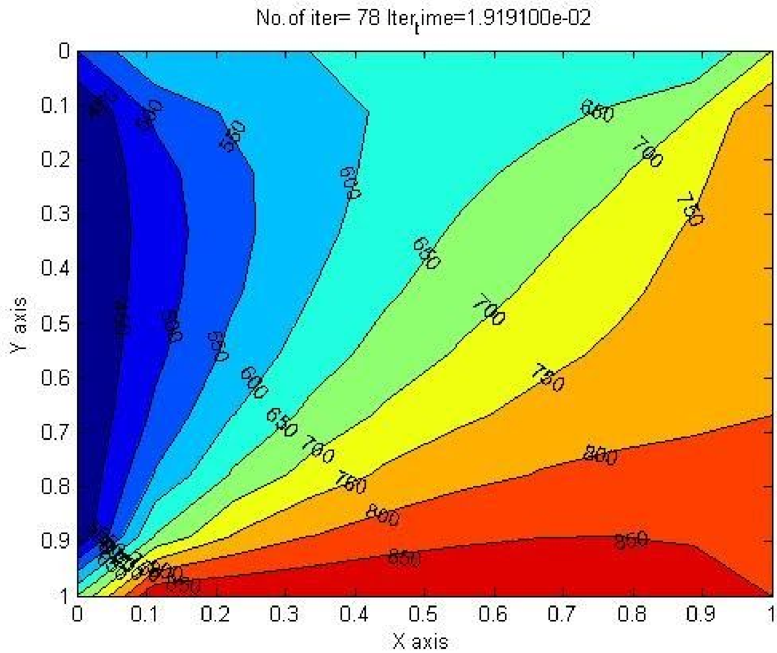
Output:

For omega(G)=1.1

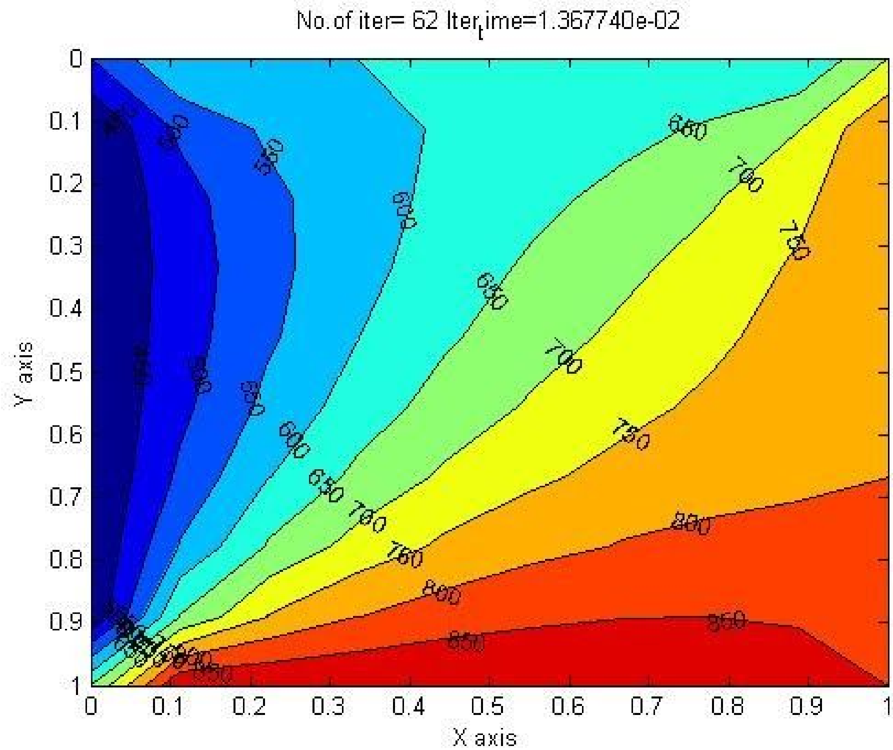


For the input given for tol=1e-4, error=9e9, omega(G) = 1.1, the solution converges at 96iterations using the SOR method and the iteration time=2.0941e-02. The selection of omega is critical in this SOR method. Below are the few contours for different values of omega(G) such as 1.2,1.3,1.4,1.5 and 1.6.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

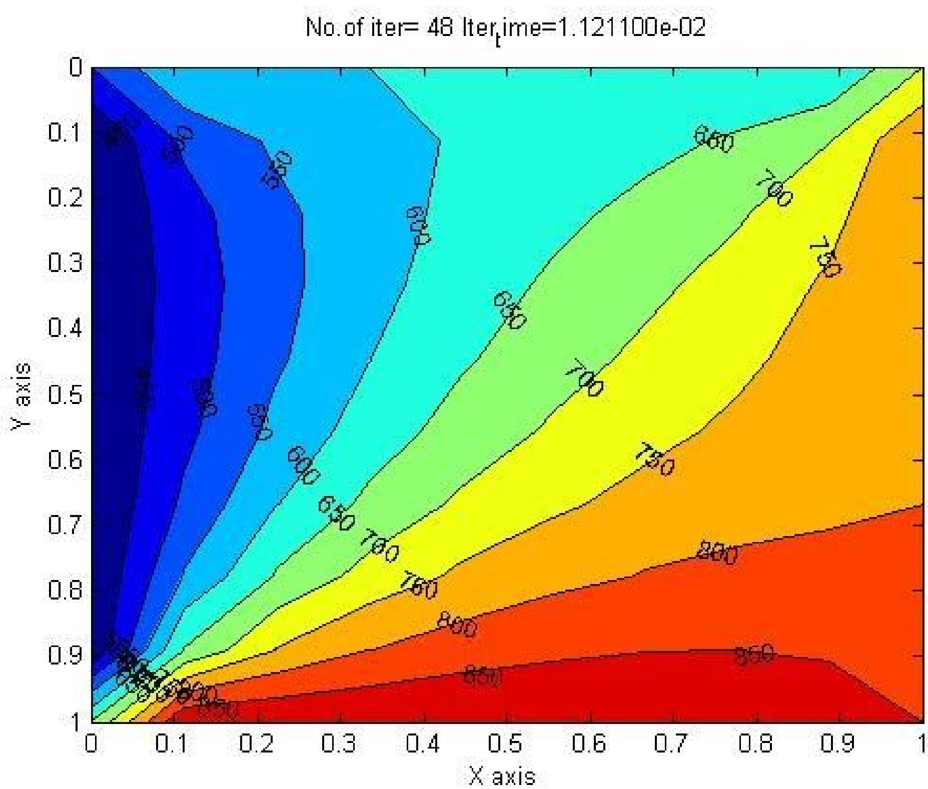
For omega(G)=1.2



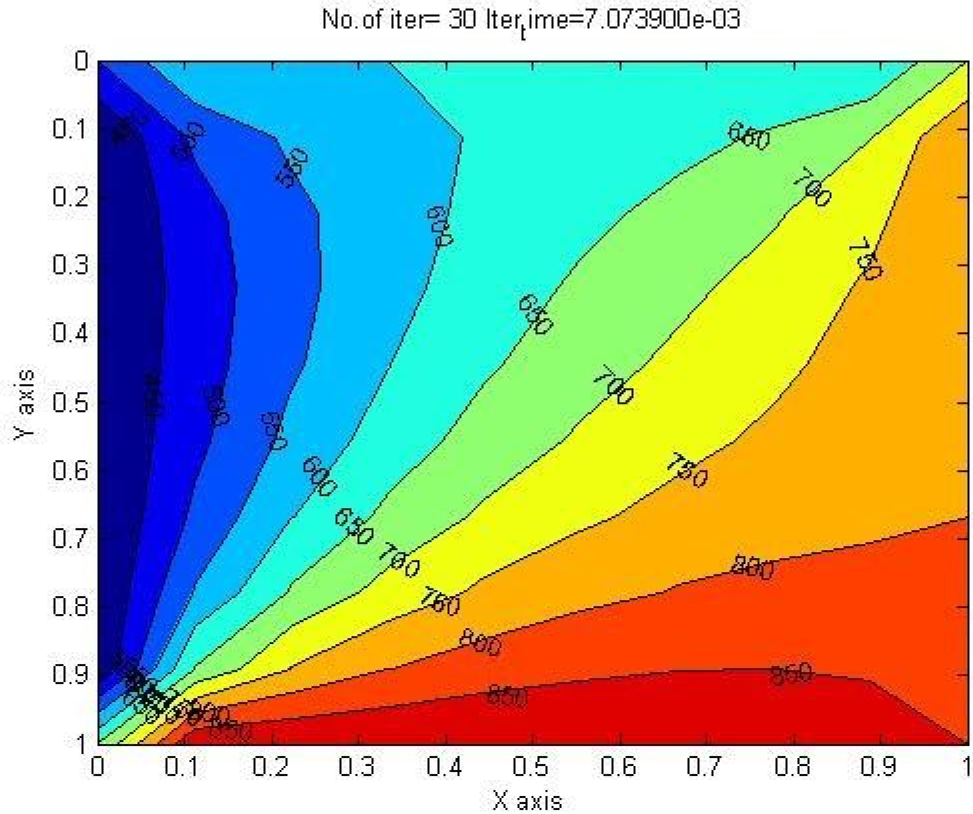
For omega(G)=1.3



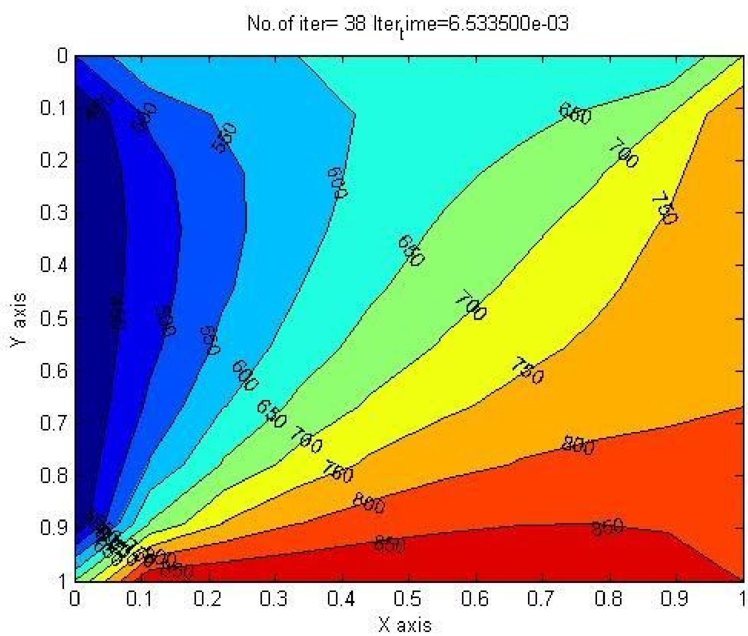
For omega(G)=1.4



For omega(G)=1.5



For omega(G)=1.6



As it is observed, when the omega 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)

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

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

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.

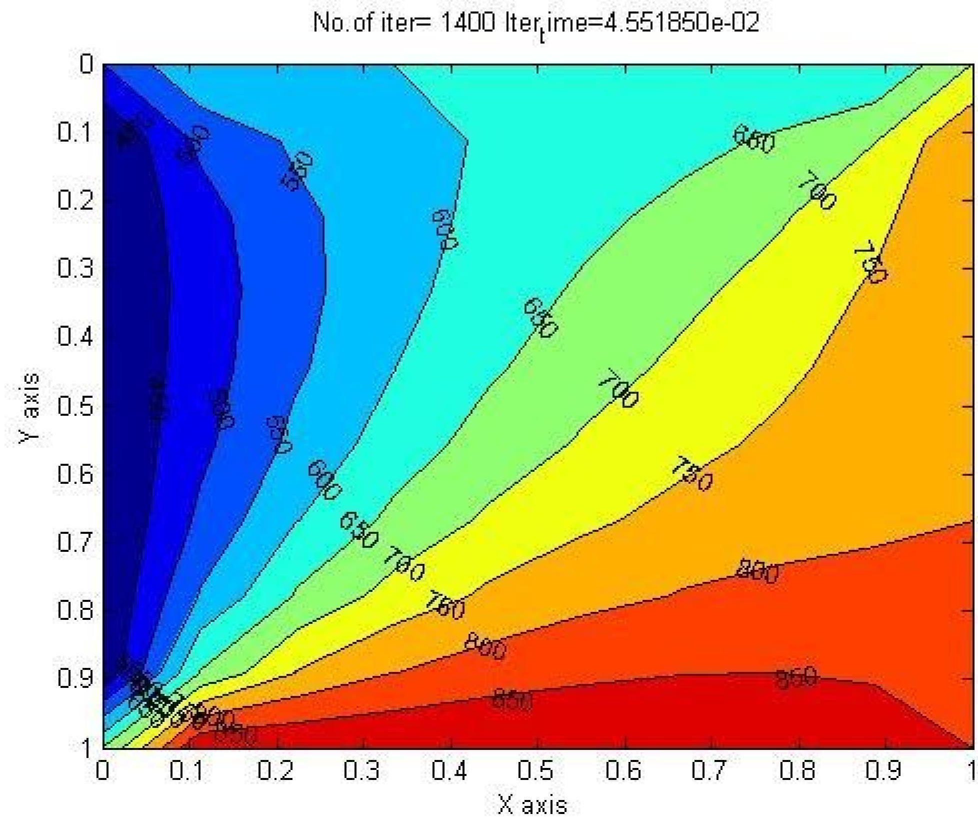
Program:

```
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;
omega=1.4;
K1=(omega*dt)/(dx^2);
K2=(omega*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:



Comparison between Steady State SOR and Transient Explicit Method:

Comparison between Steady State and Transient Method:

Steady State: The reason for SOR to converge quickly is the value of omega. Whenever omega>1,it is called as overrelaxation and when <1, it is called as under relaxation. Increasing the value of omega reduces the iterations taken. Hence SOR converges far more quickly than other schemes. However increasing the value of omega beyond a certain value (i.e., 1.5), will shoot up the solution away from the required point leading to increase in iterations from the previous state.

Transient Method:

In transient simulation, the explicit method is the fastest of all due to the fact that there is no converging criteria. However, SOR is the fastest in implicit techniques. SOR behaves similarly as it did in steady state simulation. Changing the value of time step (dt) also decreases the iterations.