```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%CFD Project 4
%Program 1
%Transient advection diffusion equation
%Central difference second order in space
%AB2 in time
%
%Author: Jithin Gopinadhan
%Date : 11/30/2015
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc
clear all

L=2*pi();           %Length of domain
N=[20 40 80 160];   %Grid sizes
Time=10;            %Maximum time for simulation
dt=0.0001;          %Delta t
M=Time/dt;          %Number of iterations in time

%Initializing error matrices for different grid sizes
Err20=zeros(N(1)-1);
Err40=zeros(N(2)-1);
Err80=zeros(N(3)-1);
%Initializing matrix to save data at x=3pi/4 and y=pi/4
Pos_data_x=zeros(5,N(4)+1);
Pos_data_y=zeros(5,N(4)+1);

%This loop runs for the different grid
%sizes specified above
for grid=1:4

    h=L/N(grid);    %Symmetric grid: delta x = delta y = h

    %Initializing temperature and its time derivative
    %for two steps in time
    Tn=zeros(N(grid)+1);     %Temperature at nth time
    Tnp1=zeros(N(grid)+1);   %Temperature at (n+1)th time
    dTn=zeros(N(grid)+1);    %dT/dt value at current time step
    dTnm1=zeros(N(grid)+1); %dT/dt value at previous time step

    %Initializing boundary conditions
    for i=1:N(grid)+1
        Tn(1,i)=1;
        Tn(N(grid)+1,i)=1;
    end

    %Time loop
    for t=0:dt:Time
        %Spatial loops
        for i=1:N(grid)+1
            for j=1:N(grid)+1
                x=h*(i-1);
                y=h*(j-1);
                u=sin(x)*cos(y); %Velocity components
                v=-cos(x)*sin(y);
```

```
                %Interior points
                if(i>1 && i<(N(grid)+1) && j>1 && j<(N(grid)+1))
                    %Evaluation of individual terms
                    dTx= (Tn(i+1,j)-Tn(i-1,j))/(2*h);
                    dTy= (Tn(i,j+1)-Tn(i,j-1))/(2*h);
                    d2Tx= (Tn(i+1,j)- 2*Tn(i,j)+Tn(i-1,j))/(h^2);
                    d2Ty= (Tn(i,j+1)- 2*Tn(i,j)+Tn(i,j-1))/(h^2);
                    %Evaluation of dT/dt
                    dTn(i,j) = -u*dTx - v*dTy + d2Tx + d2Ty;
                end
            end
        end %End of spatial loops

        %Using first order for t=0
        if(t==0)
            Tnp1 = Tn + (dTn *dt);
        end
        %Using AB2 for all other
        if(t>0)
            Tnp1 = Tn + (0.5*dt)*(3* dTn - dTnm1);
        end

        %Assigning T and dT/dt values to correct variable
        %before stepping forward in time
        Tn=Tnp1;
        dTnm1=dTn;

        %Plotting temperature profile for required times
        if(t==2.5 || t==5 || t==7.5 || t==10)
            [X,Y] = meshgrid(0:h:2*pi());
            figure,mesh(X,Y,Tnp1)
            s1=num2str(t);
            s2=num2str(N(grid));
            title(['Temperature distribution at t=',s1,'(',s2,'x',s2,')']
,'FontSize',10)
            xlabel('y');
            ylabel('x');
        end

        %Evaluation of T along x=3*pi/4 and y=pi/4
        %for required times at highest grid resolution
        if(grid==4 && (t==2.5 || t==5 || t==7.5 || t==10))
            T80=Tn;
            i_pos=(3*N(grid) +8)/8; %3pi/4
            j_pos=(N(grid) +8)/8;   %pi/4
            x_pos=3*pi()/4;
            y_pos=pi()/4;
            %Finding index of next highest nodes
            i_ceil=ceil(i_pos);
            j_ceil=ceil((j_pos));
            %Finding position of adjacent nodes
            x1= (i_ceil-2)*h;
            x2= (i_ceil-1)*h;
            y1= (j_ceil-2)*h;
            y2= (j_ceil-1)*h;
```

```
            for q=1:N(grid)+1
                %Linear interpolation of temperature based on temperature
                %values of adjacent nodes
                T_xpos=T80(i_ceil-1,q)+ (T80(i_ceil+1,q)-T80(i_ceil-
1,q))*((x_pos-x1)/h);
                T_ypos=T80(q,j_ceil-1)+ (T80(q,j_ceil+1)-T80(q,j_ceil-
1))*((y_pos-y1)/h);
                len=(q-1)*h;
                %Saving data for plotting later
                Pos_data_x(q,1)=len;
                Pos_data_x(q,(t/2.5)+1)=T_xpos;
                Pos_data_y(q,1)=len;
                Pos_data_y(q,(t/2.5)+1)=T_ypos;

            end
        end

    end %End of time loop

    %Saving values at t=10 for grid refinement study
    if(grid==1)
        T20=Tn;
    end
    if(grid==2)
        T40=Tn;
    end
    if(grid==3)
        T80=Tn;
    end
    if(grid==4)
        T160=Tn;
    end
end

%Evaluation of errors of 3 lower grid refinements against
%highest grid refinemnt assuming it to be exact
for i=1:N(1)-1
    for j=1:N(1)-1
        Err20(i,j)=T160((1+i*8),(1+j*8))-T20(i+1,j+1);
    end
end
for i=1:N(2)-1
    for j=1:N(2)-1
        Err40(i,j)=T160((1+i*4),(1+j*4))-T40(i+1,j+1);
    end
end
for i=1:N(3)-1
    for j=1:N(3)-1
        Err80(i,j)=T160((1+i*2),(1+j*2))-T80(i+1,j+1);
    end
end

%Evaluation of L2 norm for errors
E20=norm(Err20)/(N(1)-1); E40=norm(Err40)/(N(2)-1); E80=norm(Err80)/(N(3)-1);
%Evaluation of order of convergence
```

```
order=log((E80-E40)/(E40-E20))/log(0.5)

%Plotting temperature profiles along x=3pi/4
figure,plot(Pos_data_x(:,1),Pos_data_x(:,2),'.-')
hold on
plot(Pos_data_x(:,1),Pos_data_x(:,3),'.-')
hold on
plot(Pos_data_x(:,1),Pos_data_x(:,4),'.-')
hold on
plot(Pos_data_x(:,1),Pos_data_x(:,5),'.-')
title('Temperature at x= 3pi/4','FontSize',12)
xlabel('y');
ylabel('Temperature');
legend('t=2.5','t=5','t=7.5','t=10')

%Plotting temperature profiles along y=pi/4
figure,plot(Pos_data_y(:,1),Pos_data_y(:,2),'.-')
hold on
plot(Pos_data_y(:,1),Pos_data_y(:,3),'.-')
hold on
plot(Pos_data_y(:,1),Pos_data_y(:,4),'.-')
hold on
plot(Pos_data_y(:,1),Pos_data_y(:,5),'.-')
title('Temperature at y= pi/4','FontSize',12)
xlabel('x');
ylabel('Temperature');
legend('t=2.5','t=5','t=7.5','t=10')
```