```matlab
% Simulating the 1-D Diffusion equation (Fourier's equation) by the
...Finite Difference Method(a time march)
% Numerical scheme used is a first order upwind in time and a second order
...central difference in space (both Implicit and Explicit)

%%
%Specifying Parameters
nx=100;               %Number of steps in space(x)
nt=300;               %Number of time steps
dt=0.1;              %Width of each time step
dx=2/(nx-1);         %Width of space step
x=0:dx:2;            %Range of x (0,2) and specifying the grid points
u=zeros(nx,1);       %Preallocating u
un=zeros(nx,1);      %Preallocating un
vis=0.01;            %Diffusion coefficient/viscosity
beta=vis*dt/(dx*dx); %Stability criterion (0<=beta<=0.5, for explicit)
UL=0;                %Left Dirichlet B.C
UR=0;                %Right Dirichlet B.C
UnL=0;               %Left Neumann B.C (du/dn=UnL)
UnR=0;               %Right Neumann B.C (du/dn=UnR)


Cond=1;  %1 if dirichlet, 0 if Neumann

%%
%Initial Conditions: A square wave
for i=1:nx
    if ((0.75<=x(i))&&(x(i)<=1.25))
        u(i)=2;
    else
        u(i)=1;
    end
end

%Source
source=zeros(nx-2,nt+1);

for jt=1:nt+1
for jx=1:nx-2

source(jx,jt)=0.05*exp(-((x(jx+1)-1.3)/0.5).^2);
end
end





%%
%B.C vector
bc=zeros(nx-2,1);
if (Cond>0)
bc(1)=vis*dt*UL/dx^2; bc(nx-2)=vis*dt*UR/dx^2;  %Dirichlet B.Cs
end
if (Cond==0)
bc(1)=-UnL*vis*dt/dx; bc(nx-2)=UnR*vis*dt/dx;  %Neumann B.Cs
end
%Calculating the coefficient matrix for the implicit scheme
E=sparse(2:nx-2,1:nx-3,1,nx-2,nx-2);
if (Cond>0)
A=E+E'-2*speye(nx-2);        %Dirichlet B.Cs
end
```

```
if (Cond==0)
A(1,1)=-1; A(nx-2,nx-2)=-1; %Neumann B.Cs
end
D=speye(nx-2)-(vis*dt/dx^2)*A;

%%
%Calculating the velocity profile for each time step
i=2:nx-1;
for it=0:nt
pause(0.1)
    un=u;
    h=plot(x,u,'k','linewidth',2);       %plotting the velocity profile
    axis([0 2 0 3])
    title({['1-D Diffusion with \nu =',num2str(vis),' and \beta = ',num2str(beta)];↙
['time(\itt) = ',num2str(dt*it)]})
    xlabel('Spatial co-ordinate (x) \rightarrow')
    ylabel('Transport property profile (u) \rightarrow')
    drawnow;
    refreshdata(h)
    %Uncomment as necessary
    %-------------------
    %Implicit solution

    U=un;U(1)=[];U(end)=[];
    U=U+bc+dt*source(:,jt);
    U=D\U;

if (Cond>0)
    u=[UL;U;UR];
end

                    if (Cond==0)
 u=[U(1)-UnL*dx;U;U(end)+UnR*dx]; %Neumann
end
    %}
    %-------------------
    %Explicit method with F.D in time and C.D in space
    %{
    u(i)=un(i)+(vis*dt*(un(i+1)-2*un(i)+un(i-1))/(dx*dx));
    %}
end
```