ME630A

Computational Fluid Dynamics
Indian Institute of Technology, Kanpur

Submitted by
RIKESH SHARMA (180606)

Assignment 3
Solution

Date of Submission: 5^{th} Nov 2021

Question :

Given Burger equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0$$

we can write it as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \frac{1}{Re}\frac{\partial^2 u}{\partial x^2} = 0$$

Where Re=$\frac{1}{\nu}$

At t=0 the wave is located at x=0. Hence, the initial condition becomes

$u_o(x) = \overline{u}(x,0) = 1.0$ for $-x_{max} \leq x \leq 0$

$u_o(x) = \overline{u}(x,0) = 0.0$ for $0 < x \leq x_{max}$

The following boundary conditions are applied at $x = -x_{max}$ and $x = +x_{max}$

$u(-x_{max}, t) = 1.0$ and $u(x_{xmax}, t) = 0.0$ for $t > 0$

For the given combination of initial and boundary conditions the exact solution of Burger equation is given as

$$\overline{u} = \frac{\int\limits_{-\infty}^{\infty} \frac{(x-\epsilon)}{t} e^{-0.5ReG} d\epsilon}{\int\limits_{-\infty}^{\infty} e^{-0.5ReG} d\epsilon}$$

where

$$G(\zeta; x, t) = \int\limits_{0}^{\epsilon} u_o(\zeta) d\zeta + 0.5\frac{(x-\epsilon)^2}{t}$$

We need to solve the Burger equation through explicit and implicit scheme for Re=10 and Re=50

The 1D Burger's equation can be written as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \frac{1}{Re}\frac{\partial^2 u}{\partial x^2} = 0 \qquad \text{where } Re = \frac{1}{\nu}$$

$$\Rightarrow \frac{\partial u}{\partial t} + \frac{\partial u^2/2}{\partial x} - \frac{1}{Re}\frac{\partial^2 u}{\partial x^2} = 0 \qquad \text{Let } F(u) = u^2/2$$

using explicit scheme we can write

(FTFS)

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + \frac{F(u_{j+1}^n) - F(u_j^n)}{\Delta x} - \frac{1}{Re}\frac{(u_{j+1}^n - 2u_j^n + u_{j-1}^n)}{\Delta x^2} = 0$$

$$\Rightarrow \boxed{u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}\left(F(u_{j+1}^n) - F(u_j^n)\right) + \frac{\nu \Delta t}{\Delta x^2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right)}$$

Similarly

(FTBS)

$$\boxed{u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}\left(F(u_j^n) - F(u_{j-1}^n)\right) + \frac{\nu \Delta t}{\Delta x^2}\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right)}$$

Using Lax - Friedrich we can write

$$\boxed{u_j^{n+1} = \frac{1}{2}\{u_{j+1}^n + u_{j-1}^n\} - \frac{\Delta t\{F_{j+1}^n - F_{j-1}^n\}}{2\Delta x} + \frac{\nu\Delta t}{\Delta x^2}\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right)}$$

Deriving Implicit Scheme for the Burger's equation we get. (Crank nicolson)

$$\frac{\partial u}{\partial t} + \frac{\partial F(u)}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0 \qquad F(u) = \frac{u^2}{2}$$

descretizing we get

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + \frac{1}{2}\left\{ \frac{F_{j+1}^{n+1} - F_j^{n+1}}{\Delta x} + \frac{F_{j+1}^n - F_j^n}{\Delta x} \right\} - \frac{\nu}{2}\left\{ \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2} + \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} \right\} = 0 \quad \text{— (1)}$$

$$F_j^{n+1} = F_j^n + \Delta t\left(\frac{\partial F}{\partial t}\right)_j^n + \frac{\Delta t^2}{2!}\left(\frac{\partial^2 F}{\partial t^2}\right)_j^n + \dots$$

$$\Rightarrow F_j^{n+1} = F_j^n + \Delta t\left(\frac{\partial F}{\partial u}\frac{\partial u}{\partial t}\right)_j^n + O(\Delta t^2)$$

$$\Rightarrow F_j^{n+1} = F_j^n + \Delta t \cdot u_j^n \cdot \left(\frac{\partial u}{\partial t}\right)_j^n + O(\Delta t^2)$$

$$\Rightarrow F_j^{n+1} = F_j^n + u_j^n\left(u_j^{n+1} - u_j^n\right) + O(\Delta t^2) \quad \text{— (a)}$$

Similarly

$$F_{j+1}^{n+1} = F_{j+1}^n + u_{j+1}^n\left(u_{j+1}^{n+1} - u_{j+1}^n\right) + O(\Delta t^2) \quad \text{— (b)}$$

using (a) and (b) in eq^n (1) can be written as

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + \frac{1}{2\Delta x}\left\{ F_{j+1}^n + u_{j+1}^n\left(u_{j+1}^{n+1} - u_{j+1}^n\right) + F_{j+1}^n - F_j^n - u_j^n\left(u_j^{n+1} - u_j^n\right) - F_j^n \right\}$$
$$- \frac{\nu}{2\Delta x^2}\left\{ u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1} + u_{j+1}^n - 2u_j^n + u_{j-1}^n \right\} = 0$$

$$\frac{u_j^{n+1}-u_j^{n}}{\Delta t}+\frac{1}{2\Delta x}\left\{u_{j+1}^{n}u_{j+1}^{n+1}-u_j^{n}u_j^{n+1}\right\}-\frac{\nu}{2\Delta x^2}\left\{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}+u_{j+1}^{n}-2u_j^{n}+u_{j-1}^{n}\right\}=0$$

$$\Rightarrow u_{j-1}^{n+1}\left(-\frac{\nu}{2\Delta x^2}\right)+u_j^{n+1}\left(\frac{1}{\Delta t}-\frac{u_j^n}{2\Delta x}+\frac{\nu}{\Delta x^2}\right)+u_{j+1}^{n+1}\left\{\frac{u_{j+1}^n}{2\Delta x}-\frac{\nu}{2\Delta x^2}\right\}$$

$$=\frac{u_j^n}{\Delta t}+\frac{\nu}{2\Delta x^2}\left\{u_{j+1}^n-2u_j^n+u_{j-1}^n\right\}$$

$$\Rightarrow \boxed{u_{j-1}^{n+1}\left\{\frac{-0.5\nu\Delta t}{\Delta x^2}\right\}+u_j^{n+1}\left\{1-\frac{u_j^n\Delta t}{2\Delta x}+\frac{\nu\Delta t}{\Delta x}\right\}+u_{j+1}^{n+1}\left\{\frac{u_{j+1}^n\Delta t}{2\Delta x}-\frac{0.5\nu\Delta t}{\Delta x^2}\right\}=\frac{0.5\nu\Delta t}{\Delta x^2}\left\{u_{j+1}^n-2u_j^n+u_{j-1}^n\right\}+u_j^n}$$

Using boundary condition

$j=1$ $\quad u_1^n = 1.0$ $\quad \forall n$ time steps.

$j=N_g$ $\quad u_{N_g}^n = 0.0$

hence at each time n we get following.

$j=1 \quad u_1^{n+1} = 1.0$

$j=2 \quad u_1^{n+1}\left\{-\frac{0.5\nu\Delta t}{\Delta x^2}\right\} + u_2^{n+1}\left\{1-\frac{u_2^n\Delta t}{2\Delta x}+\frac{\nu\Delta t}{\Delta x}\right\} + u_3^{n+1}\left\{\frac{u_3^n\Delta t}{2\Delta x}-\frac{0.5\nu\Delta t}{\Delta x^2}\right\}$

$$= \frac{0.5\nu\Delta t}{\Delta x^2}\left\{u_3^n - 2u_2^n + u_1^n\right\} + u_2^n$$

$\vdots$

$j=Ng-1 \quad u_{Ng-2}^{n+1}\left\{-\frac{0.5\nu\Delta t}{\Delta x^2}\right\} + u_{Ng-1}^{n+1}\left\{1-\frac{u_2^n\Delta t}{2\Delta x}+\frac{\nu\Delta t}{\Delta x}\right\} + u_{Ng}^{n+1}\left\{\frac{u_{Ng}^n\Delta t}{2\Delta x}-\frac{0.5\nu\Delta t}{\Delta x^2}\right\}$

$$= \frac{0.5\nu\Delta t}{\Delta x^2}\left\{u_{Ng}^n - 2u_{Ng-1}^n + u_{Ng-2}^n\right\} + u_{Ng-1}^n$$

hence we get a tridiagonal system.
at each time n

$$\begin{bmatrix} b & c & & \\ a & \ddots & \ddots & \\ & \ddots & \ddots & \ddots \\ & & \ddots & \ddots \end{bmatrix} \begin{bmatrix} | \\ u_j^{n+1} \\ | \end{bmatrix} = \begin{bmatrix} | \\ d \\ | \end{bmatrix}$$

Solving the above using Tridiagonal Matrix Algorithm at each time step
we get "$\underline{\underline{u}}$"

EXACT SOLUTION

We have exact solution $\bar{u}$ as given by.

$$\bar{u} = \frac{\int_{-\infty}^{\infty} \left[\frac{(x-\varepsilon)}{t}\right] e^{-0.5 Re G} \, d\varepsilon}{\int_{-\infty}^{\infty} e^{-0.5 Re G} \, d\varepsilon}$$

where $G(\varepsilon; x, t) = \int_0^{\varepsilon} u_0(\xi')\, d\xi' + 0.5 \frac{(x-\xi)}{t}$

Since $u_0(\xi') = 1$ for $-\infty \leq \xi' \leq 0$

$= 0$ for $0 < \xi' \leq \infty$

$$\therefore G(\xi; x, t) = \begin{cases} \int_0^{\xi} 1 \cdot d\xi' + 0.5 \frac{(x-\xi)^2}{t} & -\infty \leq \xi \leq 0 \\ \int_0^{\xi} 0 \cdot d\xi' + 0.5 \frac{(x-\xi)^2}{t} & 0 < \xi < \infty \end{cases}$$

$$\boxed{G(\varepsilon; x, t) = \begin{cases} \varepsilon + 0.5 \frac{(x-\varepsilon)^2}{t} & -\infty < \varepsilon \leq 0 \\ 0.5 \frac{(x-\varepsilon)^2}{t} & 0 < \varepsilon < \infty \end{cases}}$$

$$\therefore \bar{u} = \frac{\int_{-\infty}^{\infty} \left(\frac{x-\xi}{t}\right) e^{-0.5 Re G} \, d\xi}{\int_{-\infty}^{\infty} e^{-0.5 Re G} \, d\xi}$$

$$\bar{u} = \frac{\int_{-\infty}^{0} \left(\frac{x-\xi}{t}\right) e^{-0.5 Re G} \, d\xi + \int_{0}^{\infty} \left(\frac{x-\xi}{t}\right) e^{-0.5 Re G} \, d\xi}{\int_{-\infty}^{0} e^{-0.5 Re G} \, d\xi + \int_{0}^{\infty} e^{-0.5 Re G} \, d\xi}$$

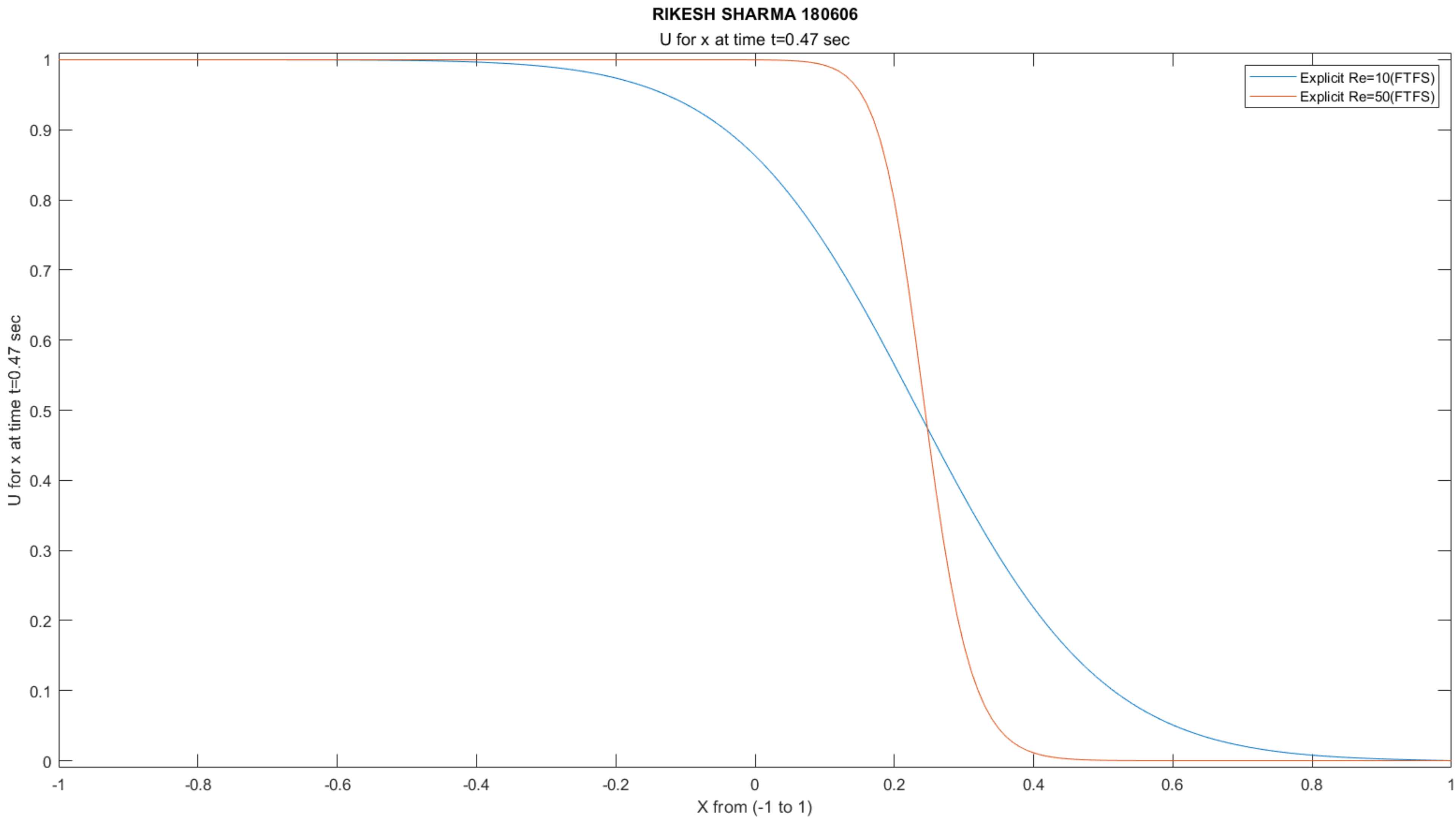
$\therefore$ Write code to calculating above.
for various values of x and t we get.

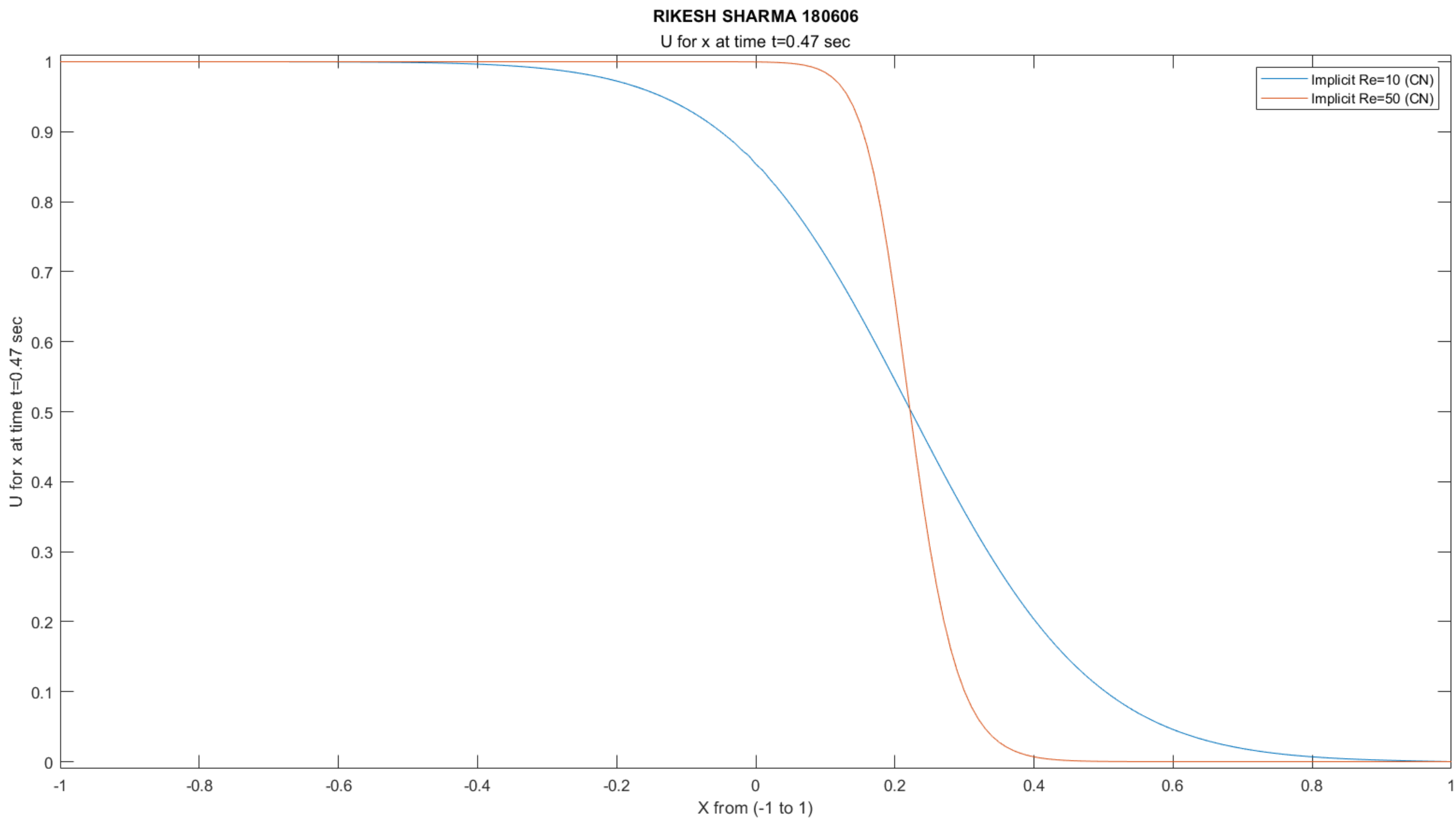
EXACT SOLUTION (Pseudocode)

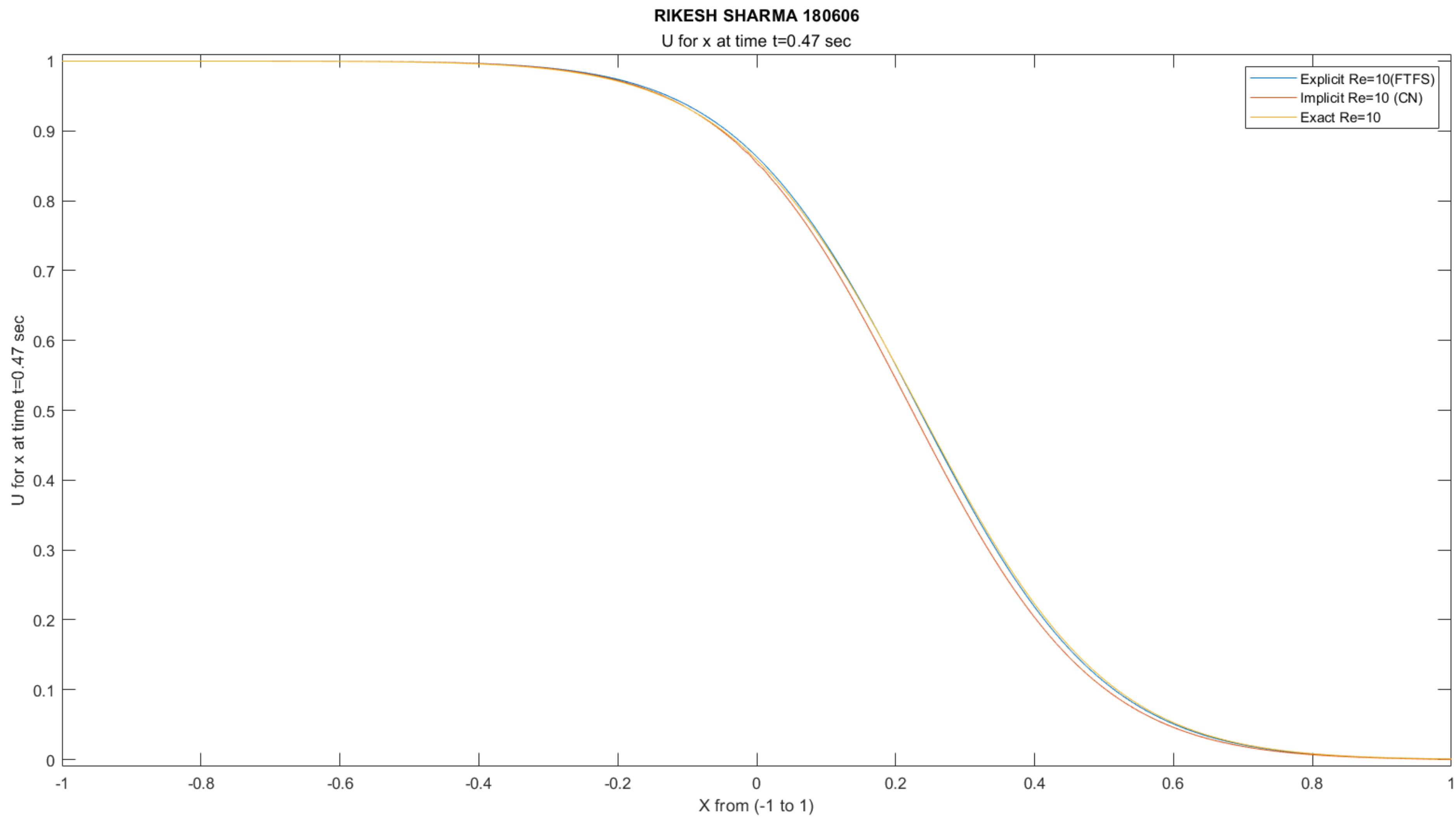
```
function uExact=exactBurgerSol(Re,t)

xmax=1;
dx=0.01;
x=-xmax:dx:xmax;
N=2*xmax/dx+1;
uExact=zeros(N,1);
for i = 1:N
    f = @(z)  ((x(i)-z)/t).*exp(-0.5*Re*(z+0.5*((x(i)-z).^2)/t)) ;
    g = @(z)  ((x(i)-z)/t).*exp(-0.5*Re*(0.5*((x(i)-z).^2)/t)) ;
    h = @(z)  exp(-0.5*Re*(z+0.5*((x(i)-z).^2)/t));
    l = @(z)  exp(-0.5*Re*(0.5*((x(i)-z).^2)/t));

    fi = integral(f,-inf,0);
    gi = integral(g,0,inf);
    hi = integral(h,-inf,0);
    li = integral(l,0,inf);

    uExact(i)=(fi+gi)/(hi+li);
end

end
```

PLOTS

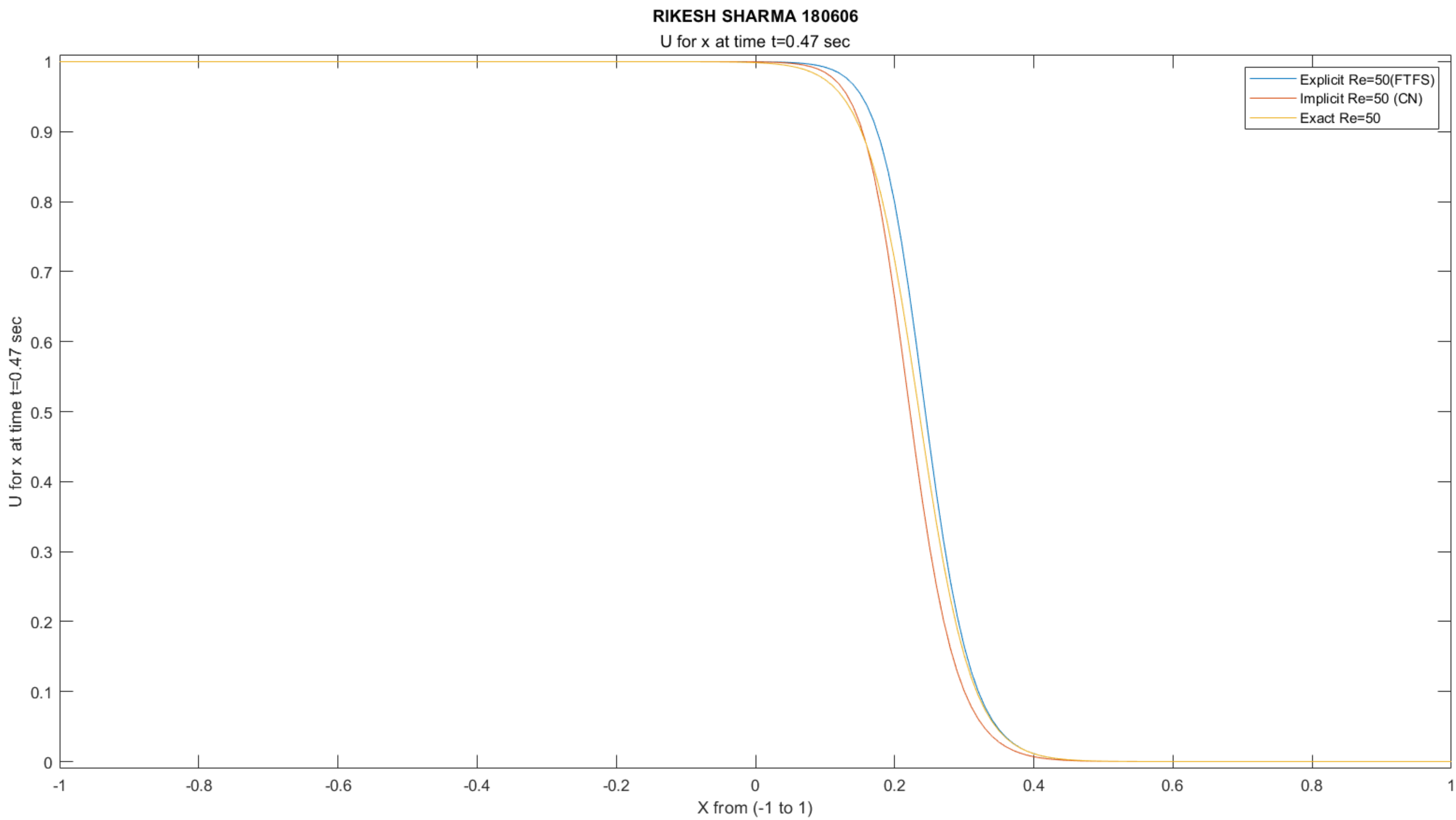
PLOTS


RIKESH SHARMA 180606
U for x at time t=0.47 sec
Explicit Re=10(FTFS)
Explicit Re=50(FTFS)
U for x at time t=0.47 sec
X from (-1 to 1)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

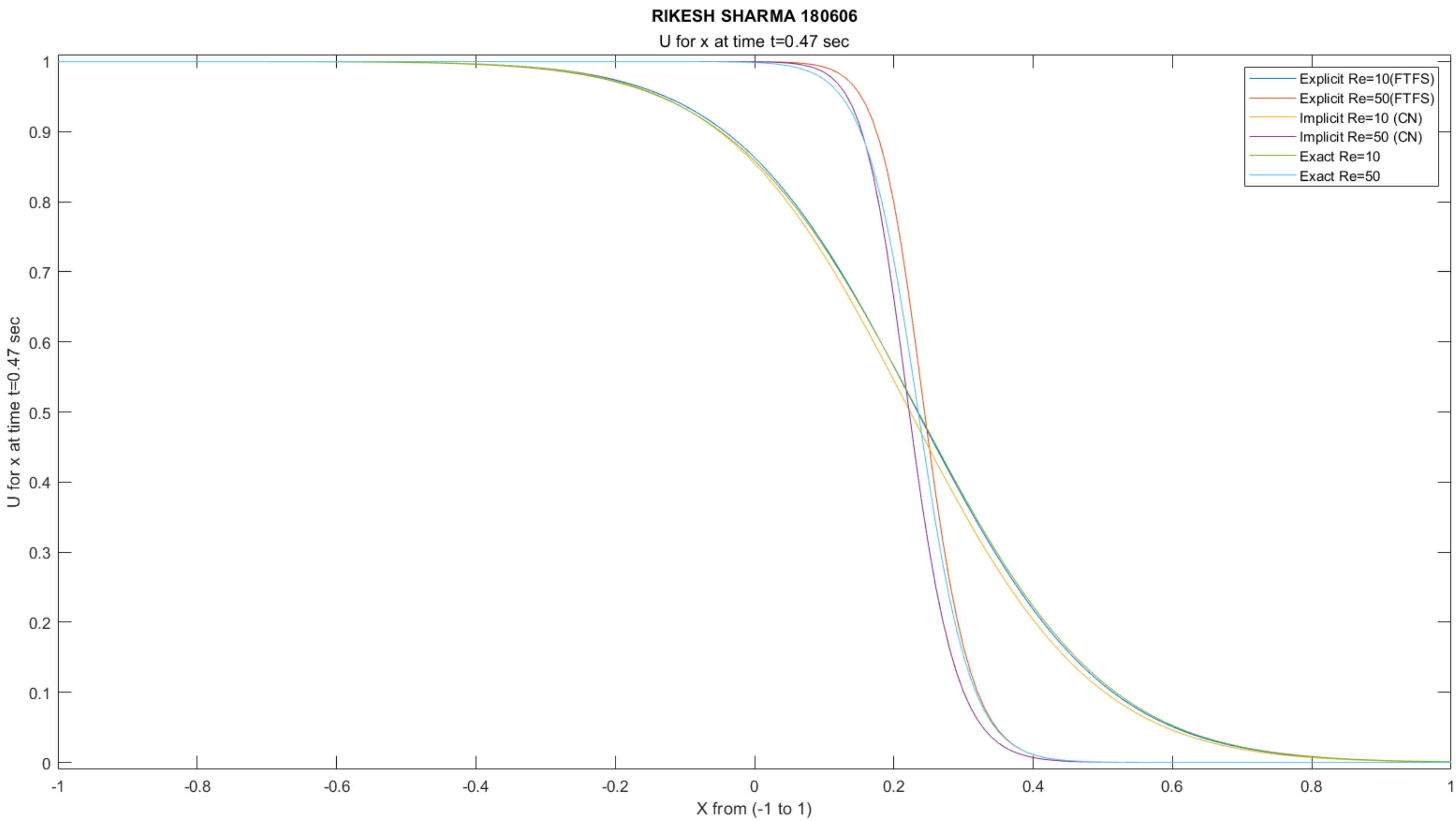

RIKESH SHARMA 180606
U for x at time t=0.47 sec
Implicit Re=10 (CN)
Implicit Re=50 (CN)
U for x at time t=0.47 sec
X from (-1 to 1)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1



RIKESH SHARMA 180606
U for x at time t=0.47 sec
Explicit Re=10(FTFS)
Implicit Re=10 (CN)
Exact Re=10
U for x at time t=0.47 sec
X from (-1 to 1)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1



RIKESH SHARMA 180606
U for x at time t=0.47 sec
Explicit Re=50(FTFS)
Implicit Re=50 (CN)
Exact Re=50
U for x at time t=0.47 sec
X from (-1 to 1)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


RIKESH SHARMA 180606
U for x at time t=0.47 sec
Explicit Re=10(FTFS)
Explicit Re=50(FTFS)
Implicit Re=10 (CN)
Implicit Re=50 (CN)
Exact Re=10
Exact Re=50
U for x at time t=0.47 sec
X from (-1 to 1)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

Solving the Burger eq^n using Explicit and Implicit scheme, ~~the~~ following Observation can be made.

Observations

i) Explicit scheme was observed to be <u>unstable</u>

if $\left|\frac{u_{max}\Delta t}{\Delta x}\right| \geq 1$ {ie, conditionally stable}

as for $\Delta x = 0.01$, $\Delta t = 0.01$] we get unstable explicit scheme

if we reduce Δt to (say) 0.001

ie, $\Delta x = 0.01$, $\Delta t = 0.001$ we get stable explicit scheme.

$\Rightarrow$ Conditionally stable

ii) Implicit scheme was ~~of~~ found to be unconditionally stable.

iii) FTFS and FTBS is $O(\Delta t, \Delta x)$ accurate

iv) Lax-Friedrich is $O(\Delta t, \Delta x^2)$ accurate

v) Lax-Wendroff is $O(\Delta t^2, \Delta x^2)$ accurate
ie, 2nd order accuracy in both time and space.

vi) FTCS is found to be unconditionally unstable and we get very large spike at ~~a~~ points about $x=0$.

vii) Upwind Schemes gives Numerical Diffusion because of the type of Finite Diffusion approximation.

viii) Lax's modification term was observed to be introducing a stabilizing diffusion to unconditionally unstable FTCS method.

CODE

```
function burgerSolutions()
xmax=1;
dx=0.01;
x=-xmax:dx:xmax;

Re=10;
ue10=explicitBurger(Re);
ui10=implicitBurger(Re);
uExact10=exactBurgerSol(Re,0.47);

Re=50;
ue50=explicitBurger(Re);
ui50=implicitBurger(Re);
uExact50=exactBurgerSol(Re,0.47);

figure
plot(x,ue10(:,4700));
hold on
plot(x,ue50(:,4700));
plot(x,ui10(:,47));
plot(x,ui50(:,47));
plot(x,uExact10);
plot(x,uExact50);
xlabel(' X from (-1 to 1) ');
ylabel('U for x at time t=0.47 sec');
```

```
title('RIKESH SHARMA 180606','U for x at time t=0.47 sec');
legend('Explicit Re=10(FTFS)','Explicit Re=50(FTFS)','Implicit Re=10
(CN)','Implicit Re=50 (CN)','Exact Re=10','Exact Re=50');
xlim([-1 1]);
ylim([-0.01 1.01]);

figure
plot(x,ue10(:,4700));
hold on
plot(x,ui10(:,47));
plot(x,uExact10);
xlabel(' X from (-1 to 1) ');
ylabel('U for x at time t=0.47 sec');
title('RIKESH SHARMA 180606','U for x at time t=0.47 sec');
legend('Explicit Re=10(FTFS)','Implicit Re=10 (CN)','Exact Re=10');
xlim([-1 1]);
ylim([-0.01 1.01]);

figure
plot(x,ue50(:,4700));
hold on
plot(x,ui50(:,47));
plot(x,uExact50);
xlabel(' X from (-1 to 1) ');
ylabel('U for x at time t=0.47 sec');
title('RIKESH SHARMA 180606','U for x at time t=0.47 sec');
legend('Explicit Re=50(FTFS)','Implicit Re=50 (CN)','Exact Re=50');
xlim([-1 1]);
ylim([-0.01 1.01]);

figure
plot(x,ue10(:,4700));
hold on
plot(x,ue50(:,4700));
xlabel(' X from (-1 to 1) ');
ylabel('U for x at time t=0.47 sec');
title('RIKESH SHARMA 180606','U for x at time t=0.47 sec');
legend('Explicit Re=10(FTFS)','Explicit Re=50(FTFS)');
xlim([-1 1]);
ylim([-0.01 1.01]);

figure
plot(x,ui10(:,47));
hold on
plot(x,ui50(:,47));
xlabel(' X from (-1 to 1) ');
ylabel('U for x at time t=0.47 sec');
title('RIKESH SHARMA 180606','U for x at time t=0.47 sec');
legend('Implicit Re=10 (CN)','Implicit Re=50 (CN)');
```

```
xlim([-1 1]);
ylim([-0.01 1.01]);
end

function u=explicitBurger(Re)
xmax=1;
nu=1/Re;
dt=0.0001;
dx=0.01;

s=nu*dt/(dx*dx);

N=(1/dt)+1;
gridP=2*xmax/dx +1;

u=zeros(gridP,N);
u(1:xmax/dx,1)=1.0;
u(xmax/dx+1:gridP,1)=0.0;
u(1,:)=1.0;
u(gridP,:)=0.0;

for i=1:N-1

%       u(2:gridP-1,i+1)=u(2:gridP-1,i)...
%           -(dt/dx)*u(2:gridP-1,i).*(u(2:gridP-1,i)-u(1:gridP-
2,i))...
%           +0.5*(dt^2/dx^2)*u(2:gridP-1,i+1).^2.*(u(3:gridP,i+1)-
2*u(2:gridP-1,i+1)+u(1:gridP-2,i+1))...
%           +s*(u(3:gridP,i)-2*u(2:gridP-1,i)+u(1:gridP-2,i));
    u(2:gridP-1,i+1)=u(2:gridP-1,i)...
        -(dt/dx)*u(2:gridP-1,i).*(u(3:gridP,i)-u(2:gridP-1,i))...
         +0.5*(dt^2/dx^2)*u(2:gridP-1,i+1).^2.*(u(3:gridP,i+1)-
2*u(2:gridP-1,i+1)+u(1:gridP-2,i+1))...
         +s*(u(3:gridP,i)-2*u(2:gridP-1,i)+u(1:gridP-2,i));

end

end
function u=implicitBurger(Re)
xmax=1;
nu=1/Re;
dt=0.01;
dx=0.01;

s=nu*dt/(dx*dx);

N=(1/dt)+1;
gridP=2*xmax/dx +1;
```

```
u=zeros(gridP,N);
u(1:xmax/dx,1)=1.0;
u(xmax/dx+1:gridP,1)=0.0;
u(1,:)=1.0;
u(gridP,:)=0.0;
a=zeros(gridP-2);
b=zeros(gridP-2);
c=zeros(gridP-2);
d=zeros(gridP-2);

for i=1:N-1
    b(1:gridP-2) = ( 1-u(2:gridP-1,i)*dt/(2*dx)+s);
    a(2:gridP-2) = -0.5*s;
    c(1:gridP-3) = (0.5*u(3:gridP-1,i)*(dt/dx)-0.5*s);
    d(1)         = 0.5*s*u(1,i)+(1-
s)*u(2,i)+0.5*s*u(3,i)+u(1,i+1)*0.5*s;
    d(gridP-2)   = 0.5*s*u(gridP-2,i)+(1-s)*u(gridP-
1,i)+0.5*s*u(gridP,i)- u(gridP,i+1)*(u(gridP,i)*0.5*(dt/dx)-0.5*s);
    d(2:gridP-3) = 0.5*s*u(2:gridP-3,i)+(1-s)*u(3:gridP-
2,i)+0.5*s*u(4:gridP-1,i);

    u(2:gridP-1,i+1) = TDMA(a,b,c,d);

end


end
function uExact=exactBurgerSol(Re,t)

xmax=1;
dx=0.01;
x=-xmax:dx:xmax;
N=2*xmax/dx+1;
uExact=zeros(N,1);
for i = 1:N
    f = @(z)  ((x(i)-z)/t).*exp(-0.5*Re*(z+0.5*((x(i)-z).^2)/t)) ;
    g = @(z)  ((x(i)-z)/t).*exp(-0.5*Re*(0.5*((x(i)-z).^2)/t)) ;
    h = @(z)  exp(-0.5*Re*(z+0.5*((x(i)-z).^2)/t));
    l = @(z)  exp(-0.5*Re*(0.5*((x(i)-z).^2)/t));

    fi = integral(f,-inf,0);
    gi = integral(g,0,inf);
    hi = integral(h,-inf,0);
    li = integral(l,0,inf);

    uExact(i)=(fi+gi)/(hi+li);
end

end
```

```
function x = TDMA(a,b,c,d)
%a, b, c are the column vectors for the compressed tridiagonal
matrix, d is the right vector
n = length(b); % n is the number of rows

% Modify the first-row coefficients
c(1) = c(1) / b(1);    % Division by zero risk.
d(1) = d(1) / b(1);    % Division by zero would imply a singular
matrix.

for i = 2:n-1
    mult = b(i) - a(i) * c(i-1);
    c(i) = c(i) / mult;
    d(i) = (d(i) - a(i) * d(i-1)) / mult;
end

d(n) = (d(n) - a(n) * d(n-1))/( b(n) - a(n) * c(n-1));

% Now back substitute.
x(n) = d(n);
for i = n-1:-1:1
    x(i) = d(i) - c(i) * x(i + 1);
end
end
```