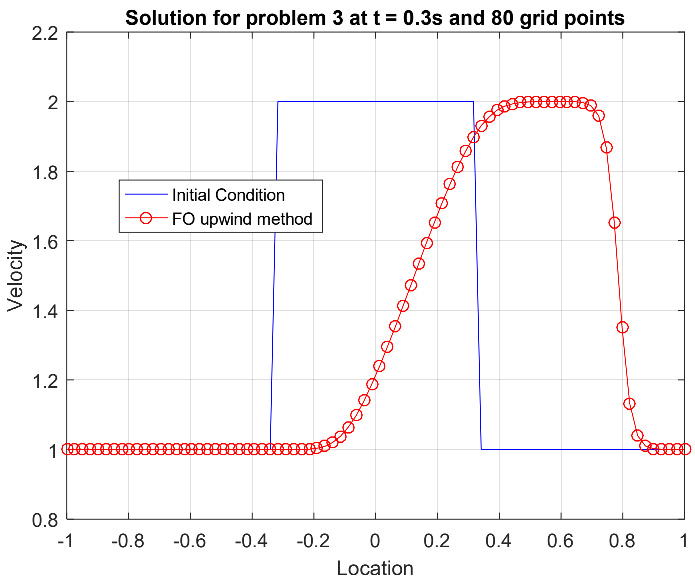
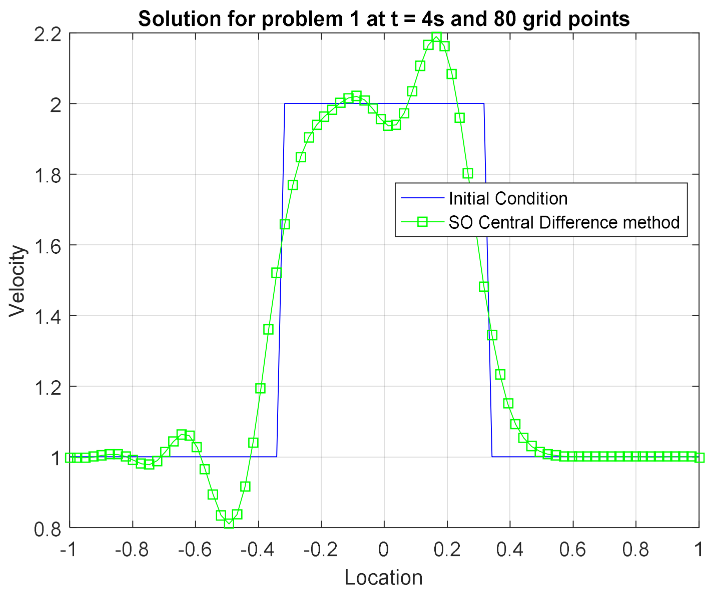
16/08/2018

ASSIGNMENT 1

NUMERICAL METHODS FOR CONSERVATION LAWS





173109003 | Sanit Prashant Bhatkar

Computational Fluid Dynamics and Heat Transfer (AE 617), Autumn 2018

Assignment 1: Scalar Laws

Name: Sanit Prashant Bhatkar

Roll No:173109003

Problem Definition:

Use the flux difference splitting algorithm in (i) first-order upwind (ii) second-order central form to numerically solve for scalar hyperbolic conservation laws

Given below with initial data

u(x; 0) = 2; |x| < 1/3

u(x; 0) = 1; elsewhere

in the domain [-1,1] and periodic boundary conditions. Discretize domain both with 40 and 80 points and use t/x = 0.8.

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0 \quad (1)$$

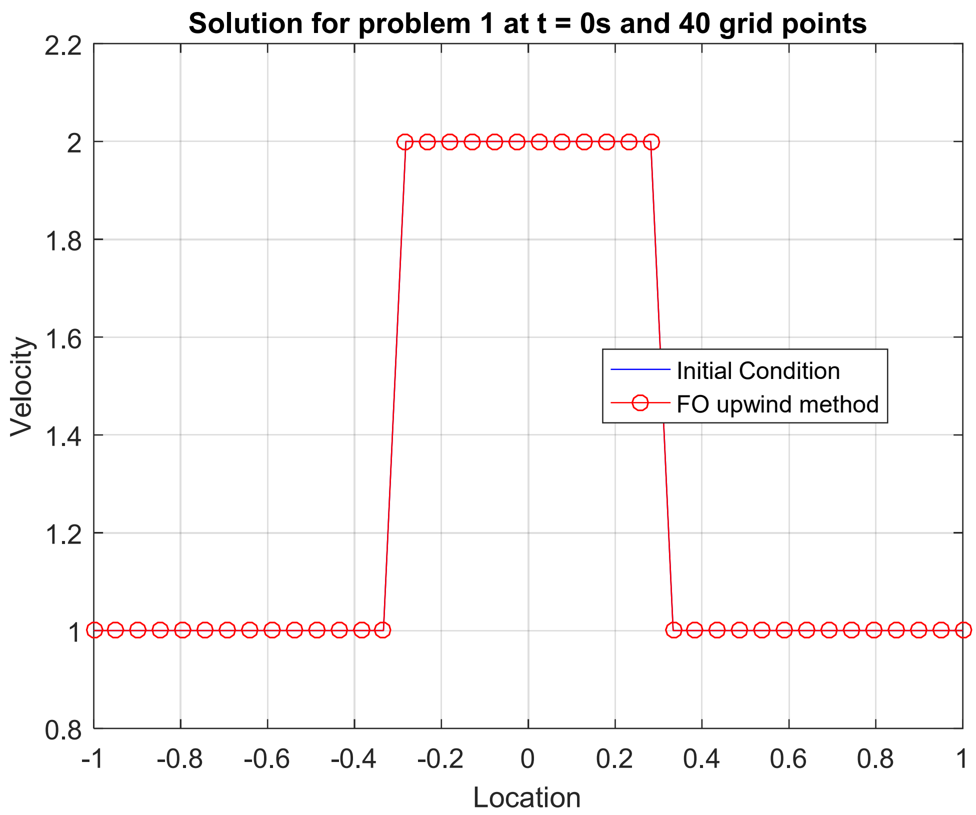
$$\frac{\partial u}{\partial t} + \frac{\partial (u^2/2)}{\partial x} = 0 \quad (2)$$

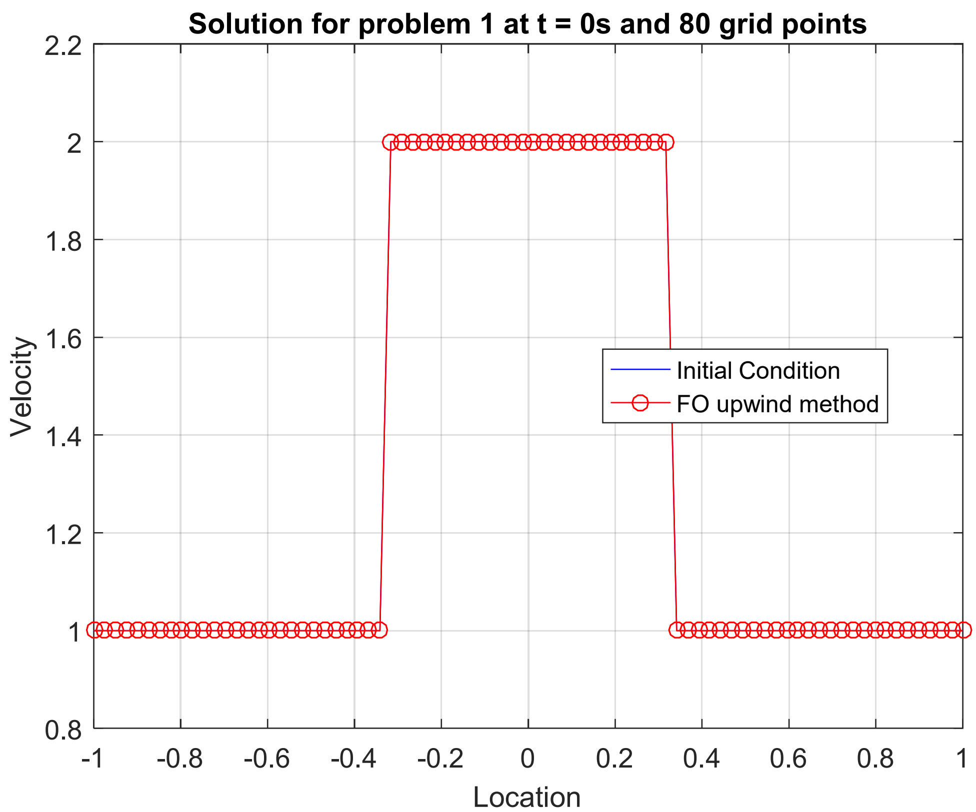
$$\frac{\partial u}{\partial t} + \frac{\partial (u \cdot (1-u))}{\partial x} = 0 \quad (3)$$

Results and Discussion

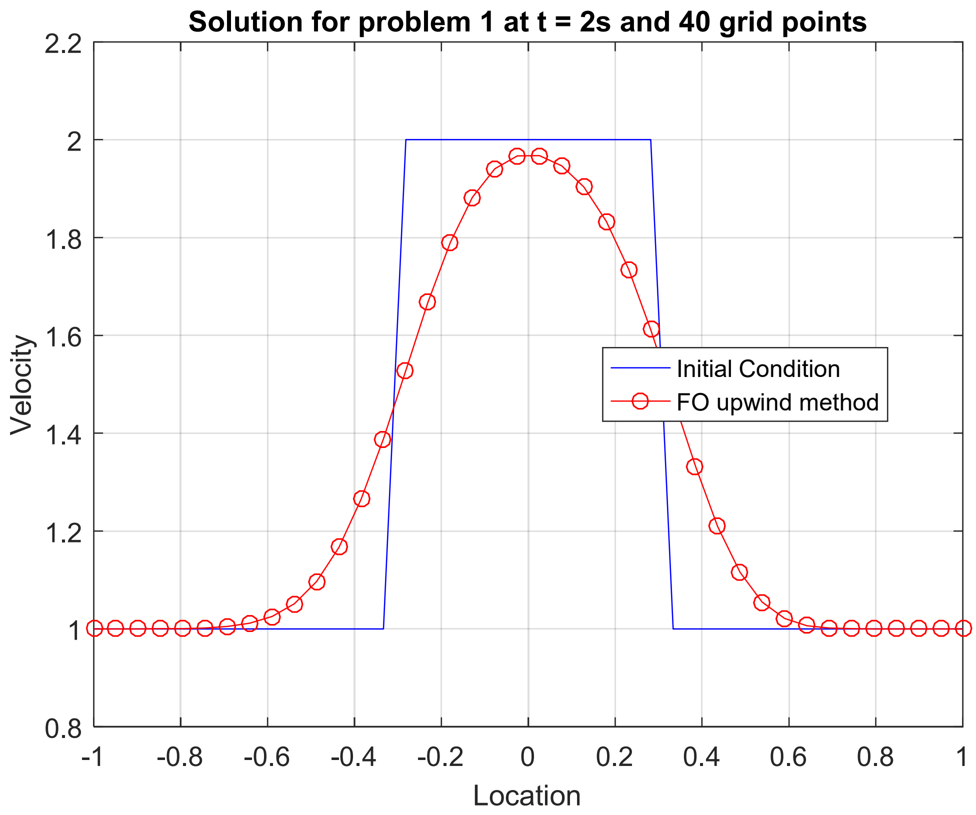
Problem 1 by upwind scheme

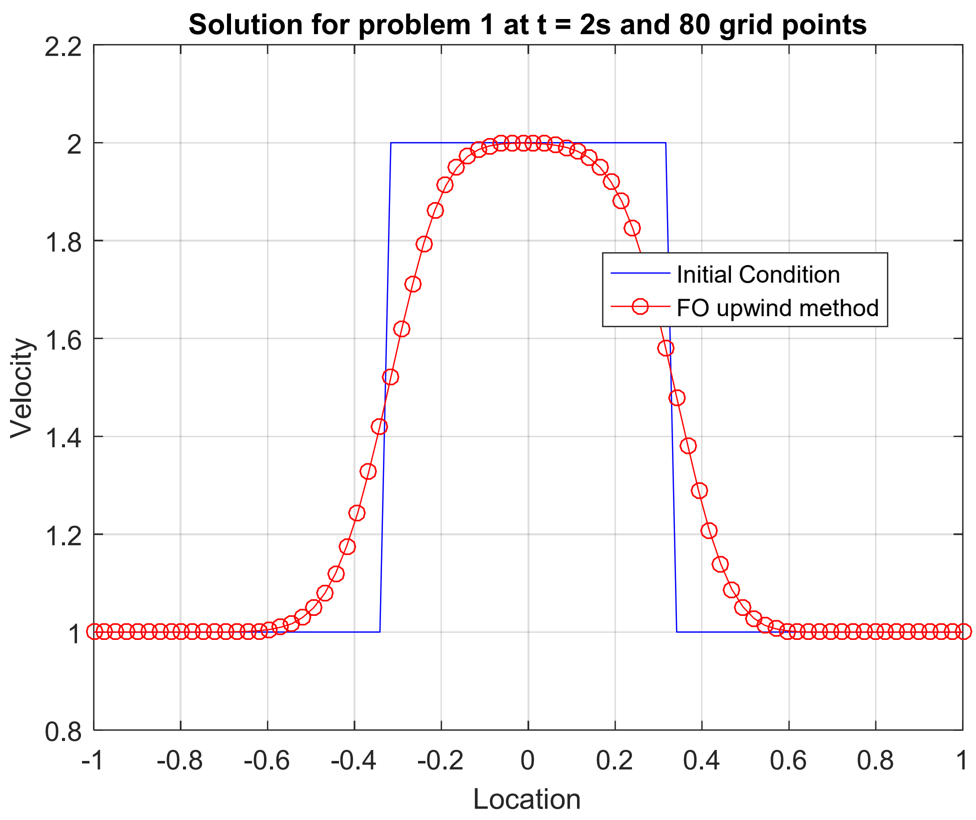
A. Solution of U(x,0)



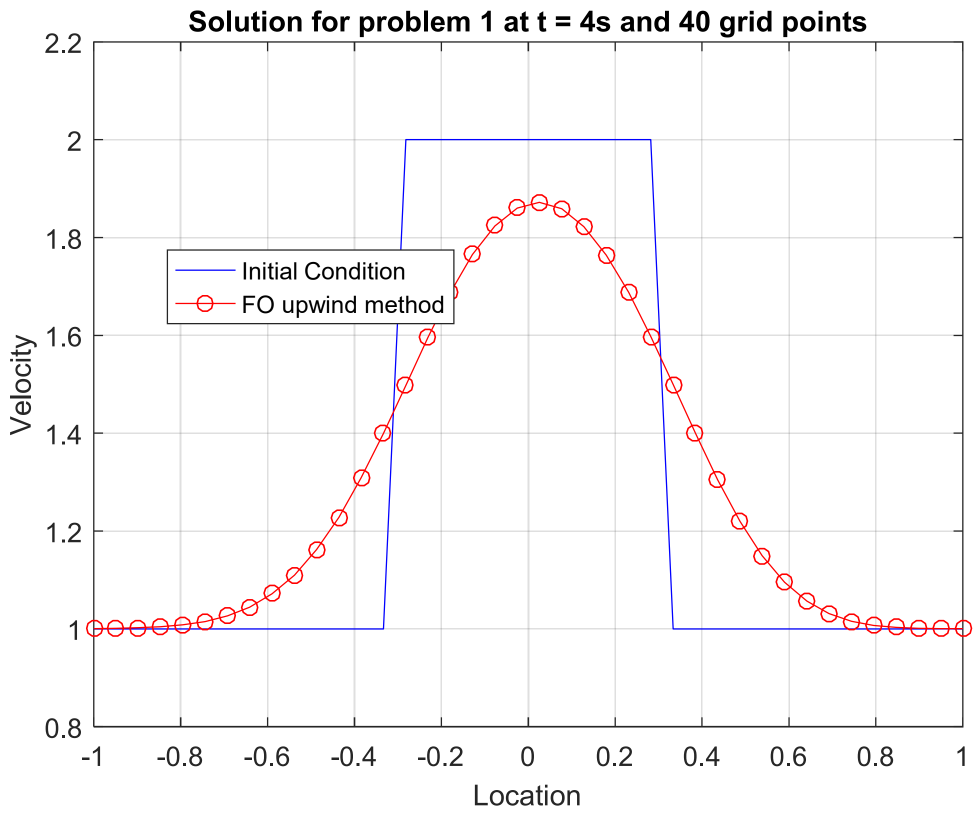


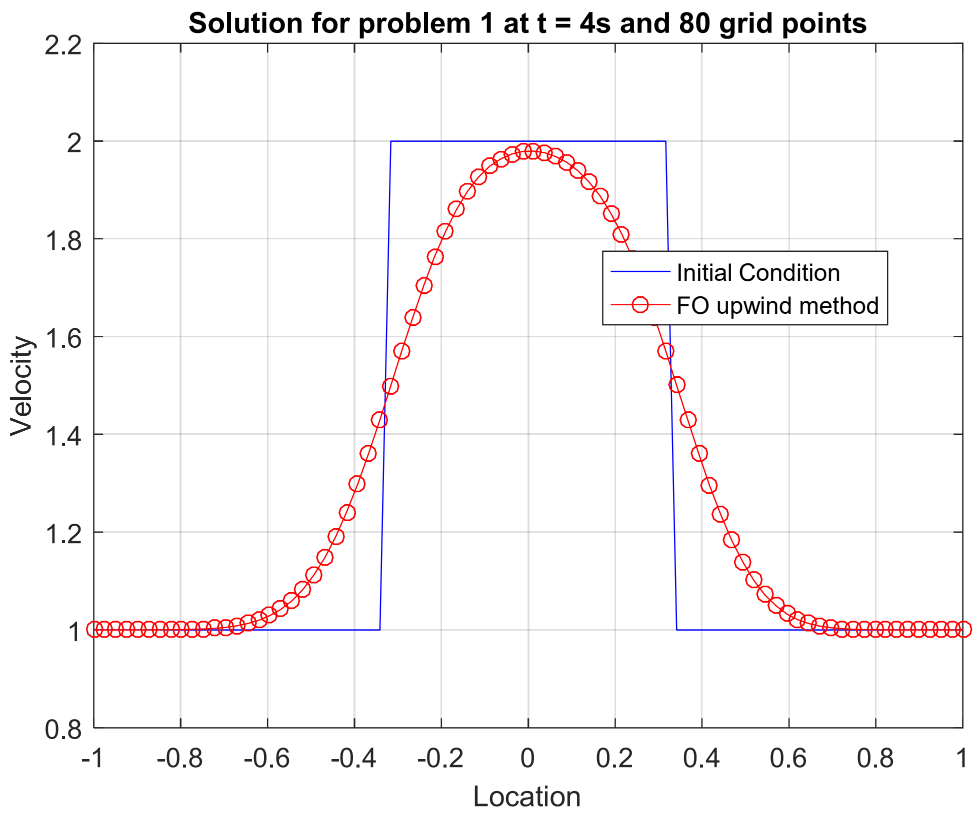
B. Solution of U(x,2)





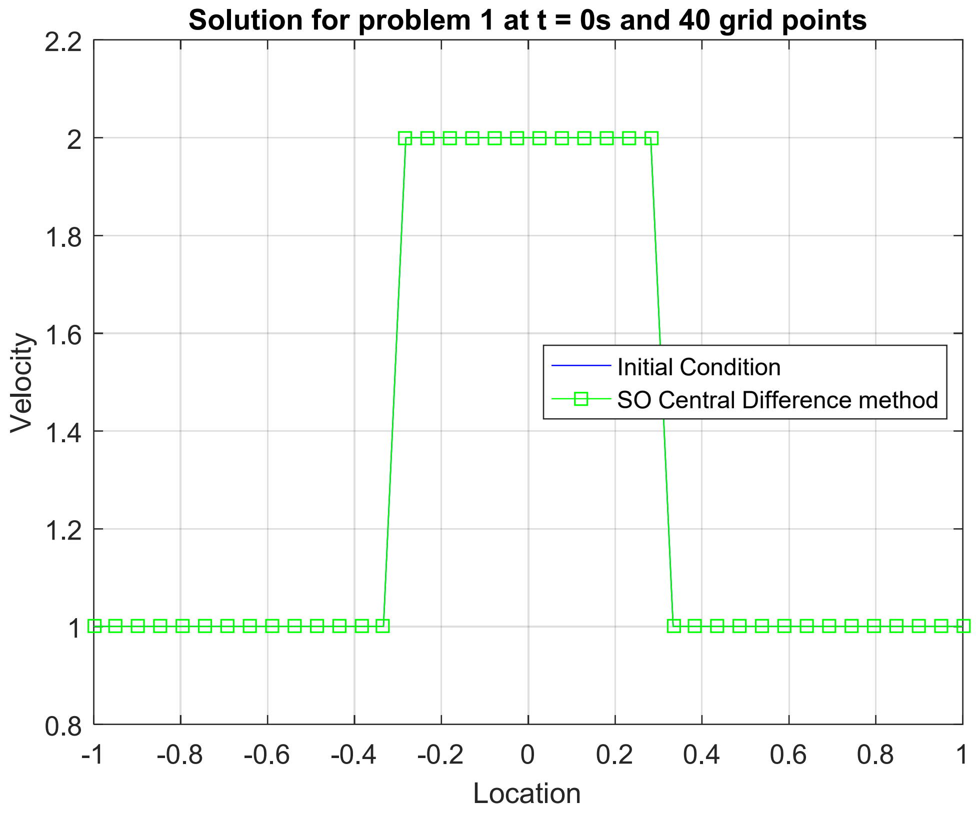
C. Solution of U(x,4)

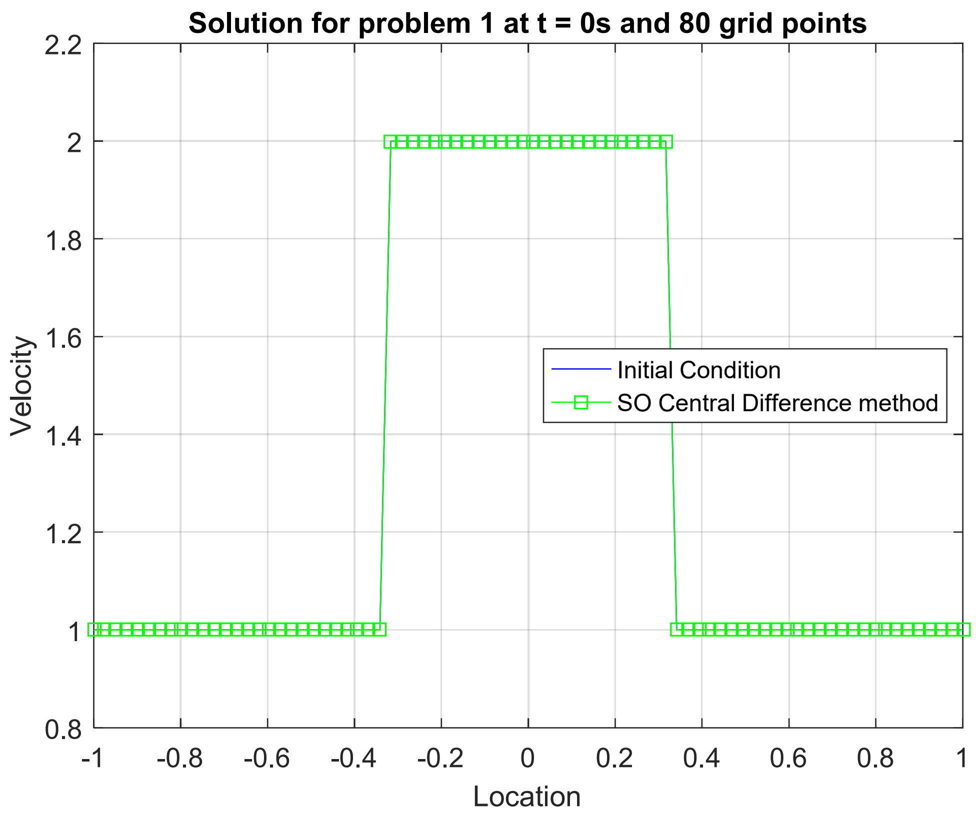




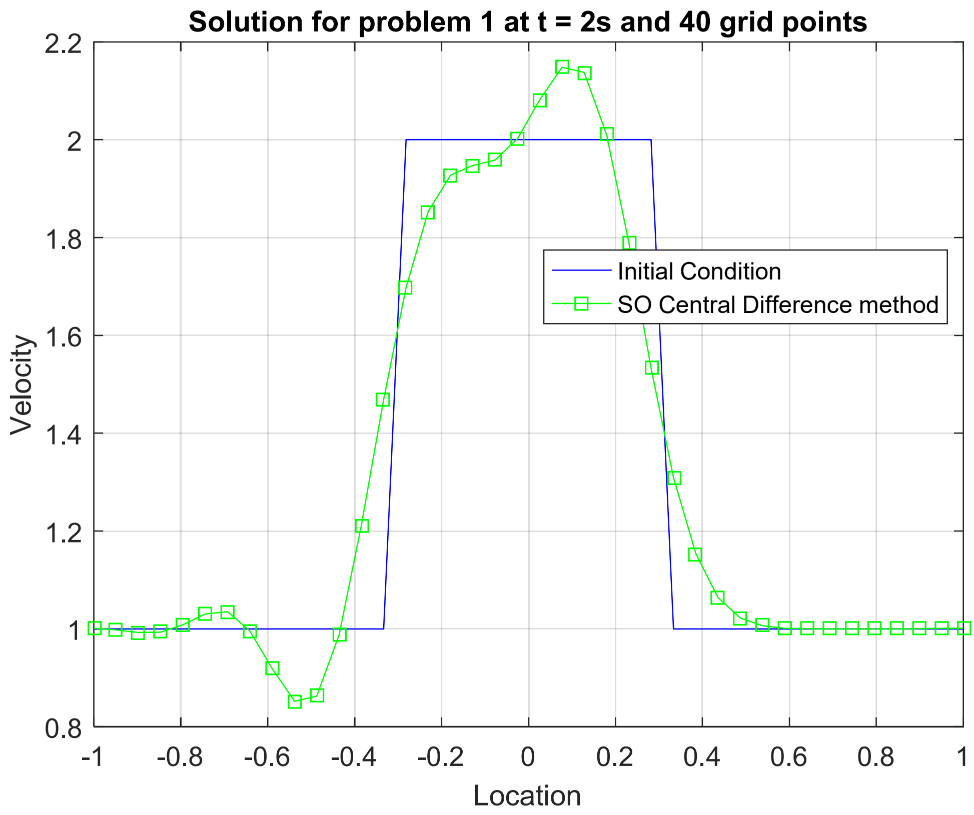
Problem 1 by central scheme

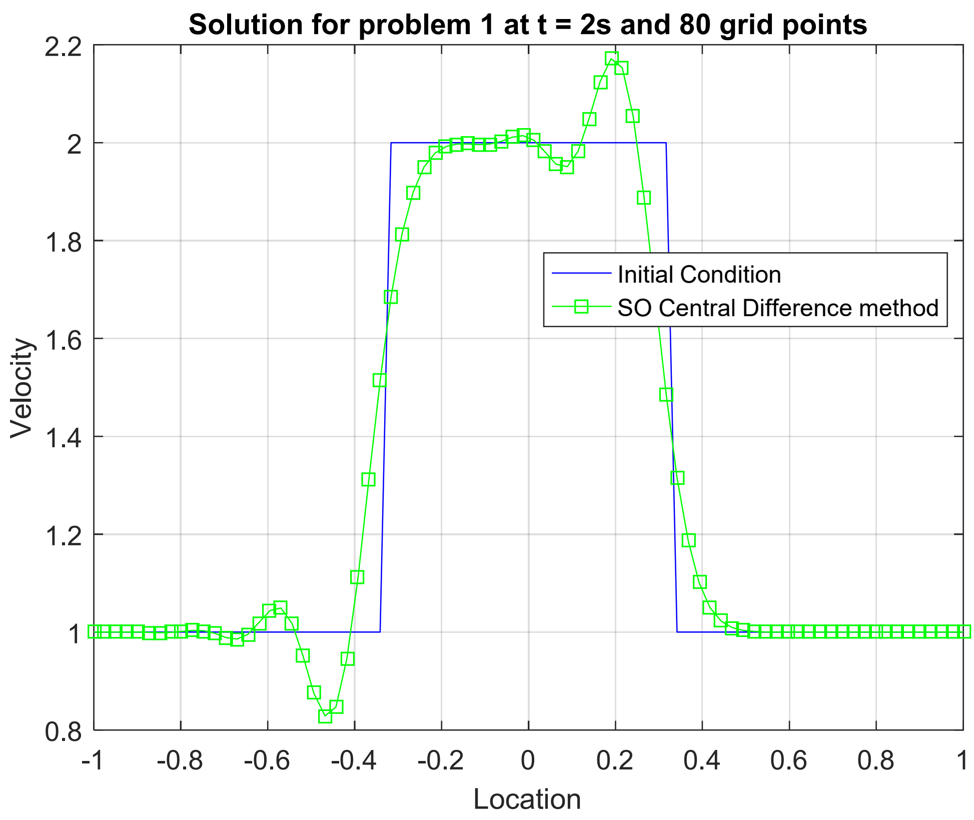
A. Solution of U(x,0)



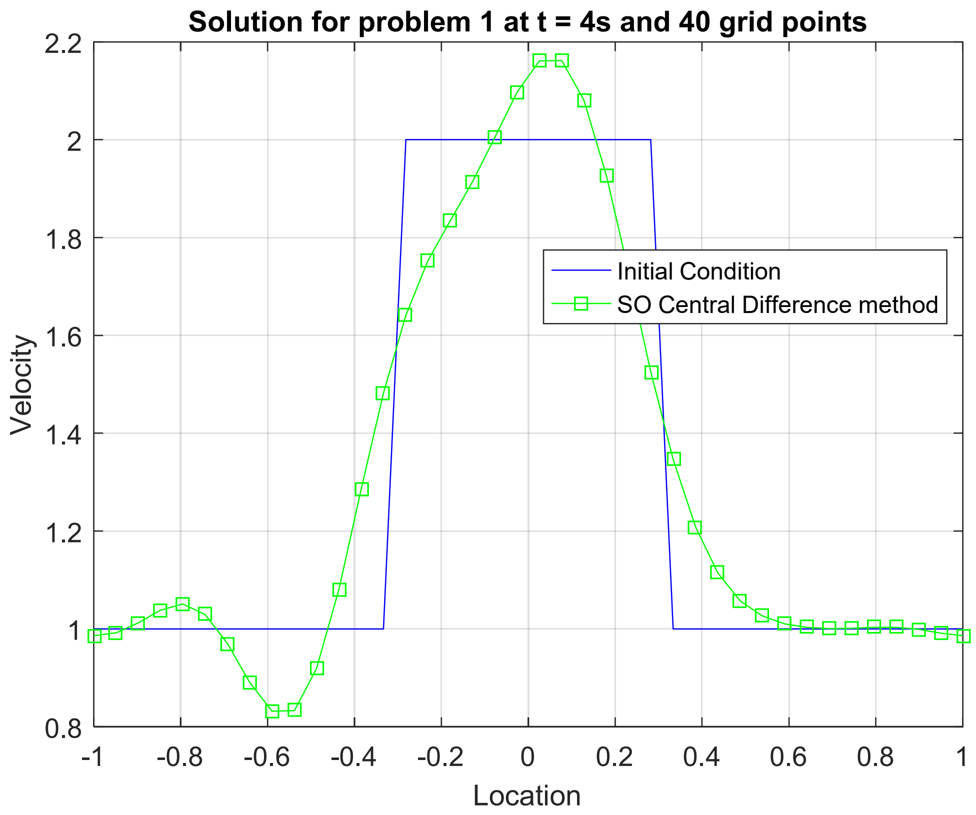


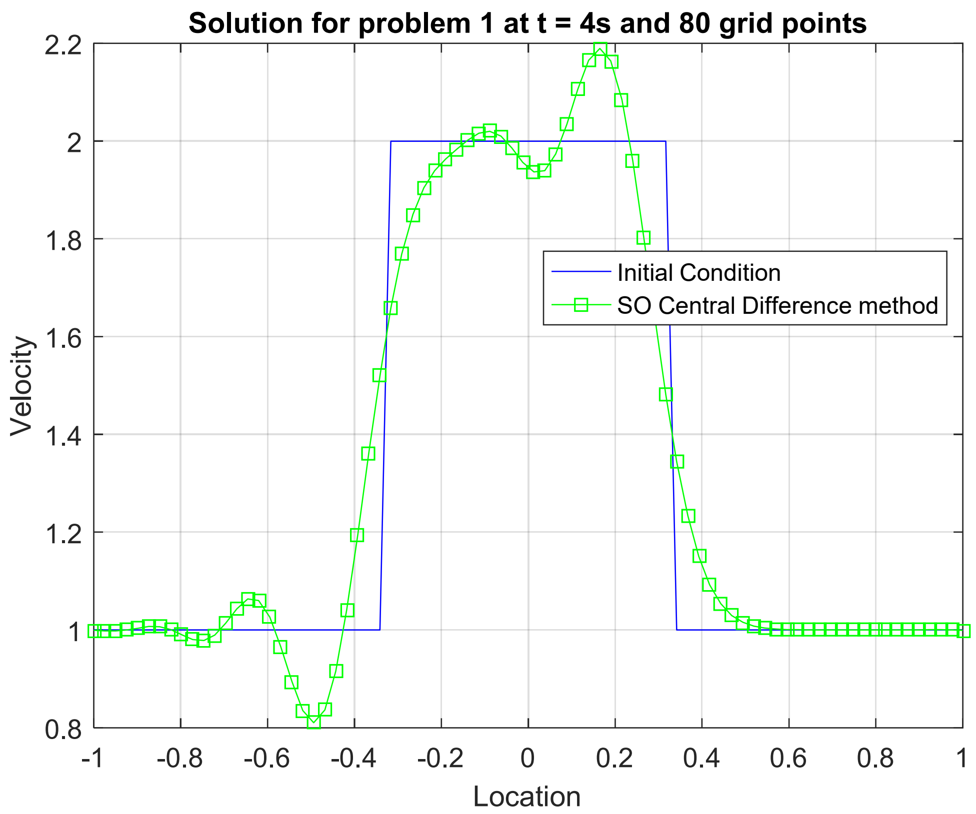
B. Solution of U(x,2)





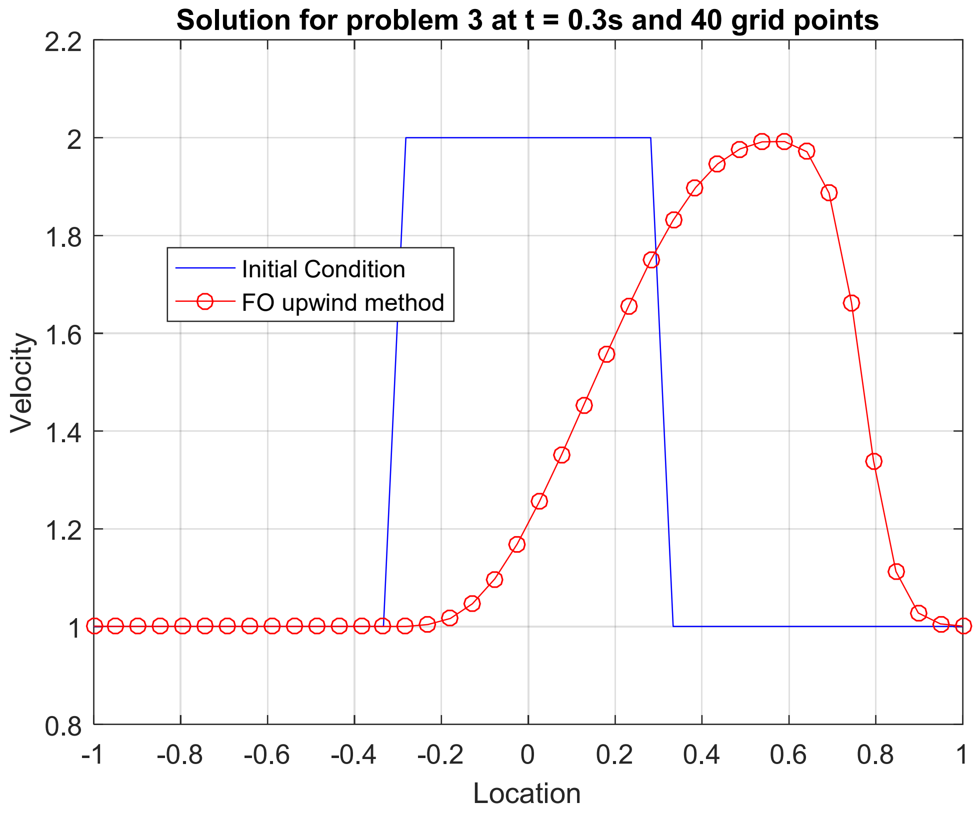
C. Solution of U(x,4)

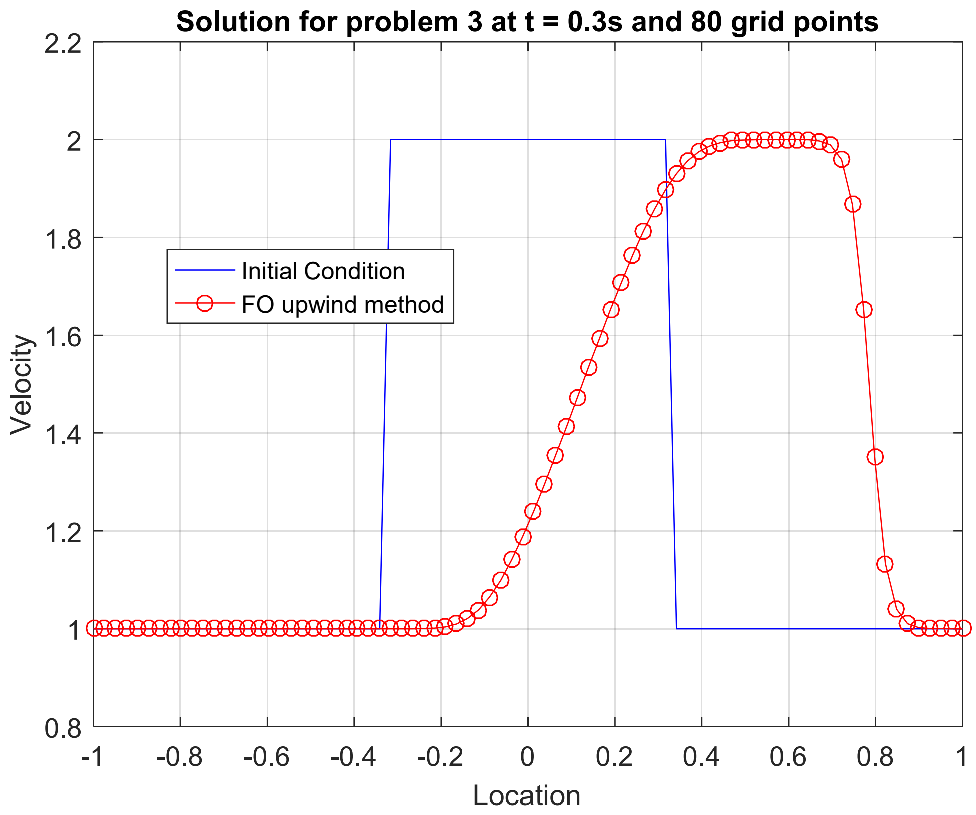




Problem 2 by upwind scheme

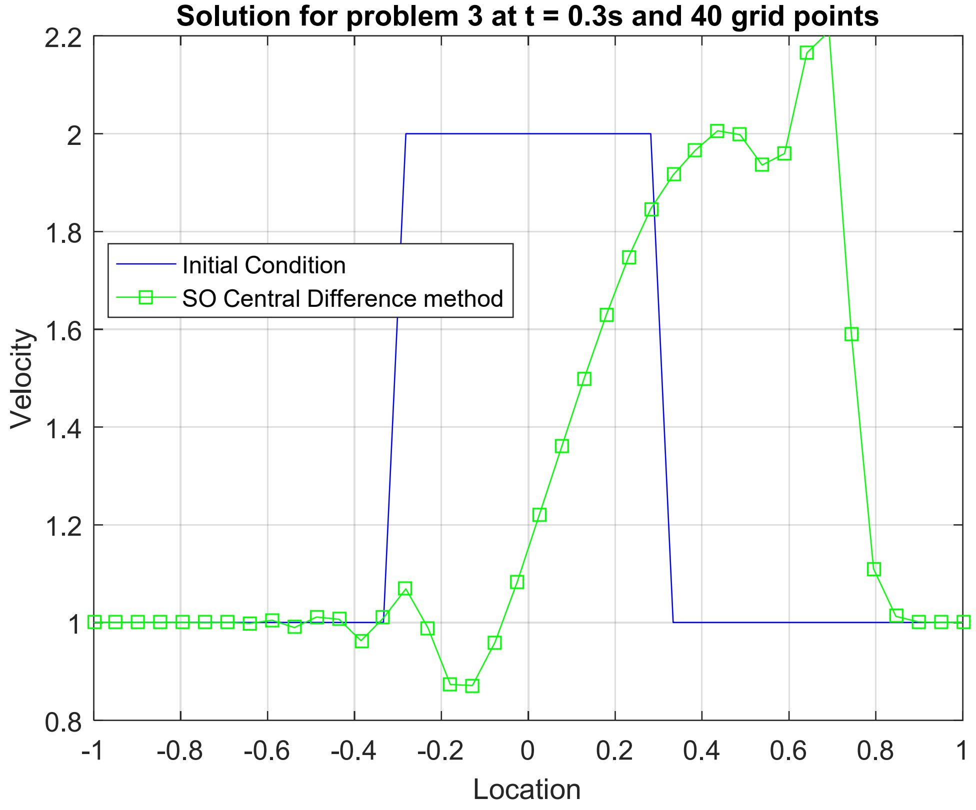
Solution of U(x,0.3)

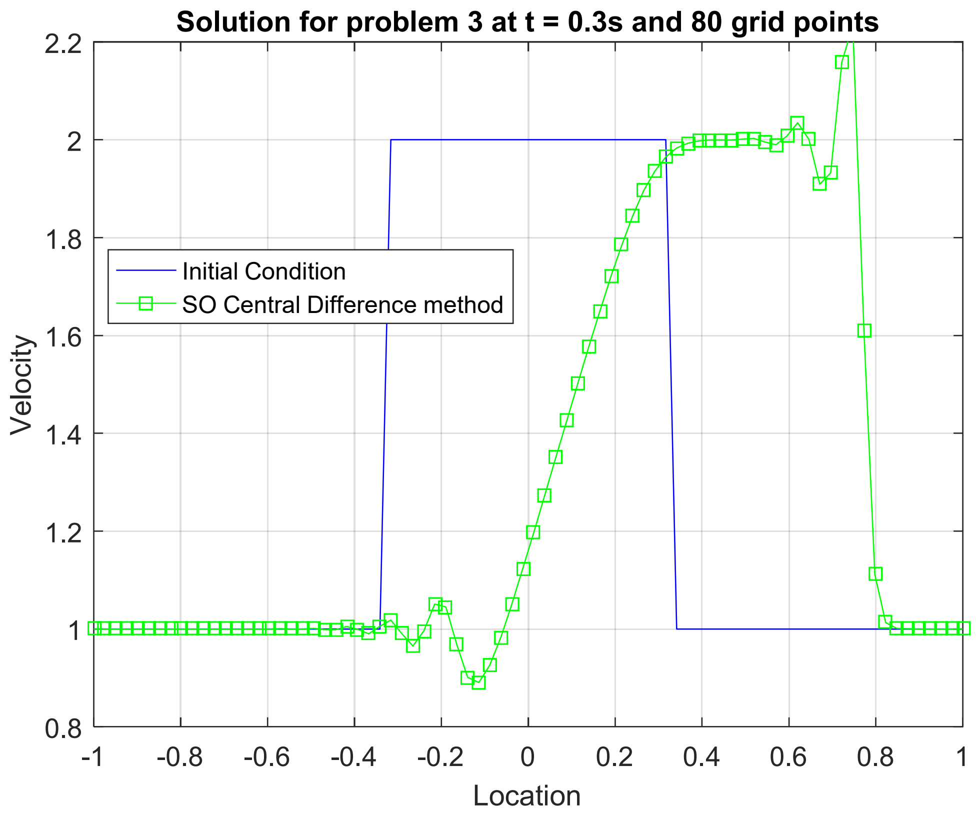




Problem 2 by central scheme

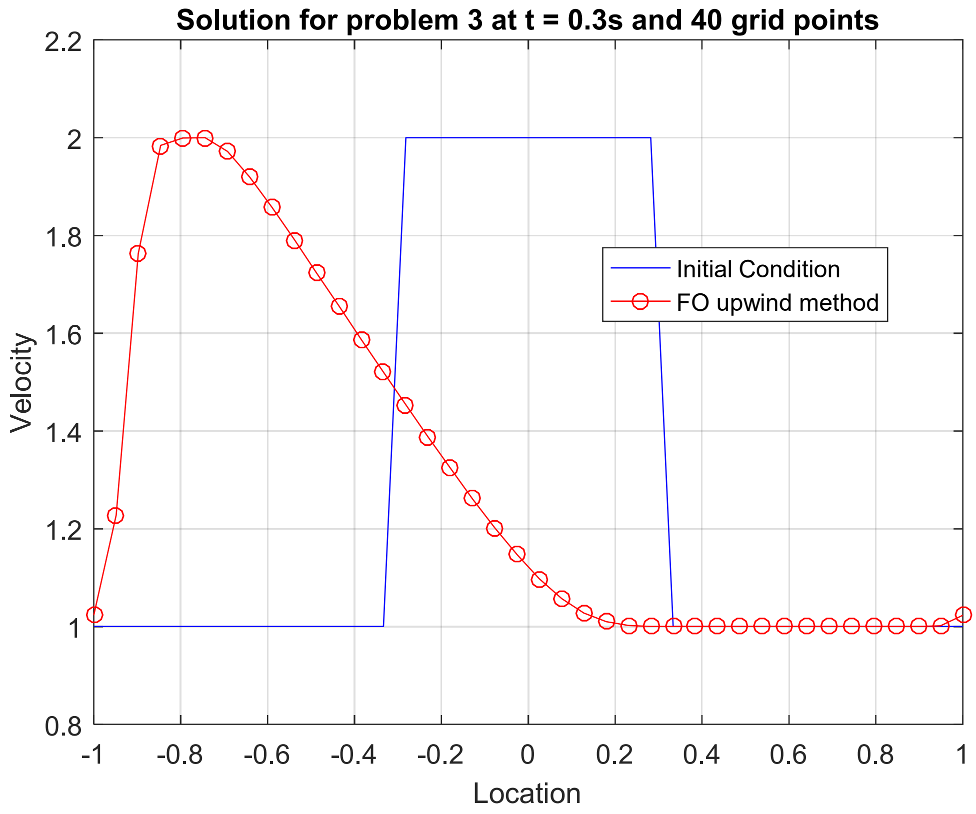
Solution of U(x,0.3)

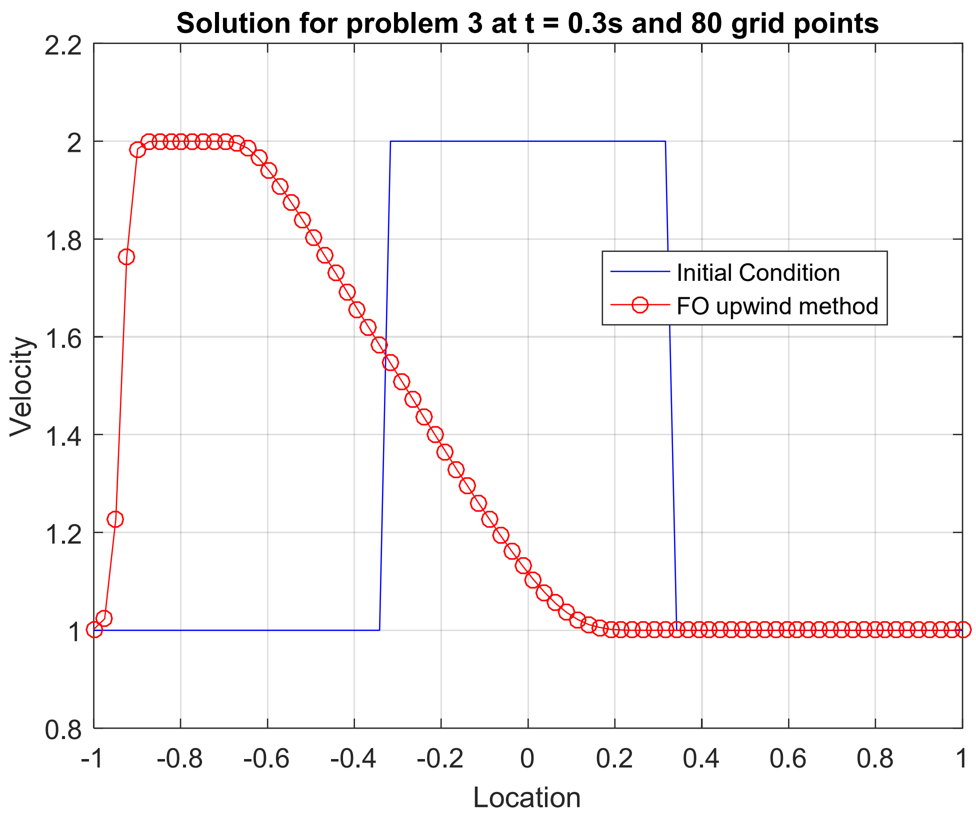




Problem 3 by upwind scheme

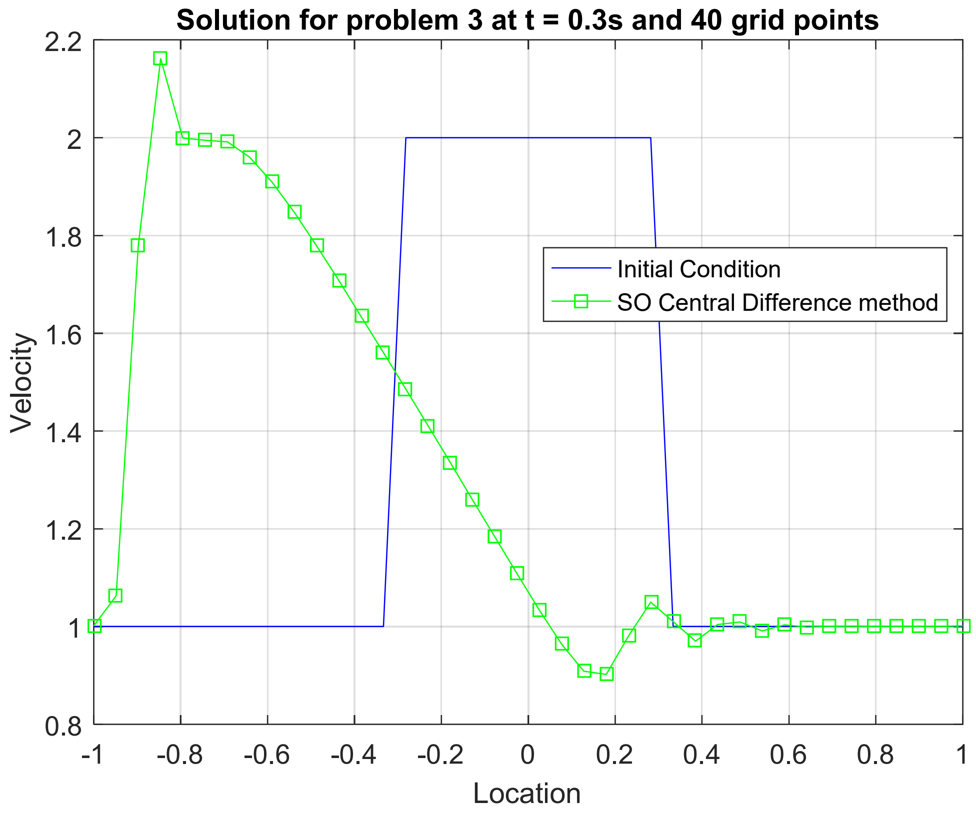
Solution of U(x,0.3)

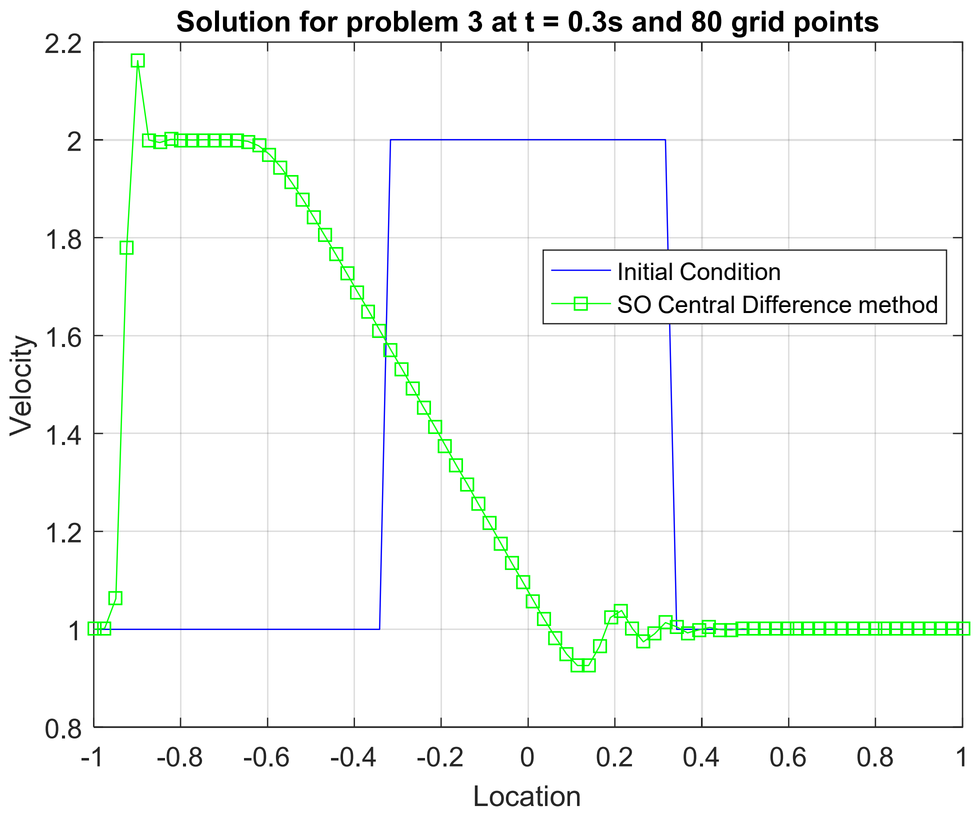




Problem 3 by central scheme

Solution of U(x,0.3)





Code of the problem

```
%===============================================================%
% Numerical Methods for Conservation Laws AE 617
%
% Assignment Number 1. Scalar Laws
%
% AUTHOR:
% Sanit P. Bhatkar (173109003@iitb.ac.in)
% Roll No: 173109003
% Place: IIT BOMBAY.
%===============================================================%

clc
clearvars

%% Input parameters

a=-1;
b=1;
l=b-a;
prbno=input('\nInput problem number: ');
fprintf('Scheme\n1.FO Upwind  2.SO Central Difference\n');
scno=input('\nInput Scheme number: ');
n=input('\nNumber of grid point: ');
tf=input('\nInput end time in s: ');



%% Grid generation

dx=l/(n-1);
x=a:dx:b;

dt=dx*0.8;
t=0:dt:(tf);

%% Initial condition

[sx sx]=size(x);
[st st]=size(t);

u=ones(1,n); %given
k=find(abs(x)<(1/3));
u(1,k(1):k(end))=2;%given
uini=u;
plot(x,u,'-b')
grid on
hold on


%% Flux calculation

f=flux(prbno,u);
uo=u;
Co=courant(prbno,u,sx);

for i=1:(sx-1)
    df(i)=f(i+1)-f(i);
end
```

```
for i=1:(sx-1)
    du(i)=u(i+1)-u(i);
end

%% First Order Upwind scheme

if scno==1

for j=1:st

% Inner node formulation

for i=2:sx-1

    if du(i-1)==0
        R=Co(i);
        sigma=sign(R);
    else
        R=df(i-1)/du(i-1);
        sigma=sign(R);
    end

    if sigma>0

        u(i)=uo(i)-df(i-1)*(dt/dx);
    else

        u(i)=uo(i)-df(i)*(dt/dx);
    end

end

% Boundary node formulation

if sigma>0

    u(sx)=uo(sx)-df(sx-1)*(dt/dx);
    u(1)=u(sx);
else

    u(1)=uo(1)-df(1)*(dt/dx);
    u(sx)=u(1);
end


%% Updating the values

f=flux(prbno,u);
uo=u;
Co=courant(prbno,u,sx);

% Flux difference update
for i=1:(sx-1)
    df(i)=f(i+1)-f(i);
end

% Velocity difference update
for i=1:(sx-1)
    du(i)=u(i+1)-u(i);
```

```
end


end

if tf==0
plot(x,uini,'-ro')
else
plot(x,u,'-ro')
end
grid on
if prbno==1
title(['Solution for problem 1 at t = ' num2str(tf) 's and ' num2str(n) '
grid points']);
elseif prbno==1
title(['Solution for problem 2 at t = ' num2str(tf) 's and ' num2str(n) '
grid points']);
else
title(['Solution for problem 3 at t = ' num2str(tf) 's and ' num2str(n) '
grid points']);
end
legend('Initial Condition','FO upwind method','location','best')
ylim([0.8 2.2])
ylabel('Velocity');
xlabel('Location');

hold off

end


%% Second Order Central Difference scheme

if scno==2

for i=1:(sx-1)

   if du(i)==0;

       nu(i)=Co(i)*(dt/dx);
   else

       nu(i)=df(i)/du(i)*(dt/dx);

   end

end

for j=1:st

% Inner node formulation
for i=2:sx-1

     u(i)=uo(i)-0.5*(dt/dx)*((1+nu(i-1))*df(i-1)+(1-nu(i))*df(i));

end

% Boundary node formulation

u(sx)=uo(sx)-0.5*(dt/dx)*((1+nu(sx-1))*df(sx-1)+(1-nu(1))*df(1));
```

```
u(1)=u(sx);


%% Updating the values

f=flux(prbno,u);
uo=u;
Co=courant(prbno,u,sx);

% Flux difference update
for i=1:(sx-1)
    df(i)=f(i+1)-f(i);
end

% Velocity difference update
for i=1:(sx-1)
    du(i)=u(i+1)-u(i);
end

% Courant number update
for i=1:(sx-1)

   if du(i)==0;

       nu(i)=Co(i)*(dt/dx);
   else

       nu(i)=df(i)/du(i)*(dt/dx);

   end

end

end

if tf==0
plot(x,uini,'-gs')
else
plot(x,u,'-gs')
end
grid on
if prbno==1
title(['Solution for problem 1 at t = ' num2str(tf) 's and ' num2str(n) '
grid points']);
elseif prbno==1
title(['Solution for problem 2 at t = ' num2str(tf) 's and ' num2str(n) '
grid points']);
else
title(['Solution for problem 3 at t = ' num2str(tf) 's and ' num2str(n) '
grid points']);
end
legend('Initial Condition','SO Central Difference method','location','best')
ylim([0.8 2.2])
ylabel('Velocity');
xlabel('Location');

hold off

end
```

Function codes of the problem

```
%% Flux function

function f = flux(prbno,u)

if prbno == 1
    f=u;
elseif prbno == 2
    f=(u.^2)/2;
else
    f=u-u.^2;
end

end
```

```
%% Courant function

function C = courant(prbno,u,sx)

if prbno == 1
    C=ones(1,sx-1);

elseif prbno == 2

    for i=1:sx-1
        C(i)=(u(i+1)+u(i))/2;
    end

else

    for i=1:sx-1
        C(i)=1-(u(i+1)+u(i));
    end
end

end
```