Jose A. Millan
April 28, 2019

# ME 254 Computational Fluid Dynamics
## Homework 6

Due on 03/25/19 at 11:59 pm (through Catcourses)
Maximum points: 250

1. (**170 points**) Consider the inviscid Burgers' equation with initial condition given by

$$u(x,0) = 3 \qquad -3 \leq x < -1 \tag{1}$$
$$u(x,0) = -1 \qquad -1 < x < 1 \tag{2}$$
$$u(x,0) = 3 \qquad 1 < x \leq 3 \tag{3}$$

with periodic conditions at the boundaries. Note that the exact solution is a combination of a shock wave and an expansion fan.

(a) (**90 points**) Write a finite volume code to solve the Burgers' equation using the following numerical schemes.
   (i) Lax method
   (ii) Lax-Wendroff method.
   (iii) Godunov scheme
   (iv) Roe scheme
   (v) Second-order Roe scheme (without any flux limiters)
   (vi) Second-order Roe scheme (with minmod flux limiter)

(b) (**80 points**) Plot the exact solution and the numerical solutions (all on the same plot) at $t = 0.25$ s, $t = 0.5$ s, $t = 0.75$ s, and $t = 1$ s . While running the code, use $\Delta x = 0.01$. I am not giving any instructions on $\Delta t$. Choose any $\Delta t$ (and specify it in your homework) ensuring that stability restrictions are satisfied.

2. (**80 points**) Now, consider the initial condition given by

$$u(x,0) = \exp\left(\frac{-x^2}{2}\right) \qquad -3 \leq x \leq 3 \tag{4}$$

with periodic conditions at the boundaries

(a) (**72 points**) Determine the numerical solutions (for various instants of time chosen by you) using all schemes considered above. Show your results at 4 instants of time.

(b) (**2 points**) Does this initial condition lead to a discontinuity?

(c) (**6 points**) If so, at what value of time?

```
%
 ----------------------------------------------------------------------
clc
close all
clear all

% The following code will try to plot everything using the
% Lax method

% initial conditions
dx = 0.01;
dt = 3E-3;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = [3*ones(1,length(-3:dx:-1)),
 -1*ones(1,length(-1+dx:dx:1)), 3*ones(1,length(1+dx:dx:3))];

% now lets do the actual algorithm
% 1st for Lax Methods - for 0.25 for testing

time = 1.0;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

% exact solution
[position_e, velocity_e] = exact(time);

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i,length(space));

%     % iterator
%     for w=2:1:length(space)-1
%         U(i,w) = 0.5*(U(i-1,w+1)+B*(-0.5*U(i-1,w
+1)^2+0.5*U(i-1,w-1)^2)+U(i-1,w-1));
%     end

    % iterator
    % lets split everything and then regroup it again. I think that's
 the
    % best way to work with this
```

```
    for w=2:1:length(space)-1
        A = U(i-1,w+1)+U(i-1,w-1);
        BB = (U(i-1,w+1)^2)/2;
        C = (U(i-1,w-1)^2)/2;

        U(i,w) = (1/2)*A-(B/2)*(BB-C);
    end
end

% figure(1)
% plot(space,U(length(tt),:),position_e, velocity_e)
% titlename = ['Lax''s Method at ', num2str(time), 's'];
% title(titlename)
% xlabel('position x')
% ylabel('velocity U')
% legend('numerical','exact')
% ylim([-3 4])
% grid on


function [x_points, y_points] = exact(time)
    % initial conditions
    xi_comp = -1;
    xi_exp = 1;

    % for compression, position
    x_comp_current = (xi_comp)+(1)*(time);
    % for expansion, position
    x_exp_current = (xi_exp)-(1)*(time);

    % maximum range for the expansion coefficient
    x_final_position = x_exp_current+4*time;

    if x_final_position > 3
        x7 = 3;
        x8 = 3;
        evaluation_point = 3 - x_exp_current;
        y7 = -1+(evaluation_point)/(time);
        y8 = y7;

        % now I need to repeat this for the points at the beggining
        x1 = -3;
        y1 = y7;

        pending = x_final_position-3;
        x2 = -3+pending;
        y2 = 3;
    else
        x7 = x_final_position;
        y7 = 3;

        x8 = 3;
        y8 = 3;
```

```
        x1 = -1;
        y1 = 3;

        x2 = -2;
        y2 = 3;
    end

    x3 = x_comp_current;
    y3 = 3;

    x4 = x_comp_current;
    y4 = 1;

    x5 = x_comp_current;
    y5 = -1;

    x6 = x_exp_current;
    y6 = -1;

    x_points = [x1,x2,x3,x4,x5,x6,x7,x8];
    y_points = [y1,y2,y3,y4,y5,y6,y7,y8];

end
```

*Published with MATLAB® R2018b*

```
%
 ----------------------------------------------------------------------
clc
close all
clear all

% The following code will try to plot everything using the
% Lax-wendroff method

% initial conditions
dx = 0.01;
dt = 3E-3;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = [3*ones(1,length(-3:dx:-1)),
 -1*ones(1,length(-1+dx:dx:1)), 3*ones(1,length(1+dx:dx:3))];

% now lets do the actual algorithm
% 1st for Lax Methods - for 0.25 for testing
time = 1.00;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

% exact solution
[position_e, velocity_e] = exact(time);

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i-1,length(space)-1);

%     % iterator
%     for w=2:+1:+length(space)-1
%
%         % parts to mount the entire equation
%         C = (1/2)*(U(i-1,w+1))^2-(1/2)*(U(i-1,w-1))^2;
%         D = (1/2)*(U(i-1,w)+U(i-1,w+1));
%         E = (1/2)*(U(i-1,w+1))^2-(1/2)*(U(i-1,w))^2;
%         F = (1/2)*(U(i-1,w)+U(i-1,w-1));
%         K = (1/2)*(U(i-1,w))^2-(1/2)*(U(i-1,w-1))^2;
%
%         % mounting entire equation
```

```
%
%           U(i,w) = U(i-1,w)-(1/2)*(B)*(C)+(1/2)*((B)^2)*(D*E-F*K);
%     end

    % iterator
    for w=2:1:length(space)-1

        % parts needed to mount the entire equation
        A = U(i-1,w);
        BB = 0.5*U(i-1,w+1)^2-0.5*U(i-1,w-1)^2;
        C = (1/2)*(U(i-1,w)+U(i-1,w+1));
        D = 0.5*U(i-1,w+1)^2-0.5*U(i-1,w)^2;
        E = (1/2)*(U(i-1,w)+U(i-1,w-1));
        F = 0.5*U(i-1,w)^2-0.5*U(i-1,w-1)^2;

        % mounting the entire equation
        U(i,w) = A-(1/2)*(B)*(BB)+(1/2)*(B^2)*(C*D-E*F);

    end

end

% figure(1)
% plot(space,U(length(tt),:),position_e, velocity_e)
% titlename = ['Lax Wendroff''s Method at ', num2str(time), 's'];
% title(titlename)
% xlabel('position x')
% ylabel('velocity U')
% legend('numerical','exact')
% ylim([-3 4])
% grid on

function [x_points, y_points] = exact(time)
    % initial conditions
    xi_comp = -1;
    xi_exp = 1;

    % for compression, position
    x_comp_current = (xi_comp)+(1)*(time);
    % for expansion, position
    x_exp_current = (xi_exp)-(1)*(time);

    % maximum range for the expansion coefficient
    x_final_position = x_exp_current+4*time;

    if x_final_position > 3
        x7 = 3;
        x8 = 3;
        evaluation_point = 3 - x_exp_current;
        y7 = -1+(evaluation_point)/(time);
        y8 = y7;

        % now I need to repeat this for the points at the beggining
        x1 = -3;
```

```
        y1 = y7;

        pending = x_final_position-3;
        x2 = -3+pending;
        y2 = 3;
    else
        x7 = x_final_position;
        y7 = 3;

        x8 = 3;
        y8 = 3;

        x1 = -1;
        y1 = 3;

        x2 = -2;
        y2 = 3;
    end

    x3 = x_comp_current;
    y3 = 3;

    x4 = x_comp_current;
    y4 = 1;

    x5 = x_comp_current;
    y5 = -1;

    x6 = x_exp_current;
    y6 = -1;

    x_points = [x1,x2,x3,x4,x5,x6,x7,x8];
    y_points = [y1,y2,y3,y4,y5,y6,y7,y8];

end
```

*Published with MATLAB® R2018b*

```
%
 -----------------------------------------------------------------------
clc
close all
clear all

% The following code will try to plot everything using the
% Godunov Method - lets see what happends

% initial conditions
dx = 0.01;
dt = 1.7E-3;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = [3*ones(1,length(-3:dx:-1)),...
    -1*ones(1,length(-1+dx:dx:1)), 3*ones(1,length(1+dx:dx:3))];

% This is for pre-setting the mesh
time = 1.0;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

% exact solution
[position_e, velocity_e] = exact(time);

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i-1,length(space)-1);

    % iterator
    for w=2:1:length(space)-1
        U(i,w) = U(i-1,w)-B*(F(U(i-1,w),U(i-1,w+1))-
F(U(i-1,w-1),U(i-1,w)));
    end
end

% figure(1)
% plot(space,U(length(tt),:),position_e, velocity_e)
% titlename = ['Godunov''s Method''s at ', num2str(time), 's'];
% title(titlename)
% xlabel('position x')
```

```
% ylabel('velocity U')
% legend('numerical','exact')
% ylim([-3 4])
% grid on

% A is the smaller one, B is the larger one
function F_ans = F(A,B)
    C = (A+B)/2;

    if A > B
        if C > 0
            F_ans = (1/2)*A^2;
        elseif C < 0
            F_ans = (1/2)*B^2;
        end
    elseif A < B
        if ((A < 0) && (B>0))
            F_ans = 0;
        elseif C > 0
            F_ans = (1/2)*A^2;
        elseif C < 0
            F_ans = (1/2)*B^2;
        end
    else
        F_ans = (1/2)*B^2;
    end
end

function [x_points, y_points] = exact(time)
    % initial conditions
    xi_comp = -1;
    xi_exp = 1;

    % for compression, position
    x_comp_current = (xi_comp)+(1)*(time);
    % for expansion, position
    x_exp_current = (xi_exp)-(1)*(time);

    % maximum range for the expansion coefficient
    x_final_position = x_exp_current+4*time;

    if x_final_position > 3
        x7 = 3;
        x8 = 3;
        evaluation_point = 3 - x_exp_current;
        y7 = -1+(evaluation_point)/(time);
        y8 = y7;

        % now I need to repeat this for the points at the beggining
        x1 = -3;
        y1 = y7;

        pending = x_final_position-3;
        x2 = -3+pending;
```

```
        y2 = 3;
    else
        x7 = x_final_position;
        y7 = 3;

        x8 = 3;
        y8 = 3;

        x1 = -1;
        y1 = 3;

        x2 = -2;
        y2 = 3;
    end

    x3 = x_comp_current;
    y3 = 3;

    x4 = x_comp_current;
    y4 = 1;

    x5 = x_comp_current;
    y5 = -1;

    x6 = x_exp_current;
    y6 = -1;

    x_points = [x1,x2,x3,x4,x5,x6,x7,x8];
    y_points = [y1,y2,y3,y4,y5,y6,y7,y8];

end
```

*Published with MATLAB® R2018b*

```
%
 --------------------------------------------------------------------------
clc
close all
clear all

% The following code will try to plot everything using the
% roe Method - lets see what happends

% initial conditions
dx = 0.01;
dt = 1E-4;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = [3*ones(1,length(-3:dx:-1)),...
    -1*ones(1,length(-1+dx:dx:1)), 3*ones(1,length(1+dx:dx:3))];

% This is for pre-setting the mesh
time = 0.250;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

% exact solution
[position_e, velocity_e] = exact(time);

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i-1,length(space)-1);

    % iterator
    for w=2:1:length(space)-1
        U(i,w) = U(i-1,w)-B*(+F(U(i-1,w),U(i-1,w+1))-
F(U(i-1,w-1),U(i-1,w)));
    end
end

% figure(1)
% plot(space,U(length(tt),:),position_e, velocity_e)
% titlename = ['Roe''s Method at ', num2str(time), 's'];
% title(titlename)
% xlabel('position x')
```

```
% ylabel('velocity U')
% legend('numerical','exact')
% ylim([-3 4])

function F_ans = F(A,B)

    if A ~= B
        Ubar1 = (A+B)/2;
    elseif A == B
        Ubar1 = A;
    end

    e = max(0,(B-A)/2);

    if Ubar1 >= e
        Ubar2 = Ubar1;
    elseif Ubar1<e
        Ubar2 = e;
    end

    F_ans = (1/2)*(0.5*A^2+0.5*B^2)-(1/2)*abs(Ubar2)*(B-A);
end

function [x_points, y_points] = exact(time)
    % initial conditions
    xi_comp = -1;
    xi_exp = 1;

    % for compression, position
    x_comp_current = (xi_comp)+(1)*(time);
    % for expansion, position
    x_exp_current = (xi_exp)-(1)*(time);

    % maximum range for the expansion coefficient
    x_final_position = x_exp_current+4*time;

    if x_final_position > 3
        x7 = 3;
        x8 = 3;
        evaluation_point = 3 - x_exp_current;
        y7 = -1+(evaluation_point)/(time);
        y8 = y7;

        % now I need to repeat this for the points at the beggining
        x1 = -3;
        y1 = y7;

        pending = x_final_position-3;
        x2 = -3+pending;
        y2 = 3;
    else
        x7 = x_final_position;
        y7 = 3;
```

```
        x8 = 3;
        y8 = 3;

        x1 = -1;
        y1 = 3;

        x2 = -2;
        y2 = 3;
    end

    x3 = x_comp_current;
    y3 = 3;

    x4 = x_comp_current;
    y4 = 1;

    x5 = x_comp_current;
    y5 = -1;

    x6 = x_exp_current;
    y6 = -1;

    x_points = [x1,x2,x3,x4,x5,x6,x7,x8];
    y_points = [y1,y2,y3,y4,y5,y6,y7,y8];

end
```

*Published with MATLAB® R2018b*

```
clc
close all
clear all

% The following code will try to plot everything using the
% roe Method 2nd order w/o limiter - lets see what happends

% initial conditions
dx = 0.01;
dt = 1E-4;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = [3*ones(1,length(-3:dx:-1)),...
    -1*ones(1,length(-1+dx:dx:1)), 3*ones(1,length(1+dx:dx:3))];

% This is for pre-setting the mesh
time = 1;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

% exact solution
[position_e, velocity_e] = exact(time);

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i-1,length(space)-1);
    U(i,length(space)-1) = U(i-1,length(space)-2);
    U(i,2) = U(i-1,length(space)-3);

    % iterator
    for w=3:1:length(space)-2
        U(i,w) = U(i-1,w)-B*(F(U(i-1,w-1),U(i-1,w),U(i-1,w+1),U(i-1,w
+2))-F(U(i-1,w-2),U(i-1,w-1),U(i-1,w),U(i-1,w+1)));
    end
end

% figure(1)
% plot(space,U(length(tt),:),position_e, velocity_e)
% titlename = ['Roe''s Method 2nd Order (No Flux Limiter) ',
 num2str(time), 's'];
% title(titlename)
```

```
% xlabel('position x')
% ylabel('velocity U')
% legend('numerical','exact')
% ylim([-3 4])
% grid on

function F_ans = F(A,B,C,D)

    U_l = B+(1/2)*(B-A);
    U_r = C-(1/2)*(D-C);

    if U_l ~= U_r
        U_bar = (0.5*U_r^2-0.5*U_l^2)/(U_r-U_l);
    else
        U_bar = U_l;
    end
    %U_bar = (0.5*U_r^2-0.5*U_l^2)/(U_r-U_l);

    F_ans = (1/2)*(0.5*U_l^2+0.5*U_r^2-abs(U_bar)*(U_r-U_l));

end

function [x_points, y_points] = exact(time)
    % initial conditions
    xi_comp = -1;
    xi_exp = 1;

    % for compression, position
    x_comp_current = (xi_comp)+(1)*(time);
    % for expansion, position
    x_exp_current = (xi_exp)-(1)*(time);

    % maximum range for the expansion coefficient
    x_final_position = x_exp_current+4*time;

    if x_final_position > 3
        x7 = 3;
        x8 = 3;
        evaluation_point = 3 - x_exp_current;
        y7 = -1+(evaluation_point)/(time);
        y8 = y7;

        % now I need to repeat this for the points at the beggining
        x1 = -3;
        y1 = y7;

        pending = x_final_position-3;
        x2 = -3+pending;
        y2 = 3;
    else
        x7 = x_final_position;
        y7 = 3;

        x8 = 3;
```

```
        y8 = 3;

        x1 = -1;
        y1 = 3;

        x2 = -2;
        y2 = 3;
    end

    x3 = x_comp_current;
    y3 = 3;

    x4 = x_comp_current;
    y4 = 1;

    x5 = x_comp_current;
    y5 = -1;

    x6 = x_exp_current;
    y6 = -1;

    x_points = [x1,x2,x3,x4,x5,x6,x7,x8];
    y_points = [y1,y2,y3,y4,y5,y6,y7,y8];

end
```

*Published with MATLAB® R2018b*

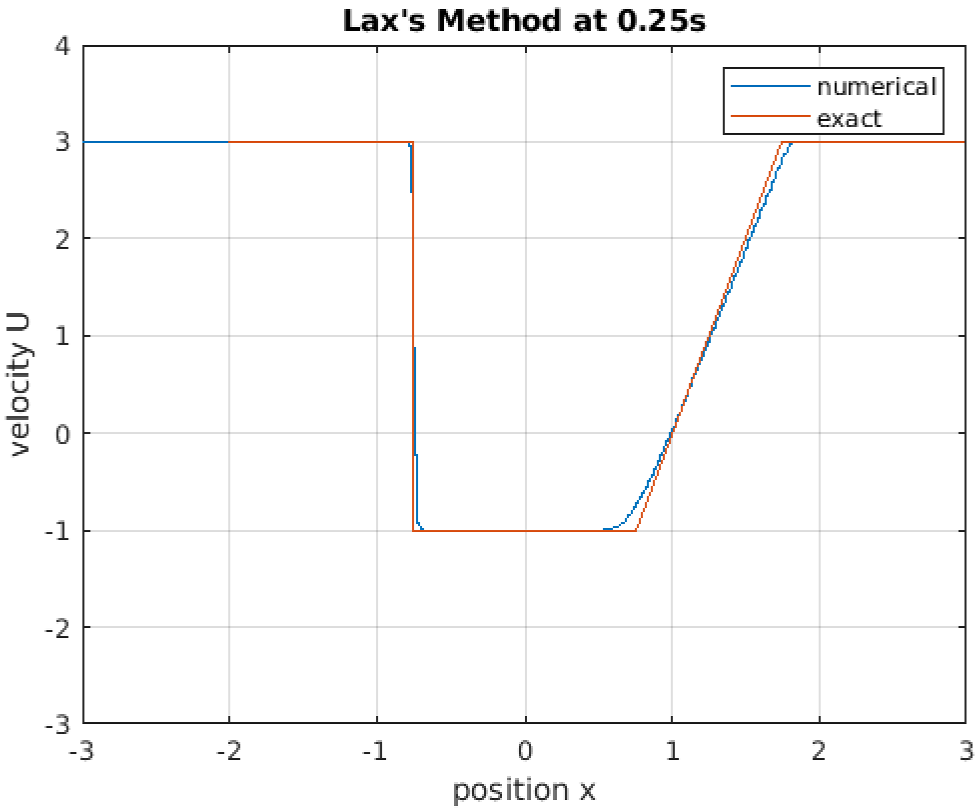
```
%
 ----------------------------------------------------------------------
clc
close all
clear all

% The following code will try to plot everything using the
% roe Method/2nd order with limiter - lets see what happends

% initial conditions
dx = 0.01;
dt = 1E-4;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = [3*ones(1,length(-3:dx:-1)),...
    -1*ones(1,length(-1+dx:dx:1)), 3*ones(1,length(1+dx:dx:3))];

% This is for pre-setting the mesh
time = 1.00;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

% exact solution
[position_e, velocity_e] = exact(time);

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i-1,length(space)-1);
    U(i,length(space)-1) = U(i-1,length(space)-2);
    U(i,2) = U(i-1,length(space)-3);

    % iterator
    for w=3:1:length(space)-2
        U(i,w) = U(i-1,w)-B*(F(U(i-1,w-1),U(i-1,w),U(i-1,w+1),U(i-1,w
+2))-F(U(i-1,w-2),U(i-1,w-1),U(i-1,w),U(i-1,w+1)));
    end
end

% figure(1)
% plot(space,U(length(tt),:),position_e, velocity_e)
```

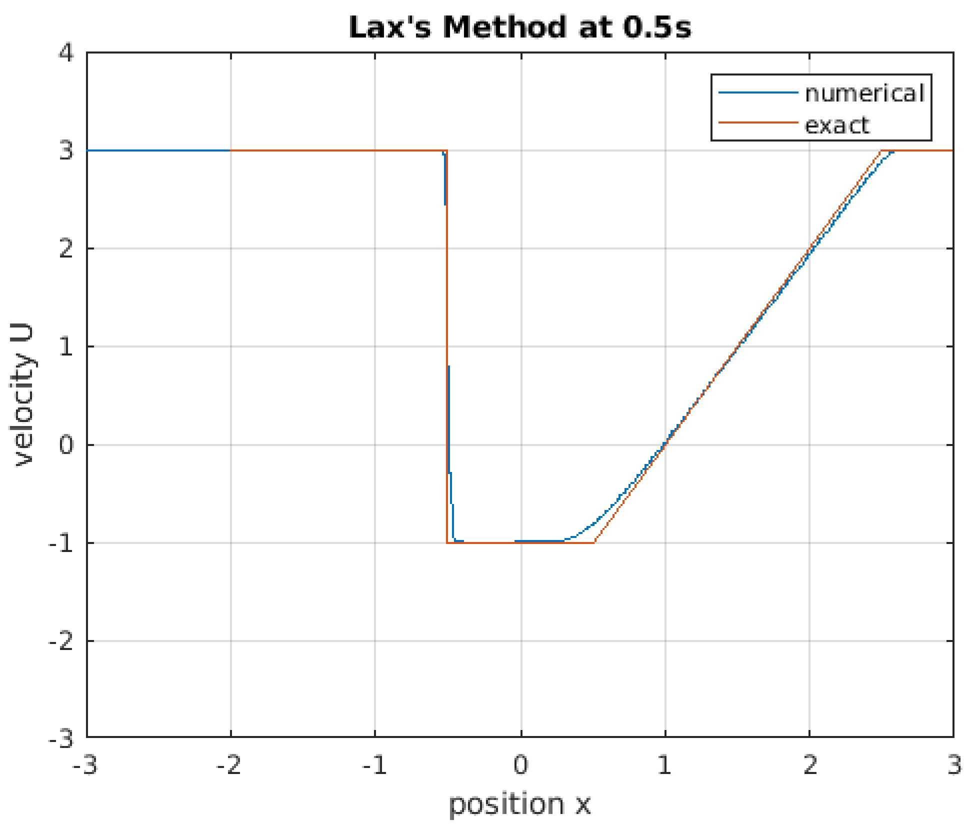
```
% titlename = ['Roe''s Method 2nd Order (Flux Limiter) at ',
 num2str(time), 's'];
% title(titlename)
% xlabel('position x')
% ylabel('velocity U')
% legend('numerical','exact')
% ylim([-3 4])
% grid on

function F_ans = F(A,B,C,D)

    % flux limiter
    r_limit = (C-B)/(B-A);
    limiter = max(0,min(1,r_limit));

    U_l = B+(1/2)*limiter*(B-A);
    U_r = C-(1/2)*limiter*(D-C);

    if U_l ~= U_r
        U_bar = (0.5*U_r^2-0.5*U_l^2)/(U_r-U_l);
    else
        U_bar = U_l;
    end
    %U_bar = (0.5*U_r^2-0.5*U_l^2)/(U_r-U_l);

    F_ans = (1/2)*(0.5*U_l^2+0.5*U_r^2-abs(U_bar)*(U_r-U_l));

end

function [x_points, y_points] = exact(time)
    % initial conditions
    xi_comp = -1;
    xi_exp = 1;

    % for compression, position
    x_comp_current = (xi_comp)+(1)*(time);
    % for expansion, position
    x_exp_current = (xi_exp)-(1)*(time);

    % maximum range for the expansion coefficient
    x_final_position = x_exp_current+4*time;

    if x_final_position > 3
        x7 = 3;
        x8 = 3;
        evaluation_point = 3 - x_exp_current;
        y7 = -1+(evaluation_point)/(time);
        y8 = y7;

        % now I need to repeat this for the points at the beggining
        x1 = -3;
        y1 = y7;

        pending = x_final_position-3;
```

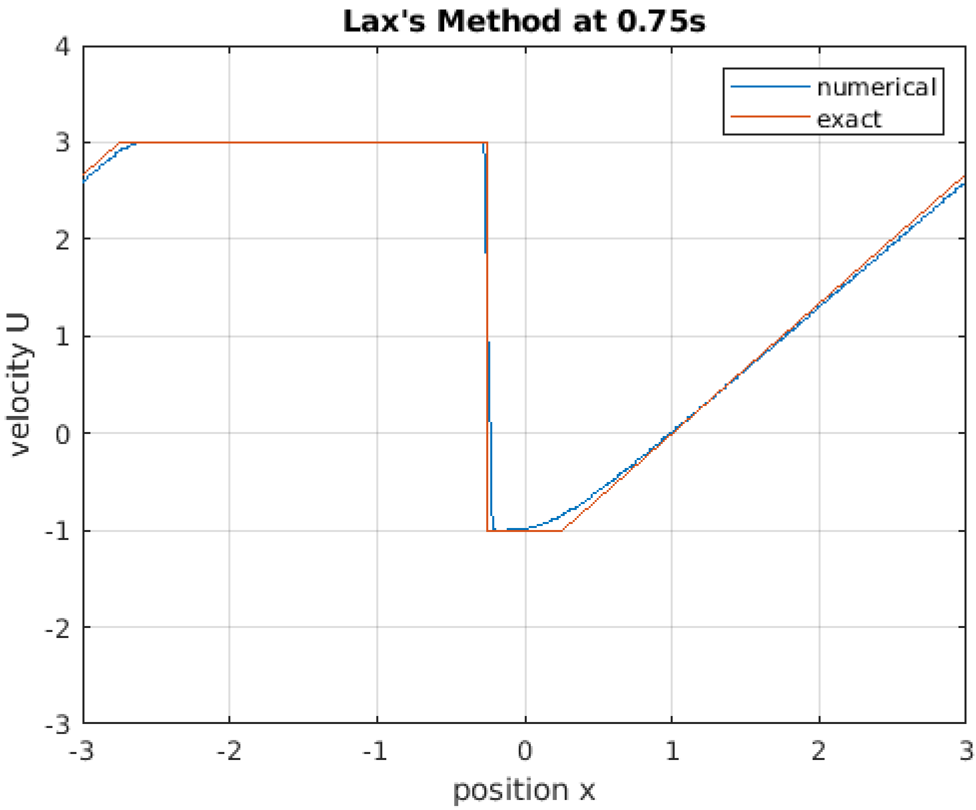
```
        x2 = -3+pending;
        y2 = 3;
    else
        x7 = x_final_position;
        y7 = 3;

        x8 = 3;
        y8 = 3;

        x1 = -1;
        y1 = 3;

        x2 = -2;
        y2 = 3;
    end

    x3 = x_comp_current;
    y3 = 3;

    x4 = x_comp_current;
    y4 = 1;

    x5 = x_comp_current;
    y5 = -1;

    x6 = x_exp_current;
    y6 = -1;

    x_points = [x1,x2,x3,x4,x5,x6,x7,x8];
    y_points = [y1,y2,y3,y4,y5,y6,y7,y8];

end
```

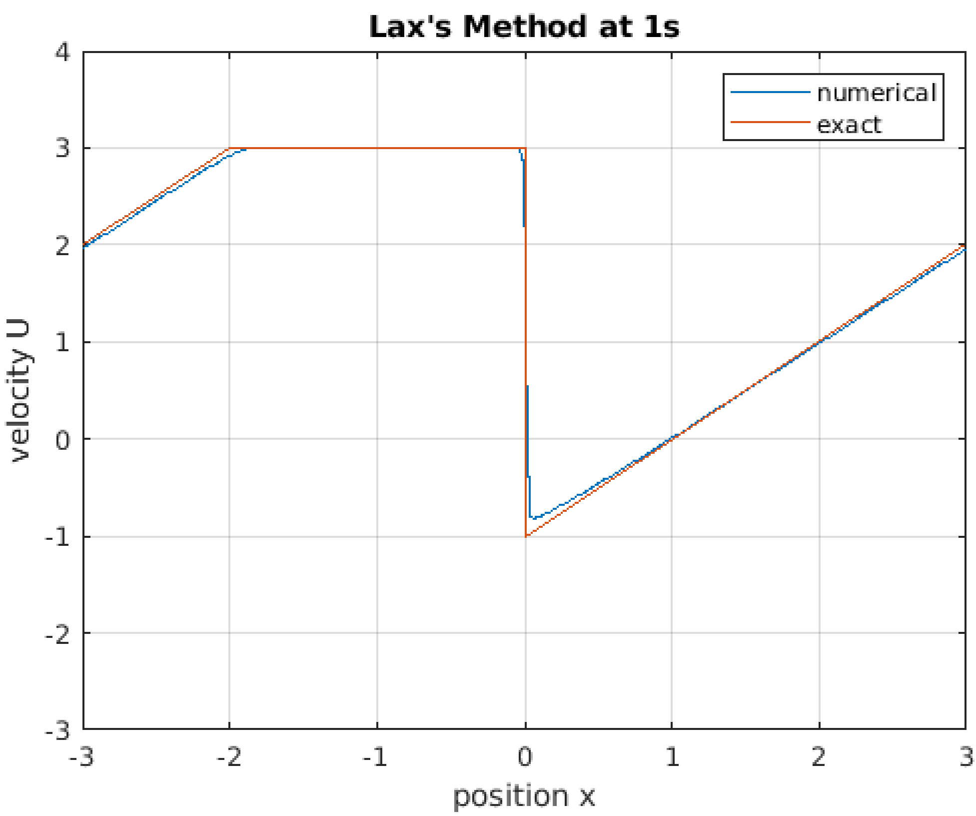
*Published with MATLAB® R2018b*

(a) Plots for Lax method -

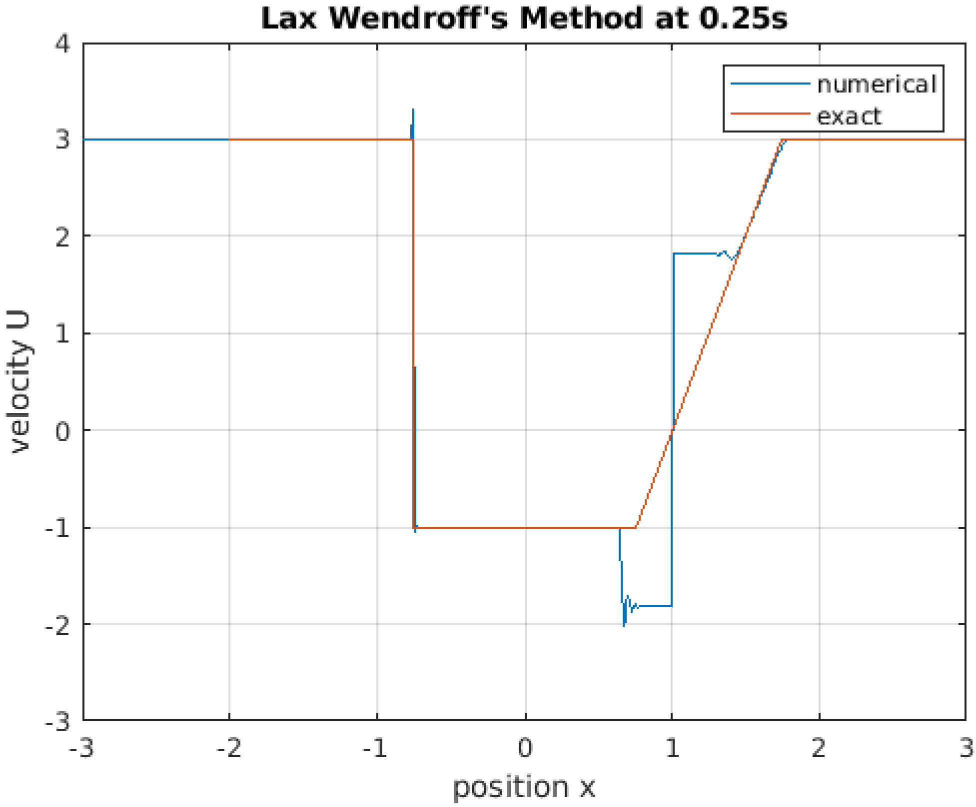


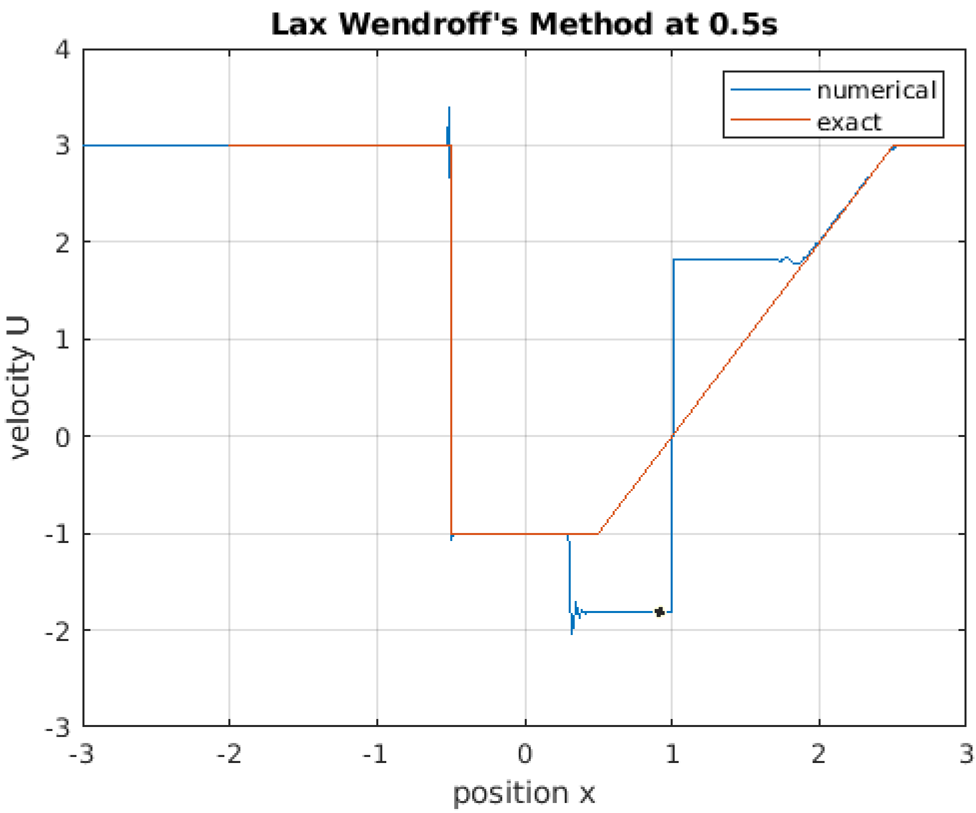



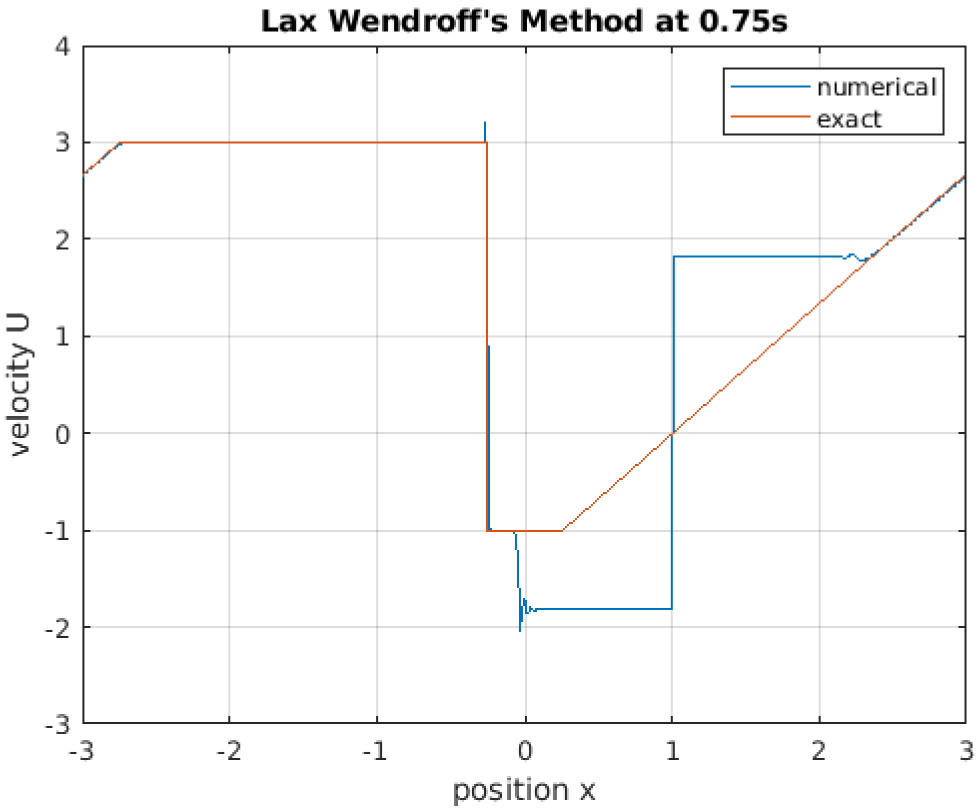

Lax's Method at 0.75s
numerical
exact
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

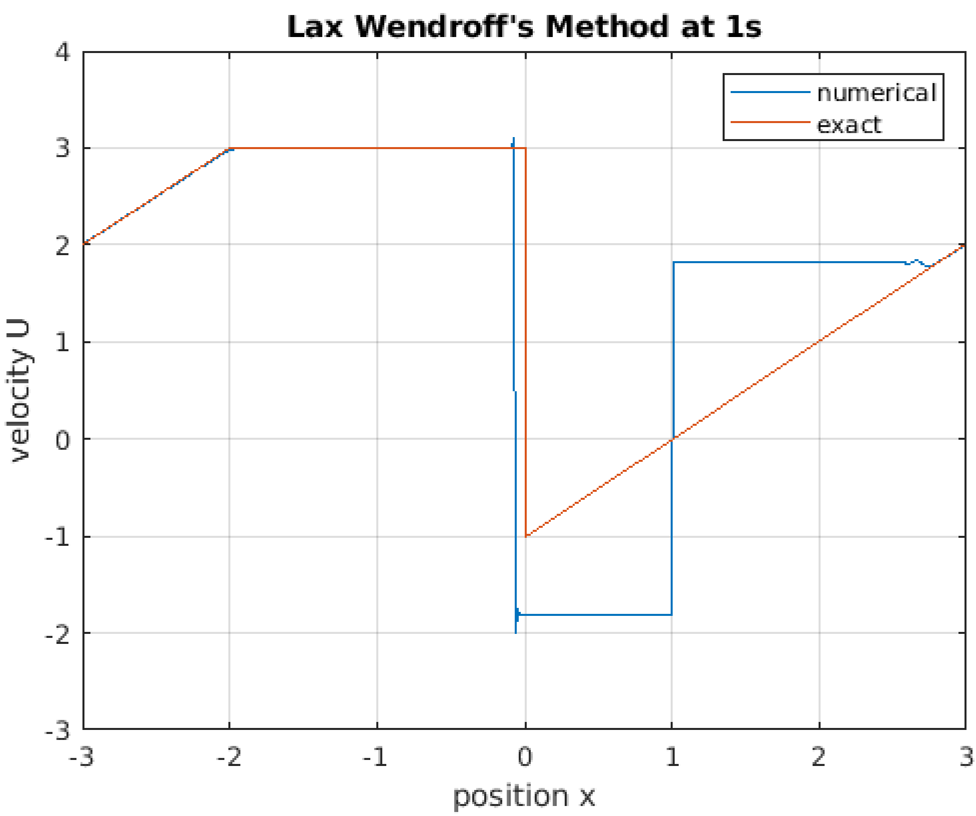

Lax's Method at 1s
numerical
exact
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

(b) Plots for lax-wendroff -

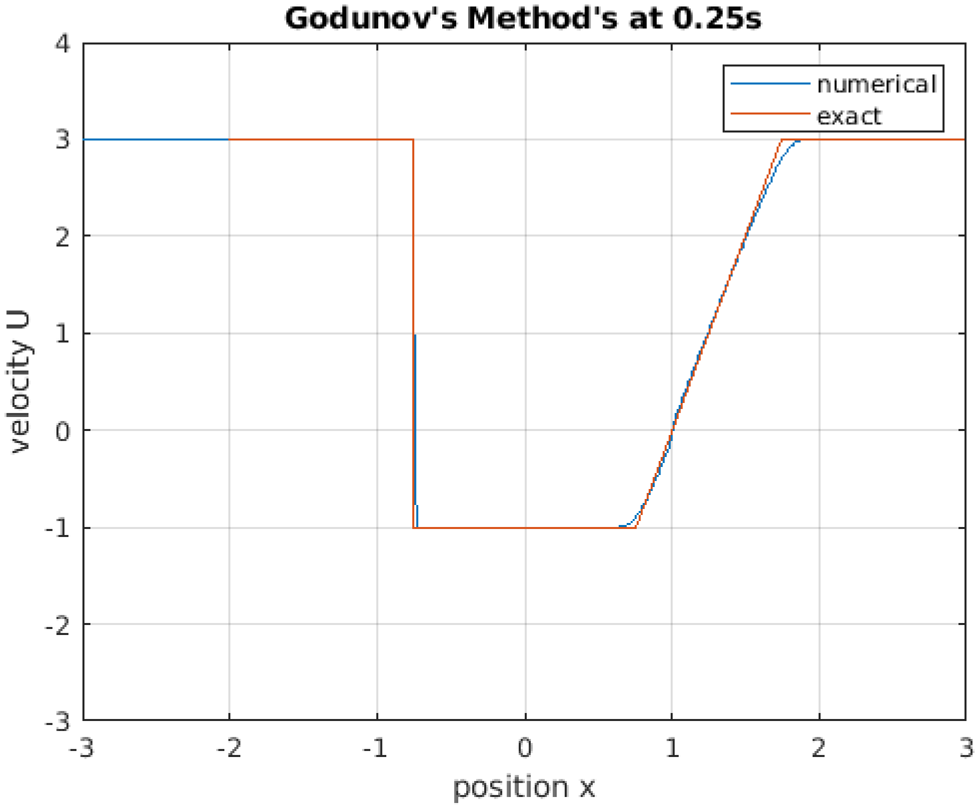


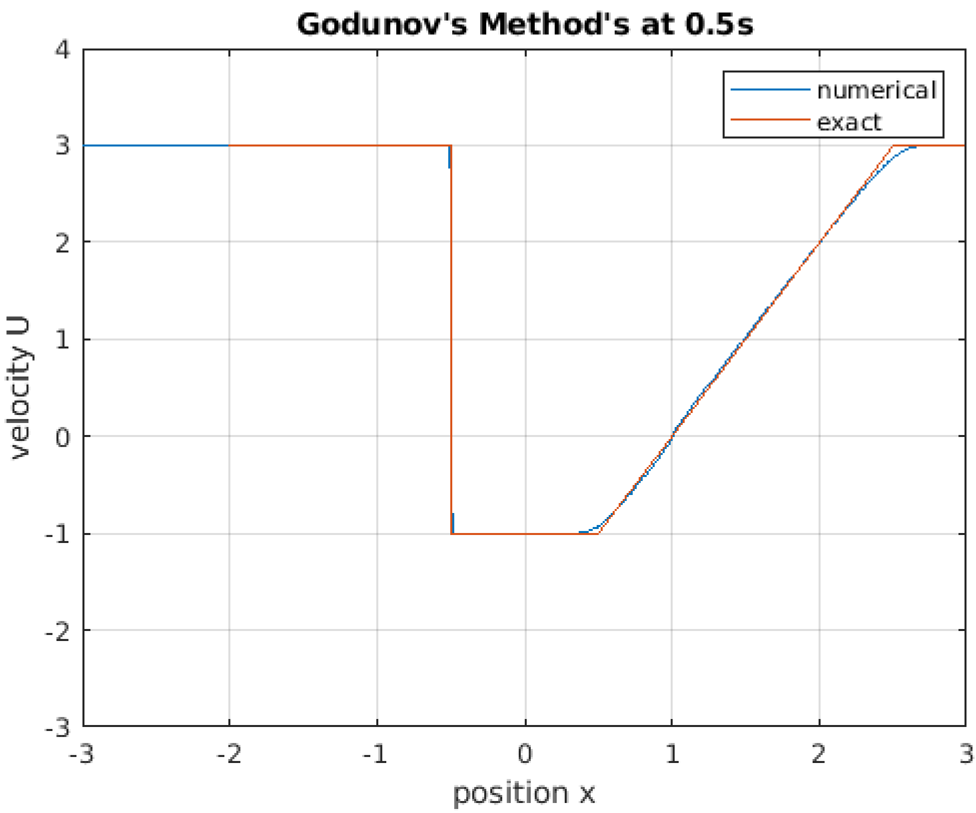



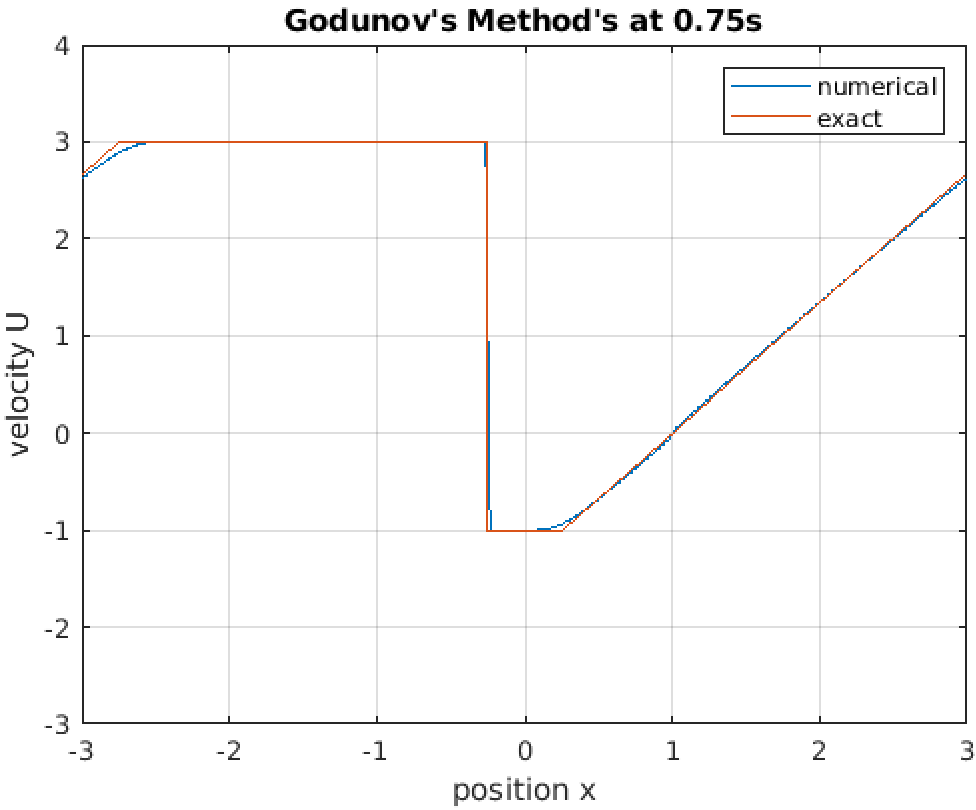

Lax Wendroff's Method at 0.75s
numerical
exact
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

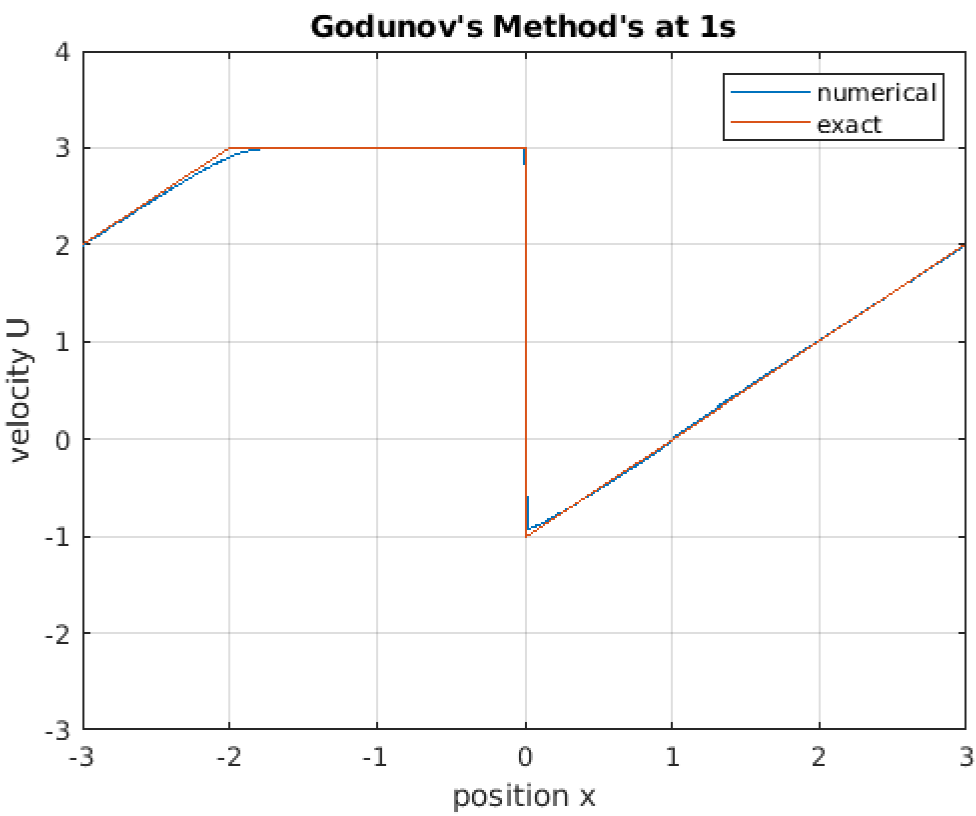

Lax Wendroff's Method at 1s
numerical
exact
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

(c) Plots for Godunov -

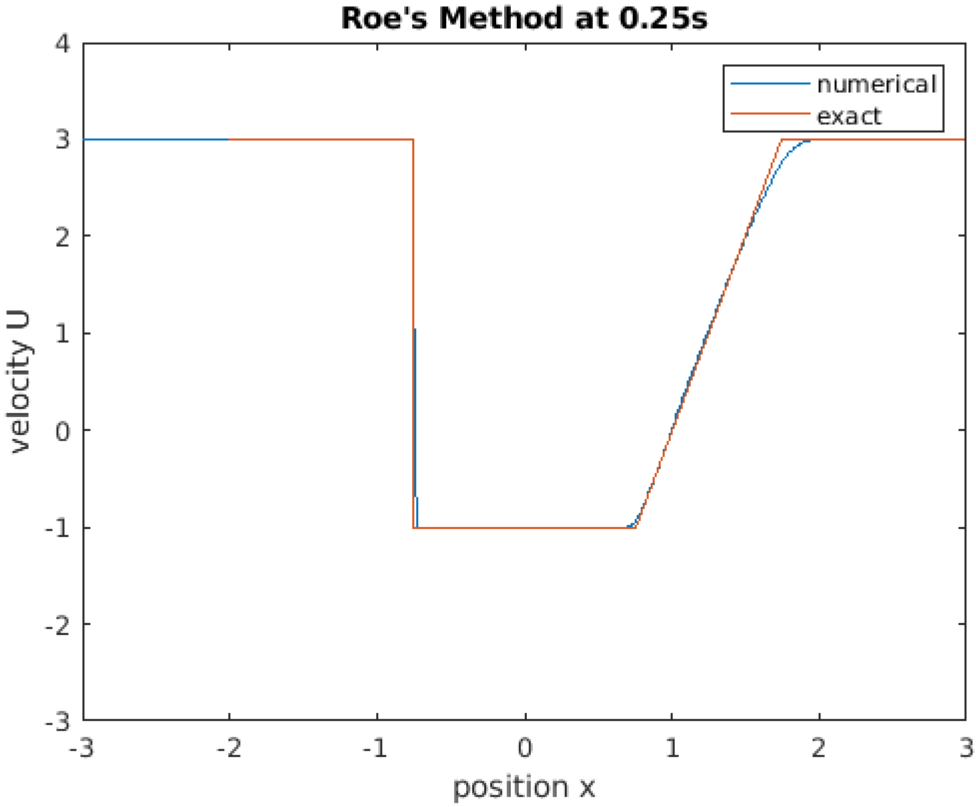


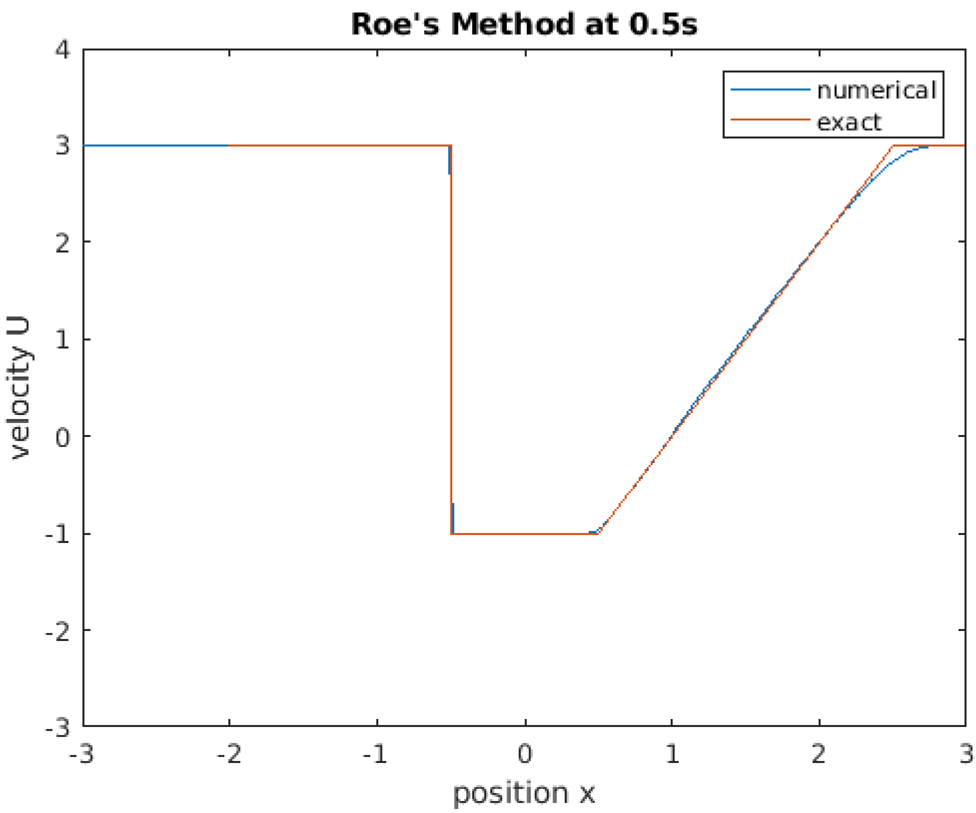



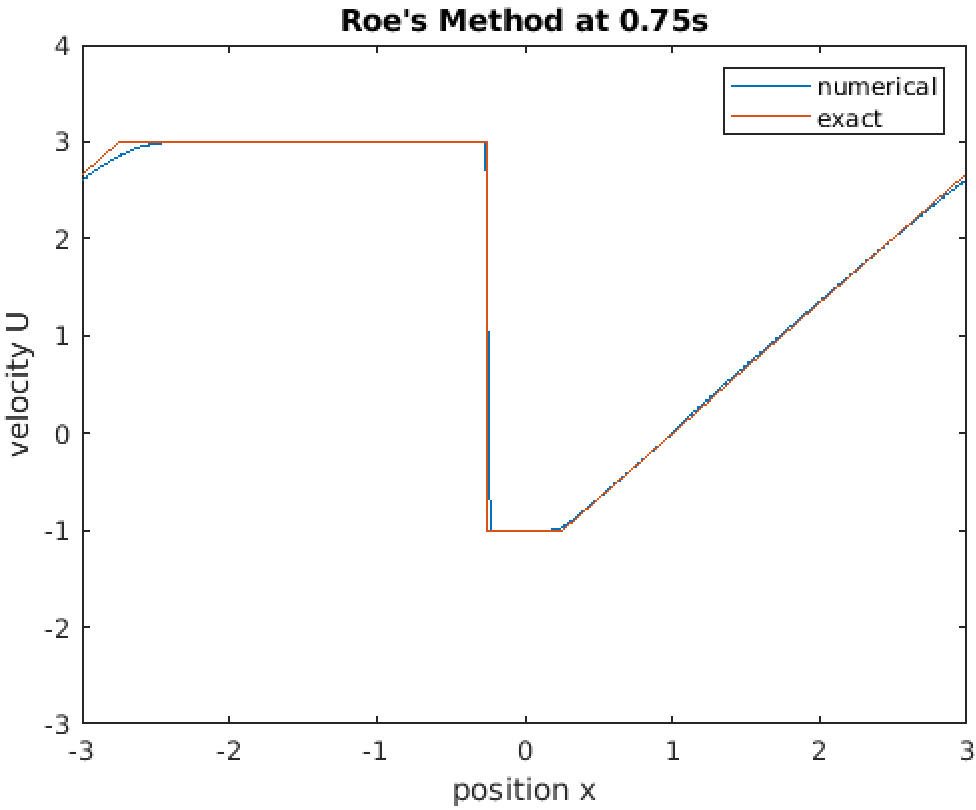

Godunov's Method's at 0.75s
numerical
exact
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

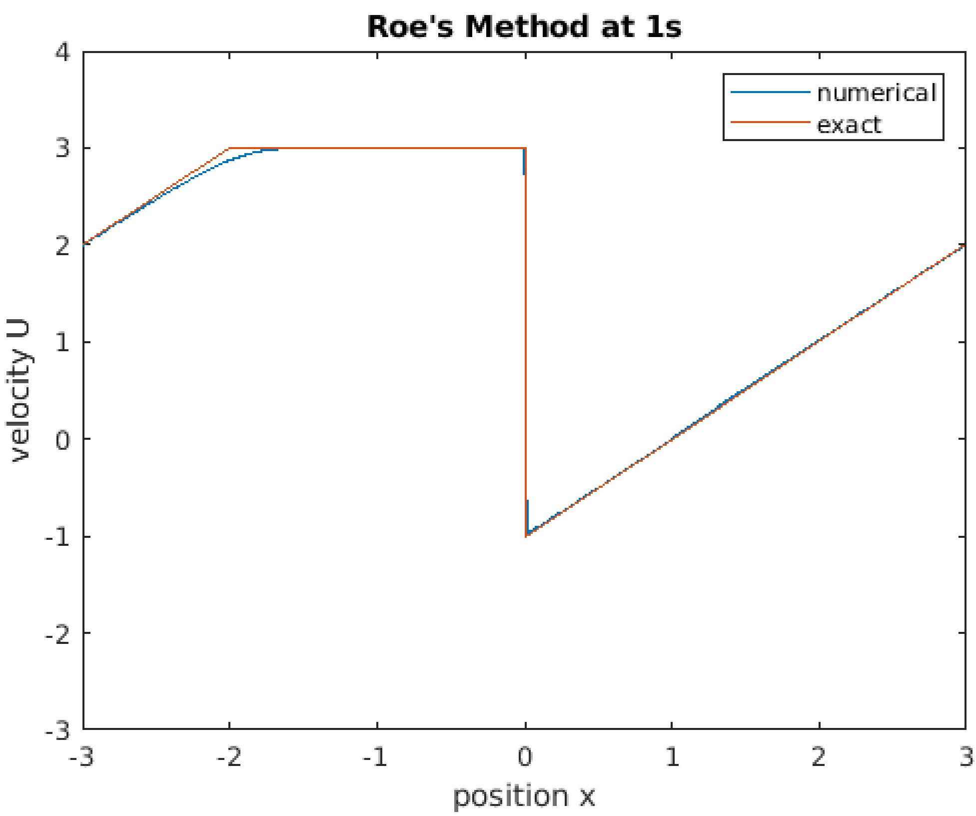

Godunov's Method's at 1s
numerical
exact
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

(d) Plots for Roe -

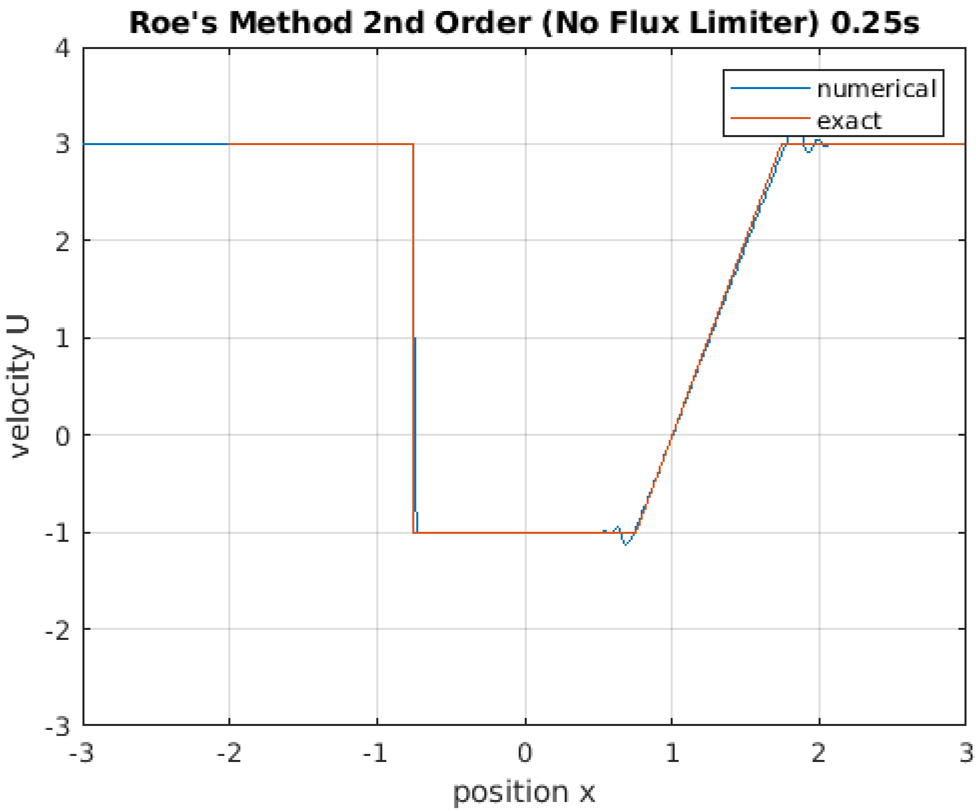


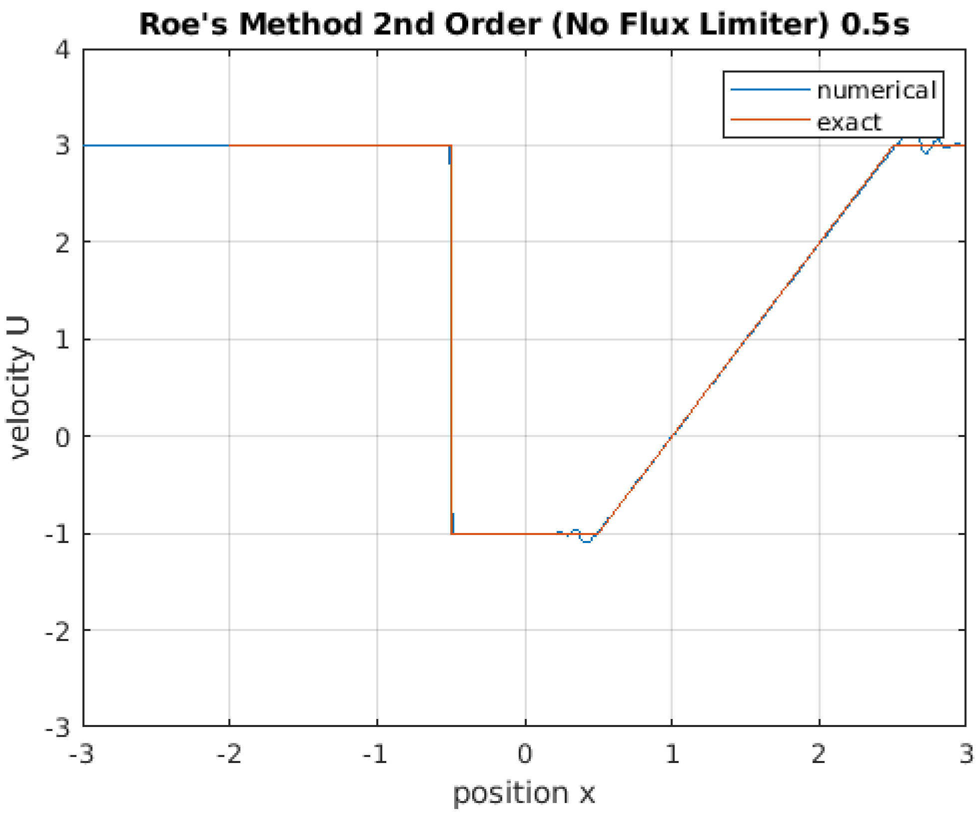



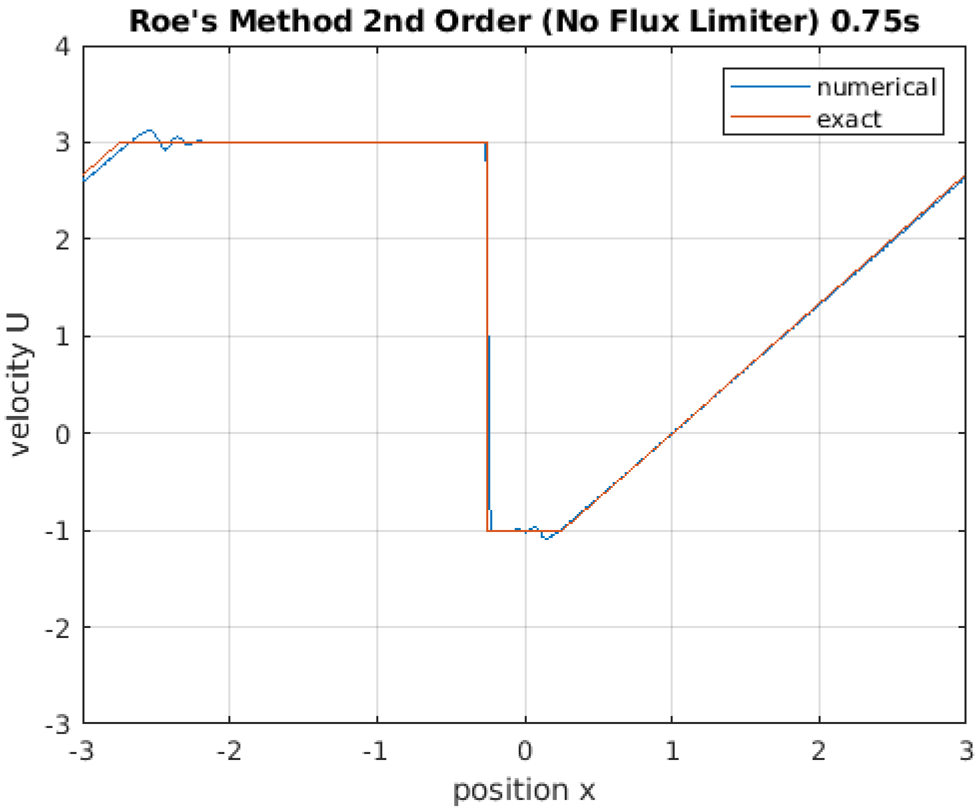

Roe's Method at 0.75s
numerical
exact
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3


Roe's Method at 1s
numerical
exact
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

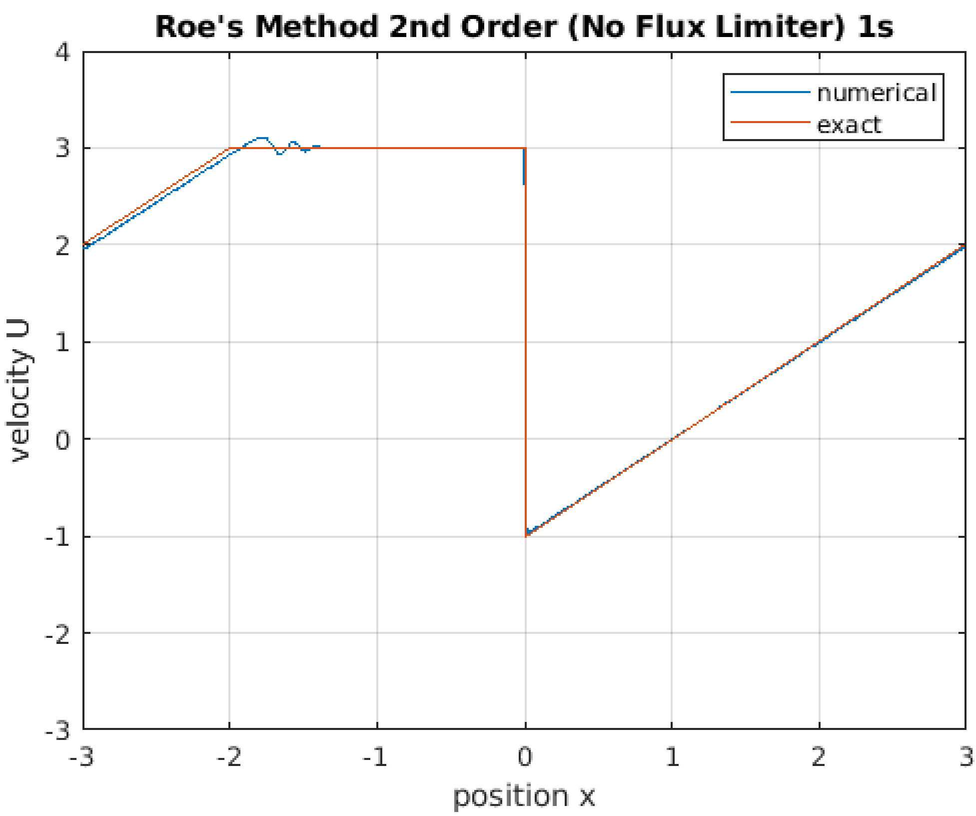
(f) Roe 2nd order w/o flux limiter



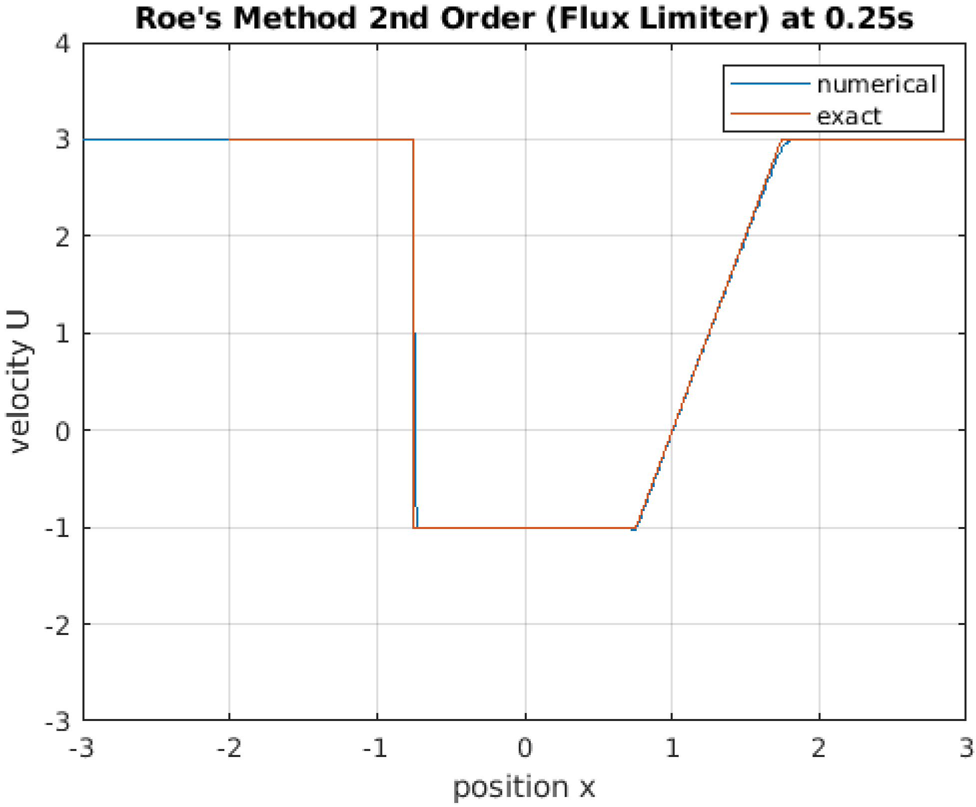


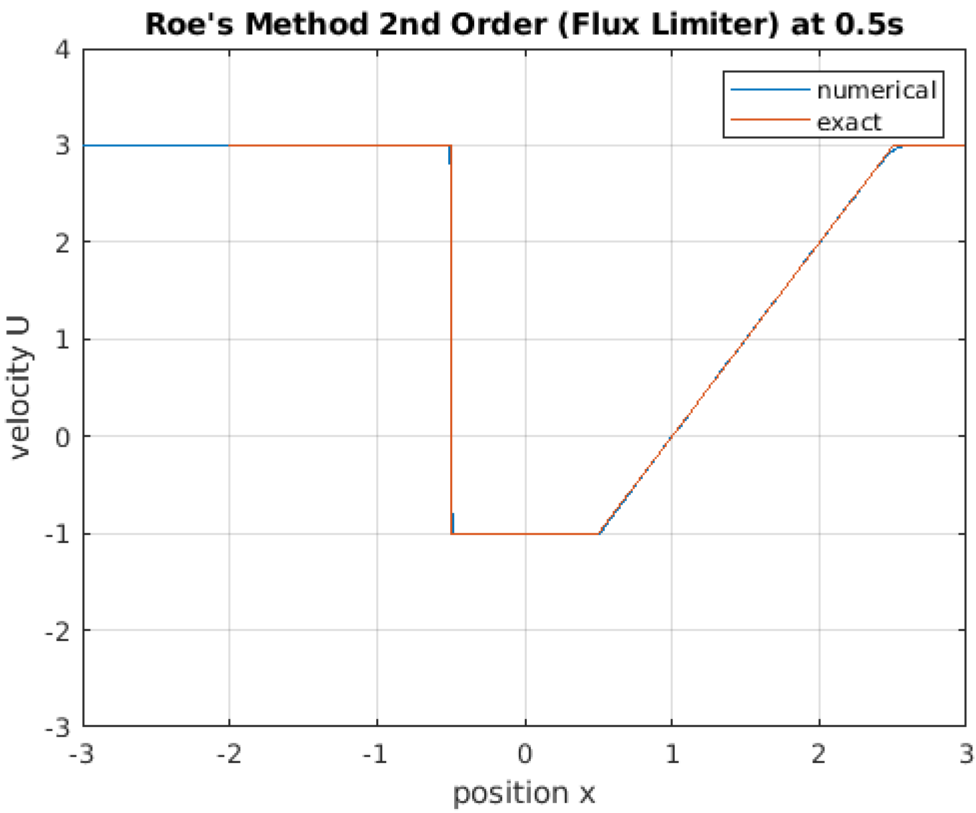
Roe's Method 2nd Order (No Flux Limiter) 0.75s
numerical
exact
velocity U
position x

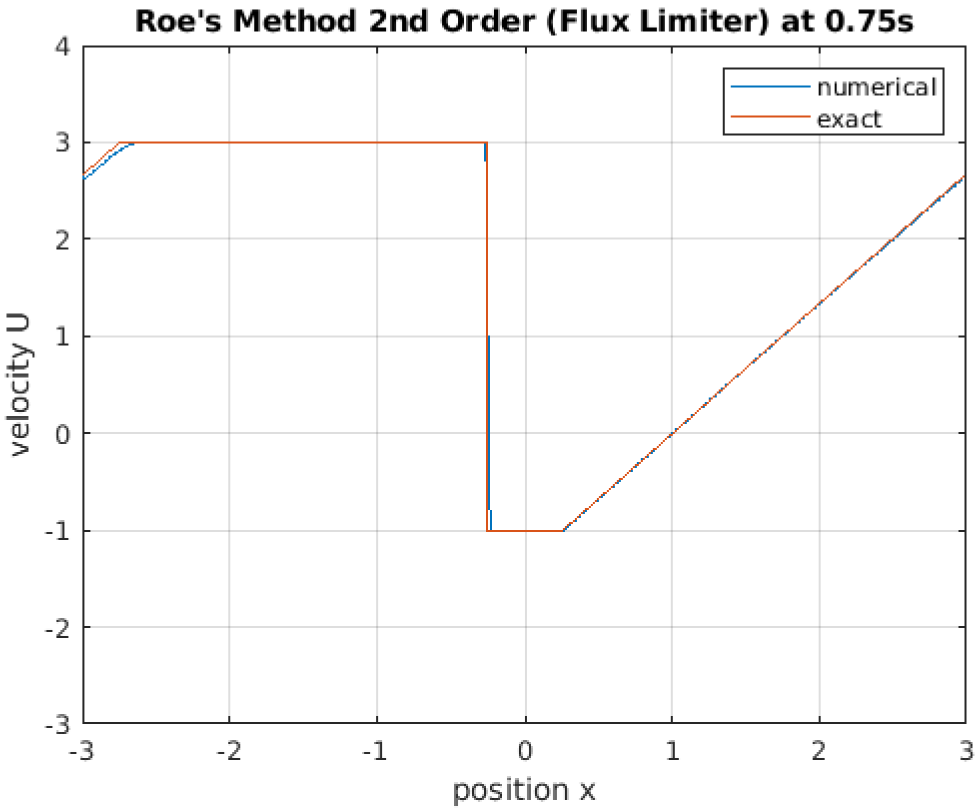



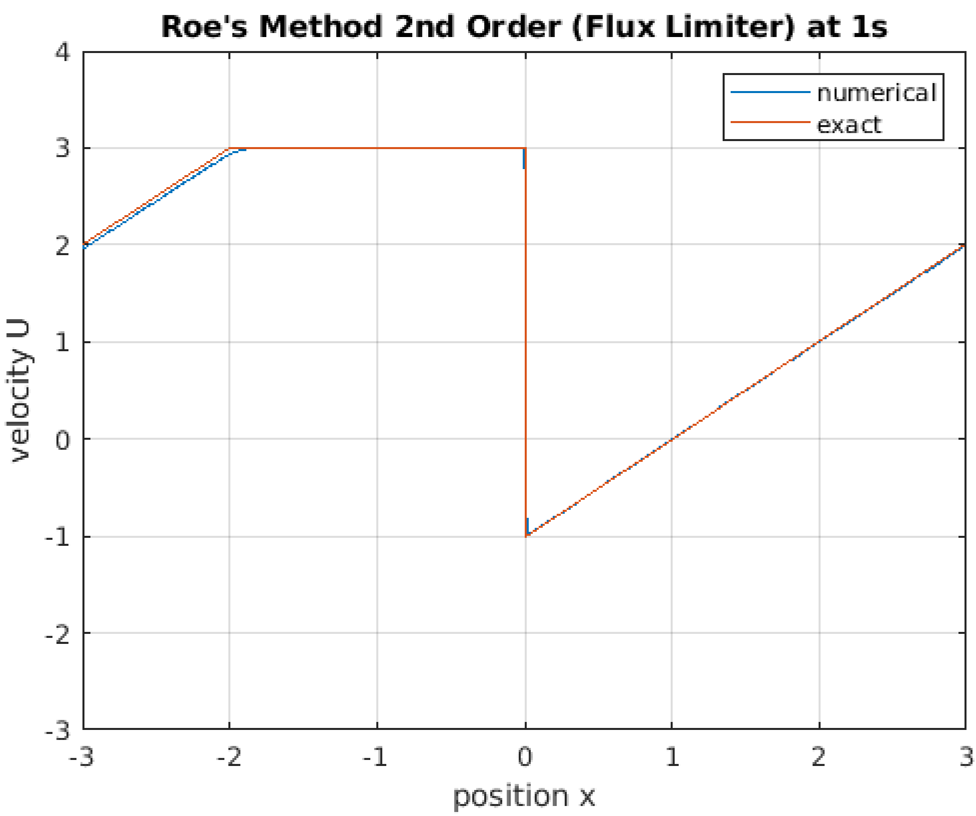
Roe's Method 2nd Order (No Flux Limiter) 1s
numerical
exact
velocity U
position x

(f) Roe 2nd order w/ flux limiter




Roe's Method 2nd Order (Flux Limiter) at 0.75s
numerical
exact
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3


Roe's Method 2nd Order (Flux Limiter) at 1s
numerical
exact
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

```
clc
close all
clear all

% The following code will try to plot everything using the
% Lax method - for this case I will be using the updated initial
% condition(s)

% initial conditions
dx = 0.01;
dt = 3E-3;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = exp(-0.5*(space).^2);

% now lets do the actual algorithm
% 1st for Lax Methods - for 0.25 for testing

time = 3.25;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

% exact solution
%[position_e, velocity_e] = exact(time);

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i,length(space));

%     % iterator
%     for w=2:1:length(space)-1
%         U(i,w) = 0.5*(U(i-1,w+1)+B*(-0.5*U(i-1,w
+1)^2+0.5*U(i-1,w-1)^2)+U(i-1,w-1));
%     end

    % iterator
    % lets split everything and then regroup it again. I think that's
 the
    % best way to work with this
    for w=2:1:length(space)-1
        A = U(i-1,w+1)+U(i-1,w-1);
```

```
        BB = (U(i-1,w+1)^2)/2;
        C = (U(i-1,w-1)^2)/2;

        U(i,w) = (1/2)*A-(B/2)*(BB-C);
    end
end

% figure(1)
% plot(space,U(length(tt),:))
% titlename = ['Lax''s Method at ', num2str(time), 's'];
% title(titlename)
% xlabel('position x')
% ylabel('velocity U')
% legend('numerical')
% ylim([-3 4])
% grid on
```

*Published with MATLAB® R2018b*

```
clc
close all
clear all

% The following code will try to plot everything using the
% Lax-wendroff method - for this case I will be using
% the updated initial conditions

% initial conditions
dx = 0.01;
dt = 3E-3;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = exp(-0.5*(space).^2);

% now lets do the actual algorithm
% 1st for Lax Methods - for 0.25 for testing
time = 3.25;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i-1,length(space)-1);

%      % iterator
%      for w=2:+1:+length(space)-1
%
%          % parts to mount the entire equation
%          C = (1/2)*(U(i-1,w+1))^2-(1/2)*(U(i-1,w-1))^2;
%          D = (1/2)*(U(i-1,w)+U(i-1,w+1));
%          E = (1/2)*(U(i-1,w+1))^2-(1/2)*(U(i-1,w))^2;
%          F = (1/2)*(U(i-1,w)+U(i-1,w-1));
%          K = (1/2)*(U(i-1,w))^2-(1/2)*(U(i-1,w-1))^2;
%
%          % mounting entire equation
%
%          U(i,w) = U(i-1,w)-(1/2)*(B)*(C)+(1/2)*((B)^2)*(D*E-F*K);
%      end

    % iterator
```

```
    for w=2:1:length(space)-1

        % parts needed to mount the entire equation
        A = U(i-1,w);
        BB = 0.5*U(i-1,w+1)^2-0.5*U(i-1,w-1)^2;
        C = (1/2)*(U(i-1,w)+U(i-1,w+1));
        D = 0.5*U(i-1,w+1)^2-0.5*U(i-1,w)^2;
        E = (1/2)*(U(i-1,w)+U(i-1,w-1));
        F = 0.5*U(i-1,w)^2-0.5*U(i-1,w-1)^2;

        % mounting the entire equation
        U(i,w) = A-(1/2)*(B)*(BB)+(1/2)*(B^2)*(C*D-E*F);

    end

end

% figure(1)
% plot(space,U(length(tt),:))
% titlename = ['Lax Wendroff''s Method at ', num2str(time), 's'];
% title(titlename)
% xlabel('position x')
% ylabel('velocity U')
% legend('numerical')
% ylim([-3 4])
% grid on
```

*Published with MATLAB® R2018b*

```
clc
close all
clear all

% The following code will try to plot everything using the
% Godunov Method - lets see what happends. For this one we are
% changing the boundary condition so there is that to consider

% initial conditions
dx = 0.01;
dt = 1.7E-3;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = exp(-0.5*(space).^2);

% This is for pre-setting the mesh
time = 3.25;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i-1,length(space)-1);

    % iterator
    for w=2:1:length(space)-1
        U(i,w) = U(i-1,w)-B*(F(U(i-1,w),U(i-1,w+1))-
F(U(i-1,w-1),U(i-1,w)));
    end
end

% figure(1)
% plot(space,U(length(tt),:))
% titlename = ['Godunov''s Method''s at ', num2str(time), 's'];
% title(titlename)
% xlabel('position x')
% ylabel('velocity U')
% legend('numerical')
% ylim([-3 4])
% grid on
```

```
% A is the smaller one, B is the larger one
function F_ans = F(A,B)
    C = (A+B)/2;

    if A > B
        if C > 0
            F_ans = (1/2)*A^2;
        elseif C < 0
            F_ans = (1/2)*B^2;
        end
    elseif A < B
        if ((A < 0) && (B>0))
            F_ans = 0;
        elseif C > 0
            F_ans = (1/2)*A^2;
        elseif C < 0
            F_ans = (1/2)*B^2;
        end
    else
        F_ans = (1/2)*B^2;
    end
end
```

*Published with MATLAB® R2018b*

```
%
 ----------------------------------------------------------------------
clc
close all
clear all

% The following code will try to plot everything using the
% roe Method - lets see what happends. This is with an updated IC

% initial conditions
dx = 0.01;
dt = 1E-4;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = exp(-0.5*(space).^2);

% This is for pre-setting the mesh
time = 3.250;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i-1,length(space)-1);

    % iterator
    for w=2:1:length(space)-1
        U(i,w) = U(i-1,w)-B*(+F(U(i-1,w),U(i-1,w+1))-
F(U(i-1,w-1),U(i-1,w)));
    end
end

% figure(1)
% plot(space,U(length(tt),:))
% titlename = ['Roe''s Method at ', num2str(time), 's'];
% title(titlename)
% xlabel('position x')
% ylabel('velocity U')
% legend('numerical')
% ylim([-3 4])
```

```
function F_ans = F(A,B)

    if A ~= B
        Ubar1 = (A+B)/2;
    elseif A == B
        Ubar1 = A;
    end

    e = max(0,(B-A)/2);

    if Ubar1 >= e
        Ubar2 = Ubar1;
    elseif Ubar1<e
        Ubar2 = e;
    end

    F_ans = (1/2)*(0.5*A^2+0.5*B^2)-(1/2)*abs(Ubar2)*(B-A);
end
```

*Published with MATLAB® R2018b*

```
clc
close all
clear all

% The following code will try to plot everything using the
% roe Method - lets see what happends. This time is with an
% updated IC. This is one without a limiter and is 2nd order

% initial conditions
dx = 0.01;
dt = 1E-4;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = exp(-0.5*(space).^2);

% This is for pre-setting the mesh
time = 3.25;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i-1,length(space)-1);
    U(i,length(space)-1) = U(i-1,length(space)-2);
    U(i,2) = U(i-1,length(space)-3);

    % iterator
    for w=3:1:length(space)-2
        U(i,w) = U(i-1,w)-B*(F(U(i-1,w-1),U(i-1,w),U(i-1,w+1),U(i-1,w
+2))-F(U(i-1,w-2),U(i-1,w-1),U(i-1,w),U(i-1,w+1)));
    end
end

% figure(1)
% plot(space,U(length(tt),:))
% titlename = ['Roe''s Method 2nd Order (No Flux Limiter) ',
 num2str(time), 's'];
% title(titlename)
% xlabel('position x')
% ylabel('velocity U')
% legend('numerical')
```

```
% ylim([-3 4])
% grid on

function F_ans = F(A,B,C,D)

    U_l = B+(1/2)*(B-A);
    U_r = C-(1/2)*(D-C);

    if U_l ~= U_r
        U_bar = (0.5*U_r^2-0.5*U_l^2)/(U_r-U_l);
    else
        U_bar = U_l;
    end
    %U_bar = (0.5*U_r^2-0.5*U_l^2)/(U_r-U_l);

    F_ans = (1/2)*(0.5*U_l^2+0.5*U_r^2-abs(U_bar)*(U_r-U_l));

end
```

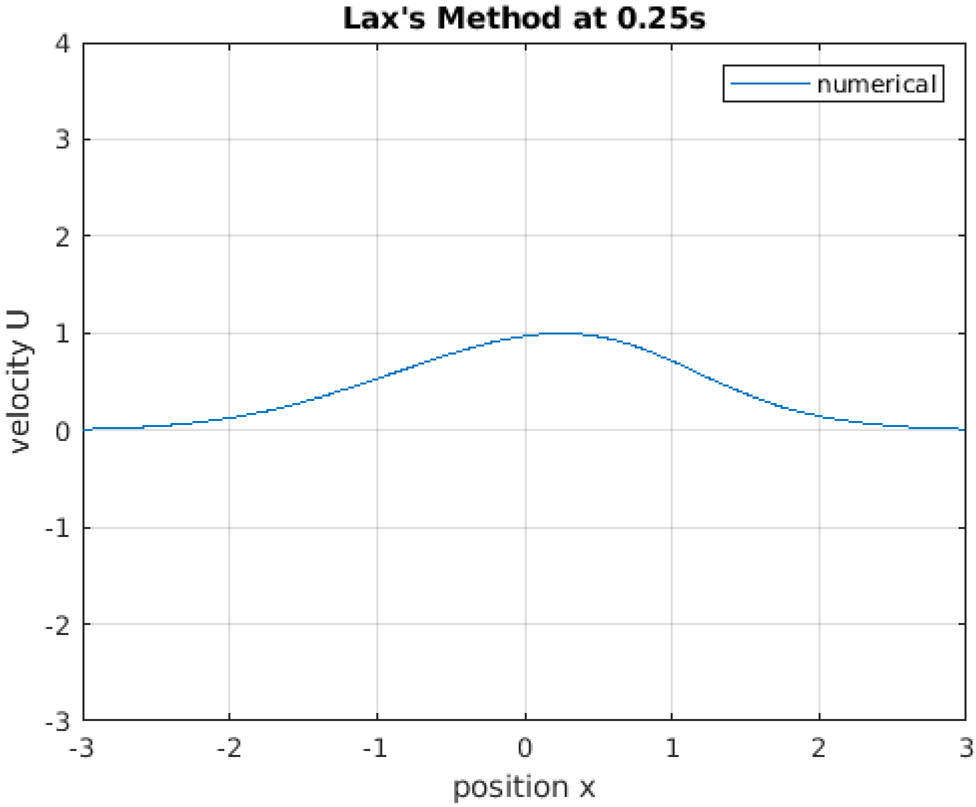
*Published with MATLAB® R2018b*

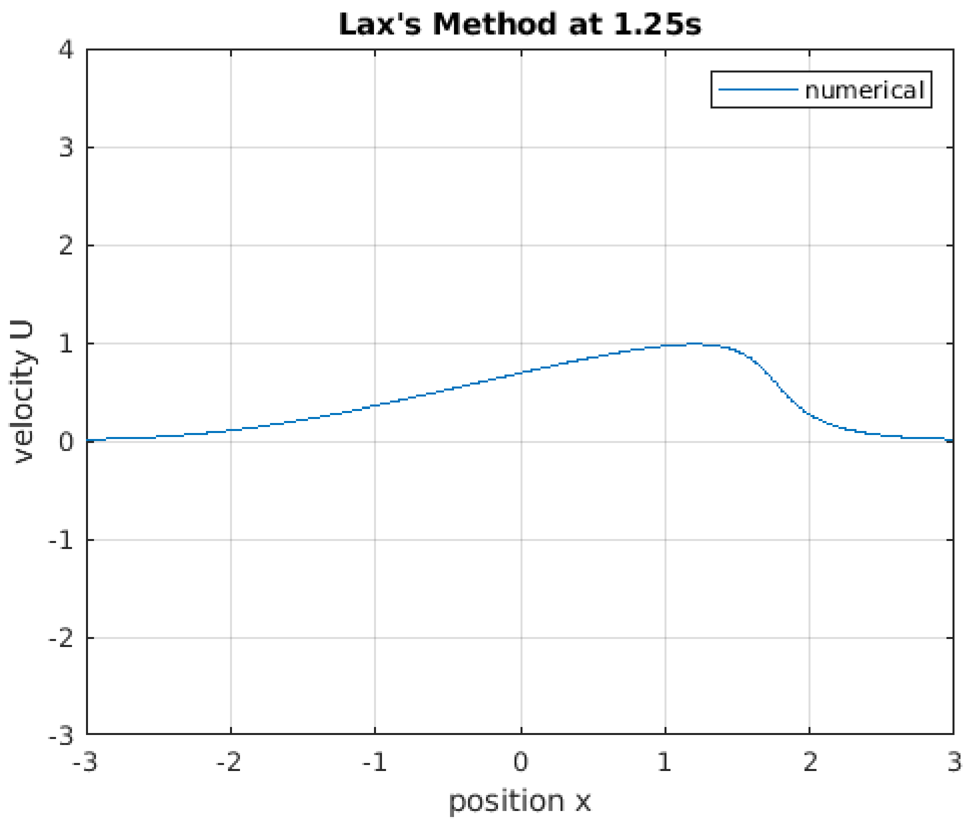
```matlab
clc
close all
clear all

% The following code will try to plot everything using the
% roe Method - lets see what happends. This one is with an updated IC.
% This one has a limiter and is 2nd order

% initial conditions
dx = 0.01;
dt = 1E-4;

% dt/dx
B = dt/dx;

% lets set up the x mesh
space = -3:dx:3;

% now lets set up the initial matrix
U_initial = exp(-0.5*(space).^2);

% This is for pre-setting the mesh
time = 3.25;
tt = 0:dt:time;
U = zeros(length(tt), length(space));

% just setting initial condition
U(1,:) = U_initial;

for i=2:1:length(tt)

    % setting initial conditions so this will work
    U(i,length(space)) = U(i-1,length(space)-1);
    U(i,1) = U(i-1,length(space)-1);
    U(i,length(space)-1) = U(i-1,length(space)-2);
    U(i,2) = U(i-1,length(space)-3);

    % iterator
    for w=3:1:length(space)-2
        U(i,w) = U(i-1,w)-B*(F(U(i-1,w-1),U(i-1,w),U(i-1,w+1),U(i-1,w
+2))-F(U(i-1,w-2),U(i-1,w-1),U(i-1,w),U(i-1,w+1)));
    end
end

% figure(1)
% plot(space,U(length(tt),:))
% titlename = ['Roe''s Method 2nd Order (Flux Limiter) at ',
 num2str(time), 's'];
% title(titlename)
% xlabel('position x')
% ylabel('velocity U')
% legend('numerical')
```

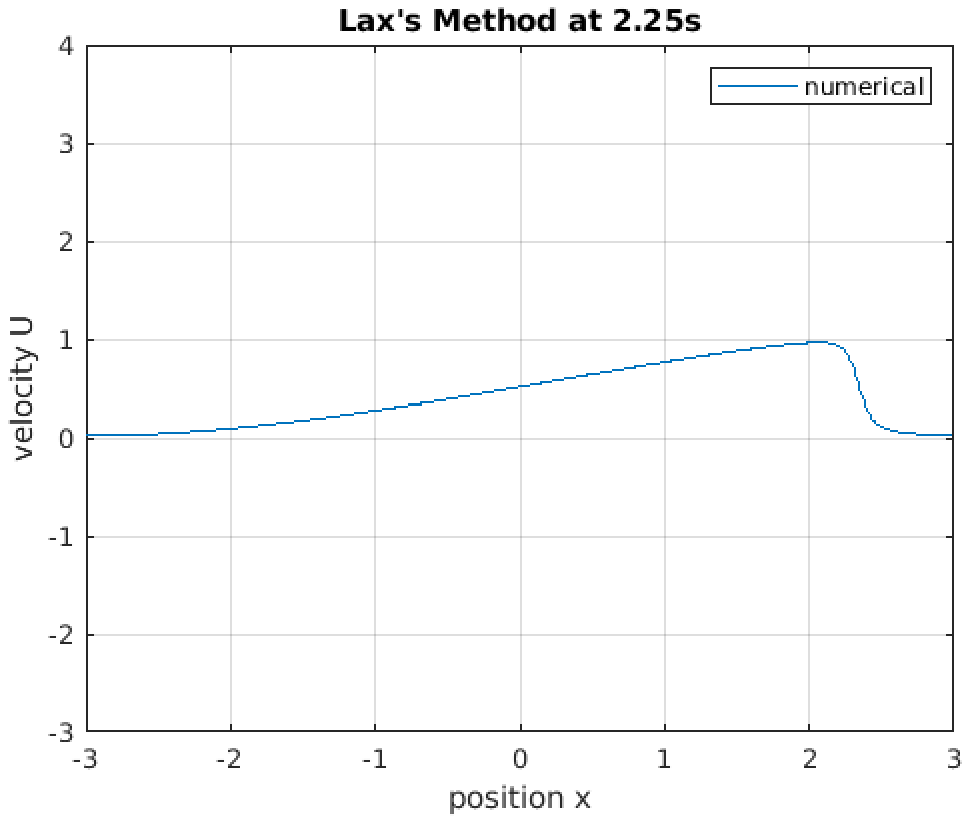
```
% ylim([-3 4])
% grid on

function F_ans = F(A,B,C,D)

    % flux limiter
    r_limit = (C-B)/(B-A);
    limiter = max(0,min(1,r_limit));

    U_l = B+(1/2)*limiter*(B-A);
    U_r = C-(1/2)*limiter*(D-C);

    if U_l ~= U_r
        U_bar = (0.5*U_r^2-0.5*U_l^2)/(U_r-U_l);
    else
        U_bar = U_l;
    end
    %U_bar = (0.5*U_r^2-0.5*U_l^2)/(U_r-U_l);

    F_ans = (1/2)*(0.5*U_l^2+0.5*U_r^2-abs(U_bar)*(U_r-U_l));

end
```

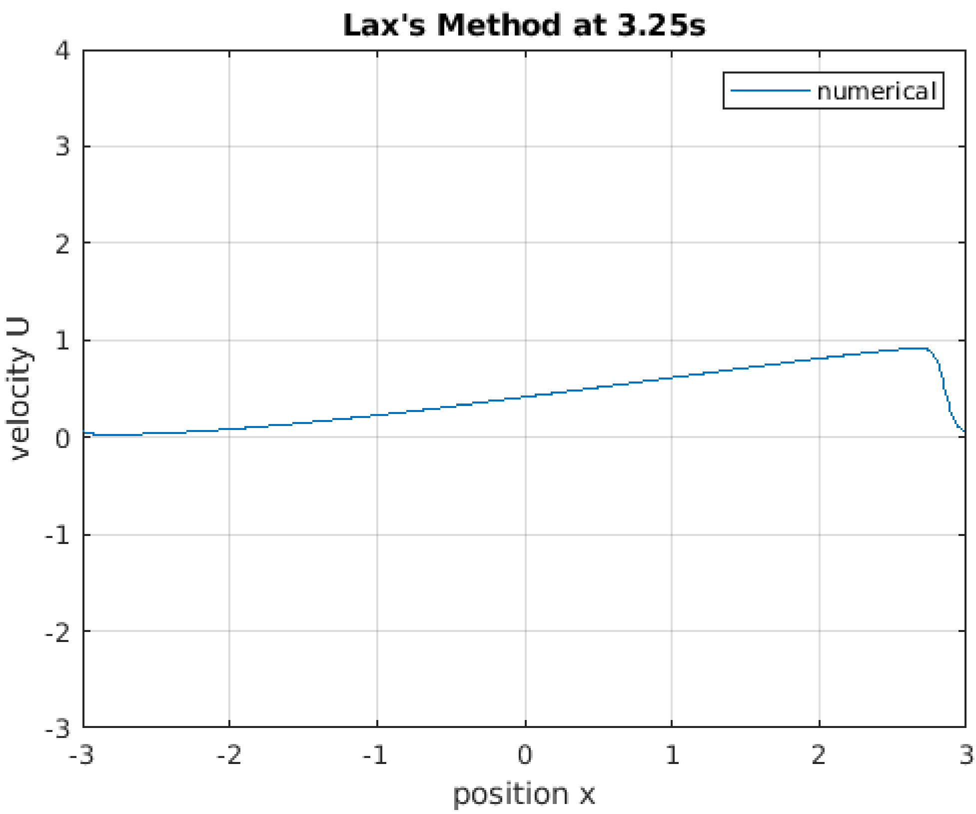
*Published with MATLAB® R2018b*

(a) Plots for Lax method -

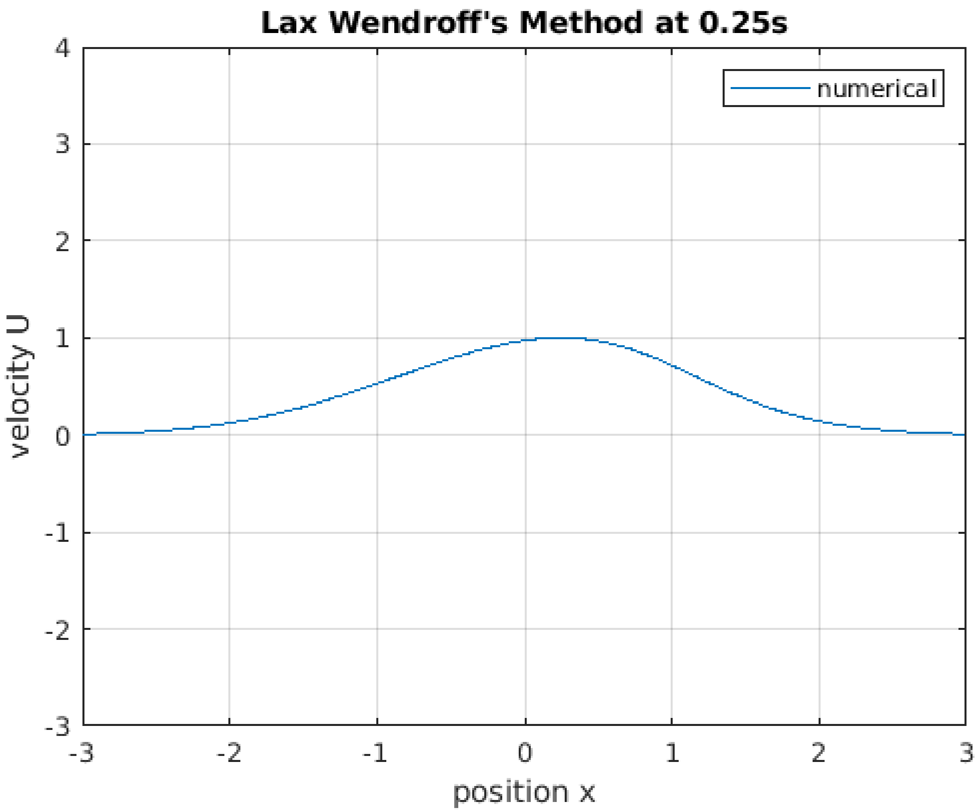


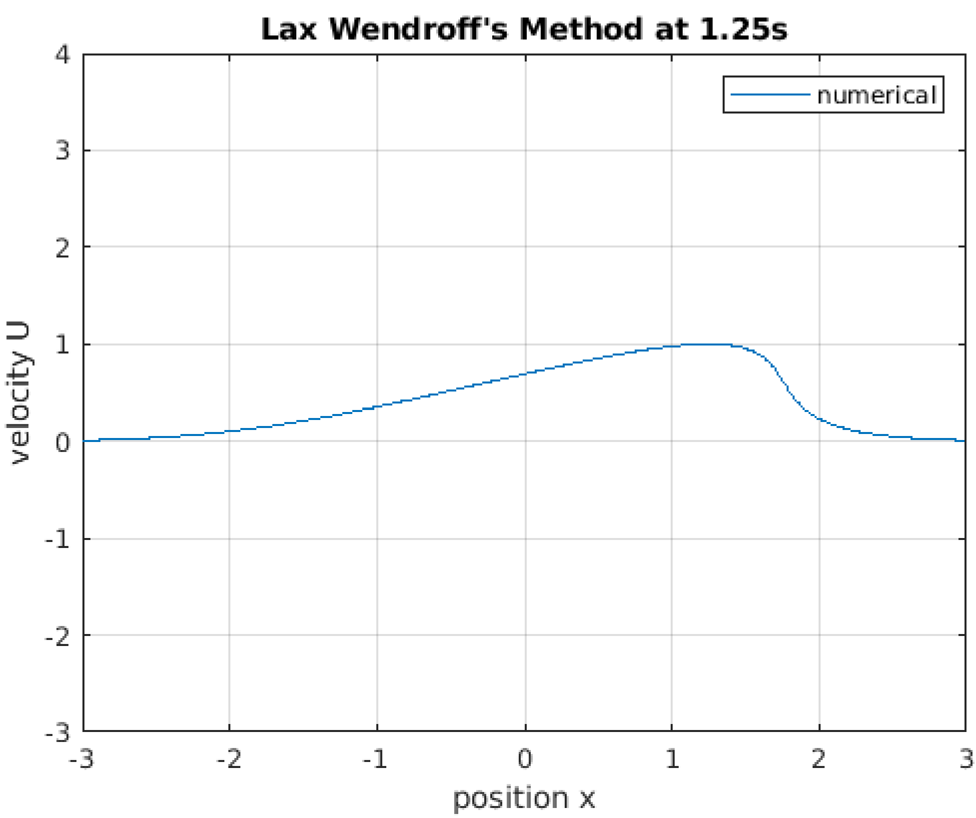



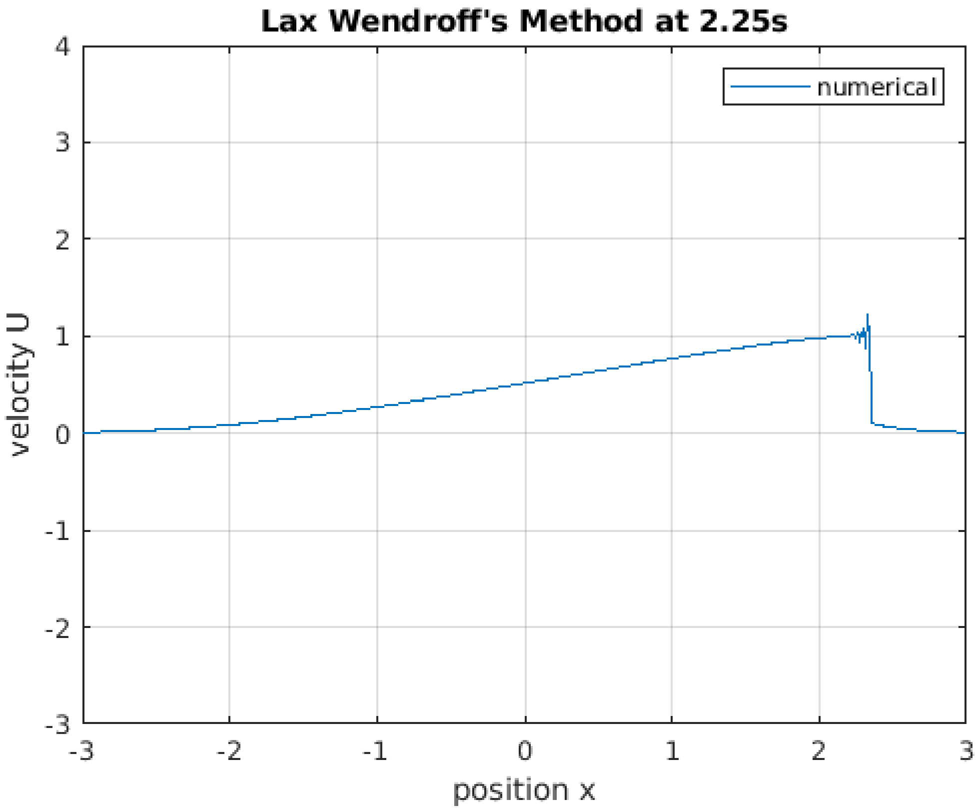

Lax's Method at 2.25s
numerical
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

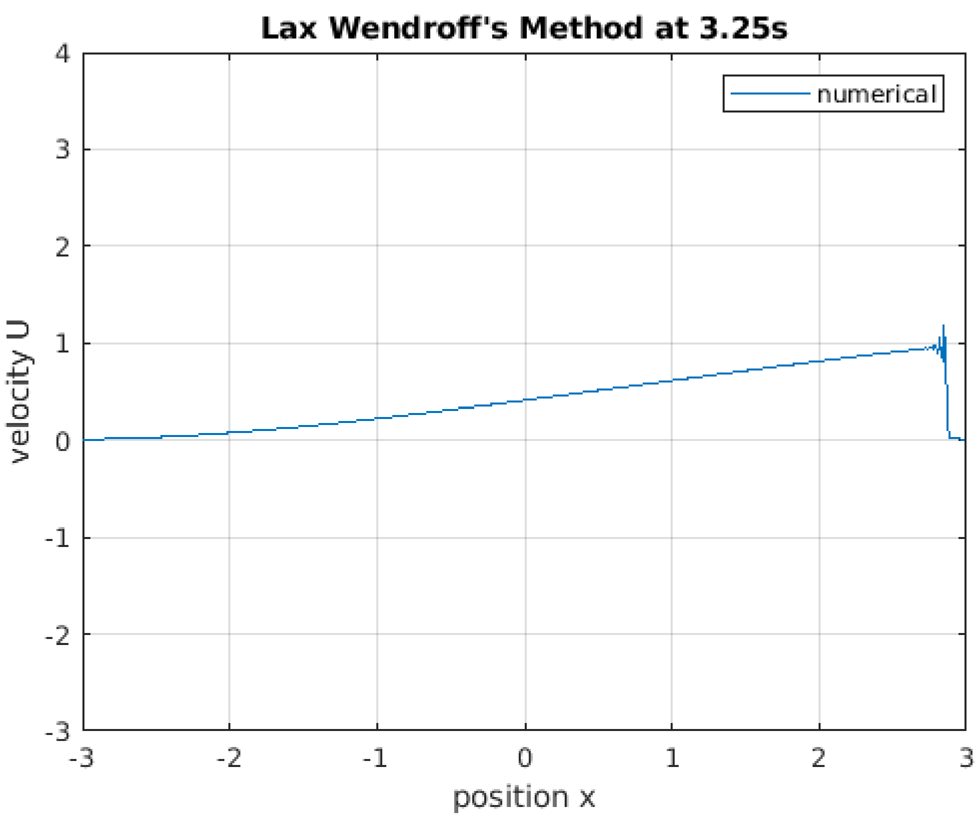

Lax's Method at 3.25s
numerical
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

(b) Plots for lax-wendroff -

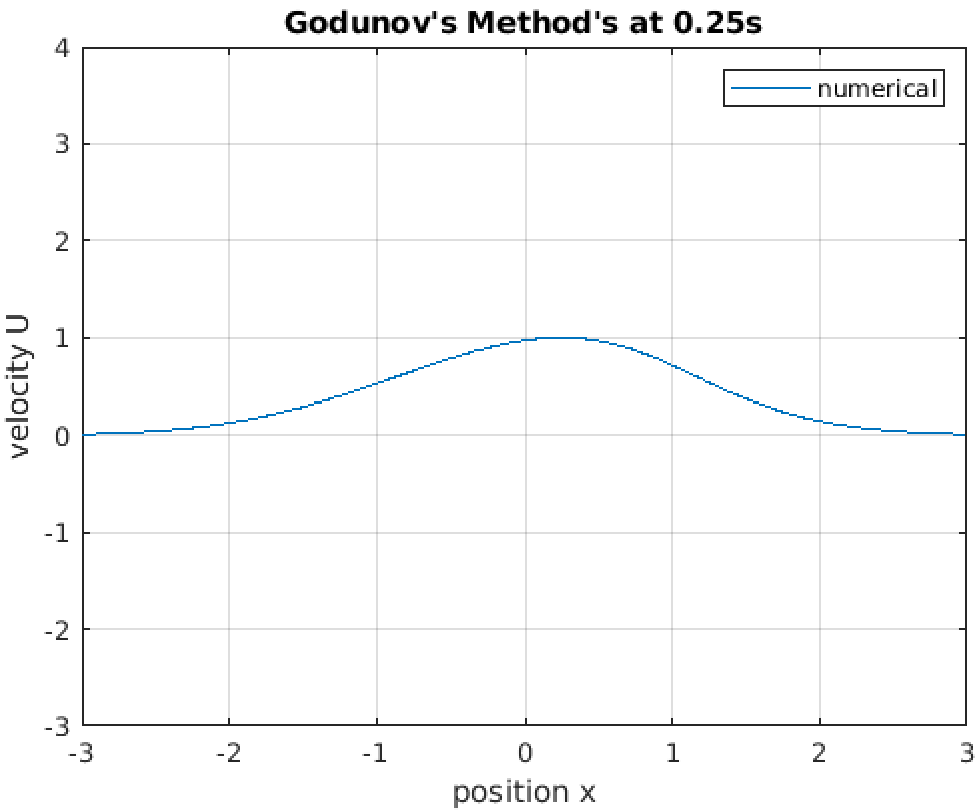


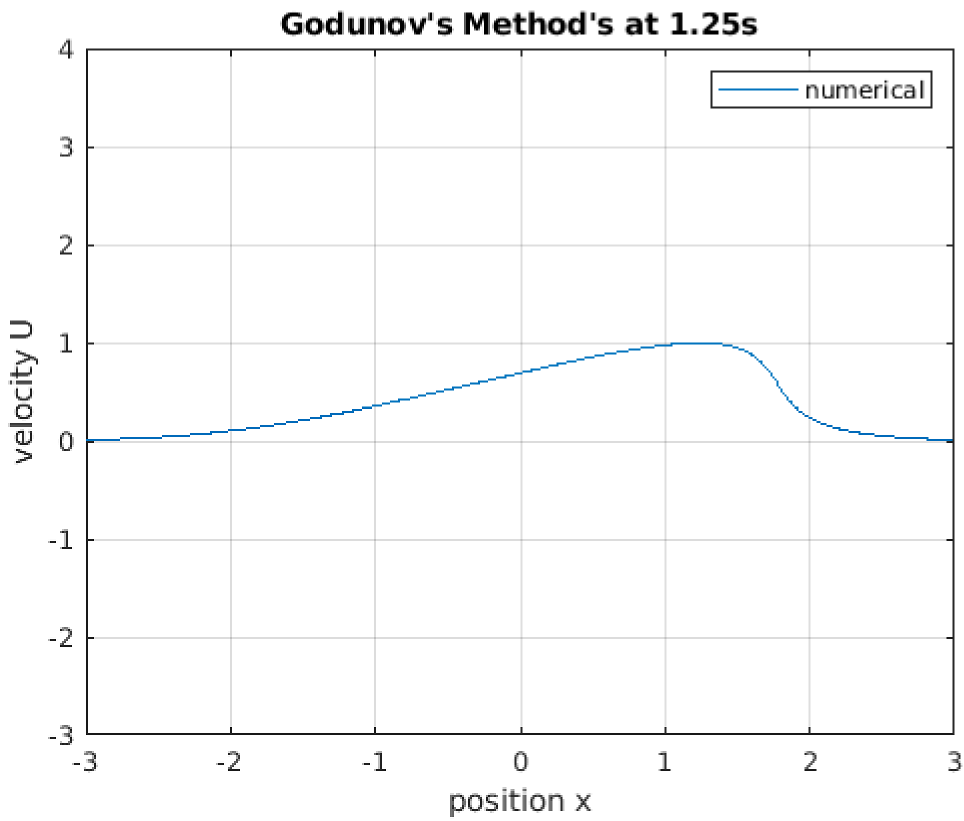



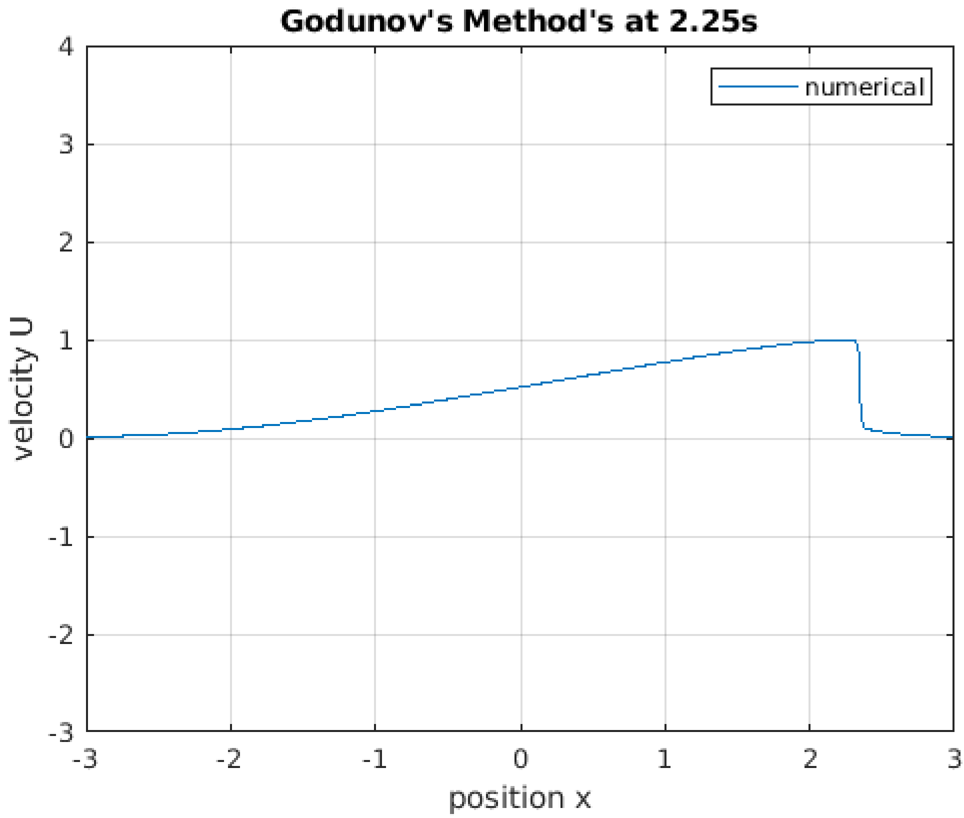

Lax Wendroff's Method at 2.25s
numerical
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

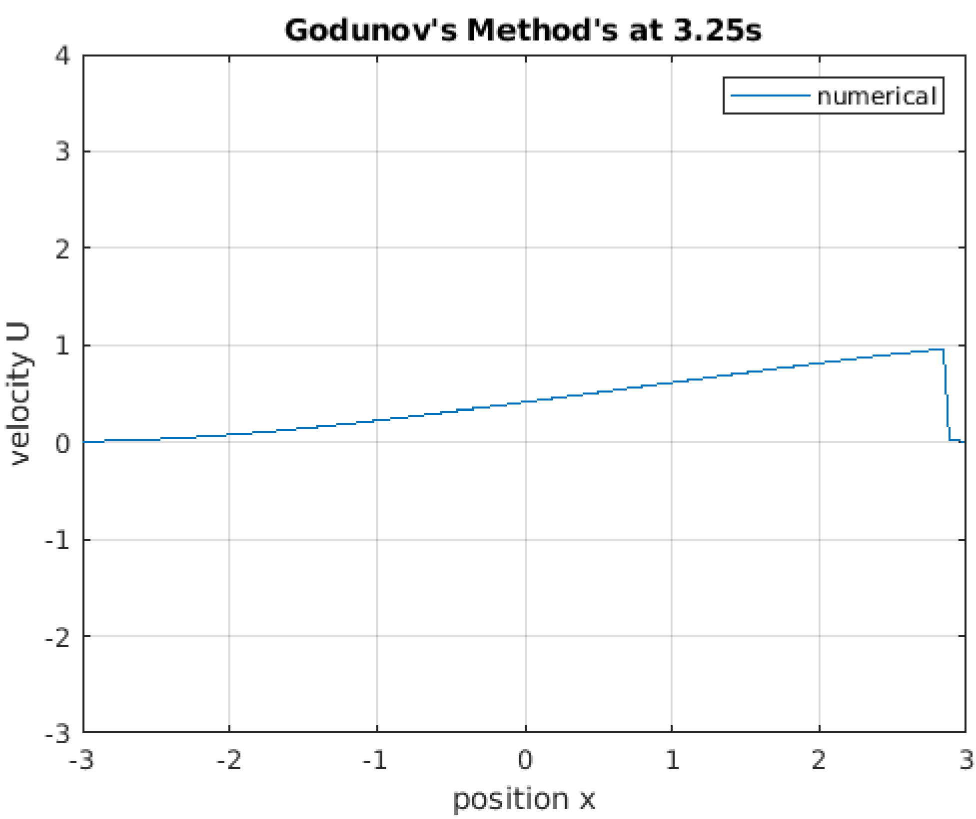

Lax Wendroff's Method at 3.25s
numerical
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

(c) Plots for Godunov -

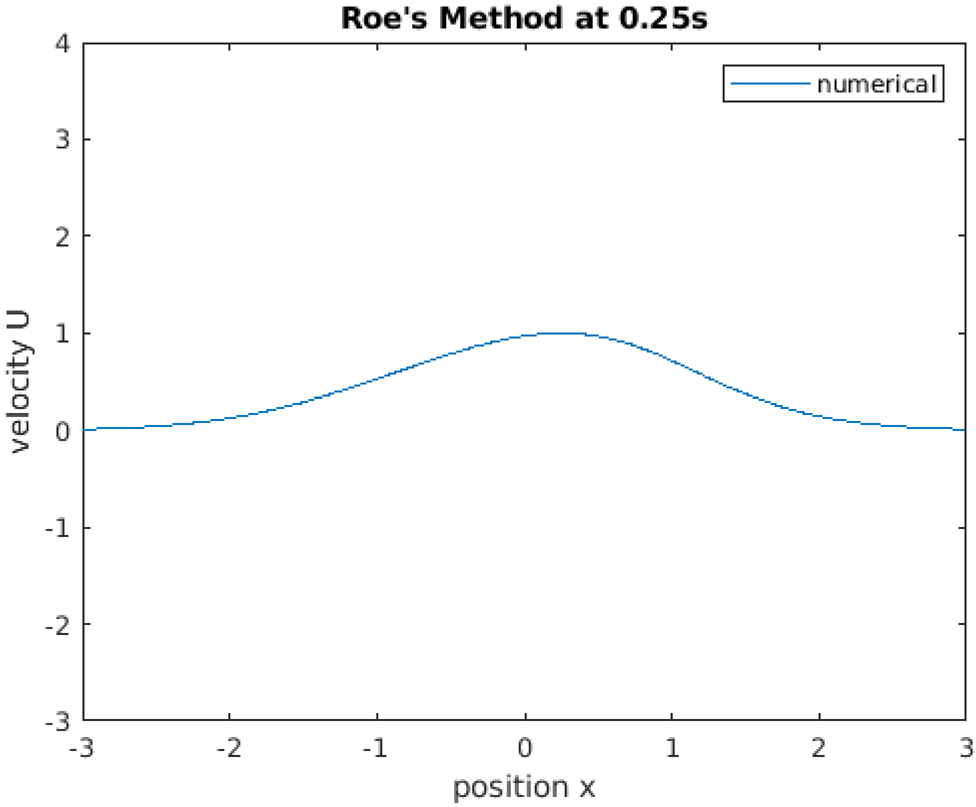


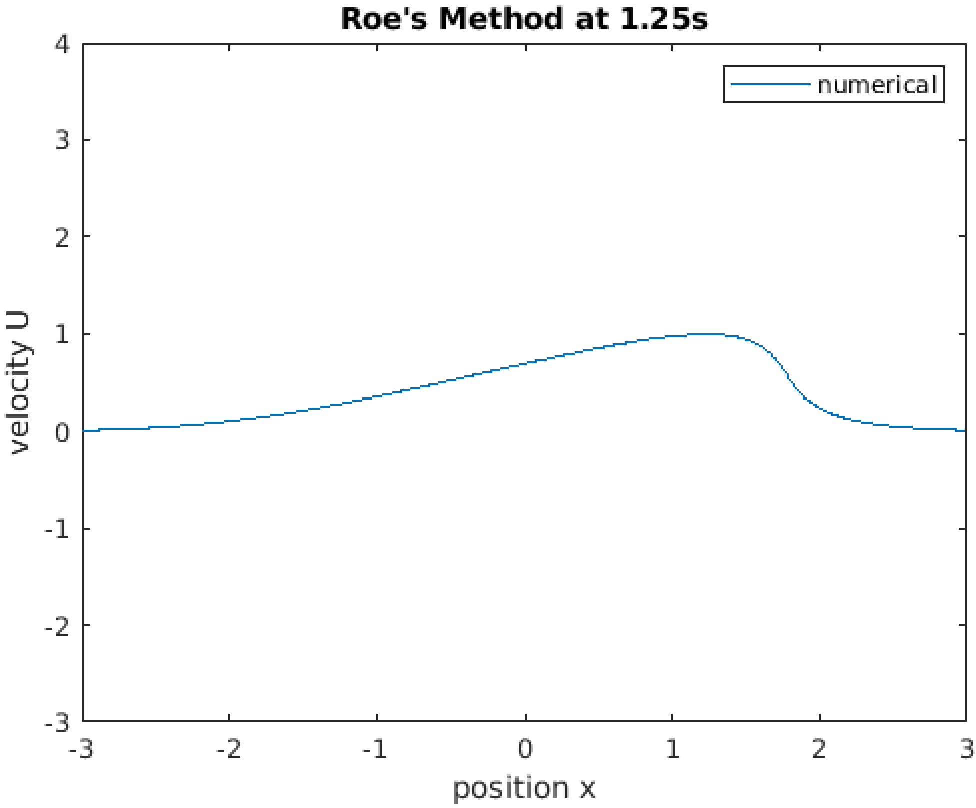



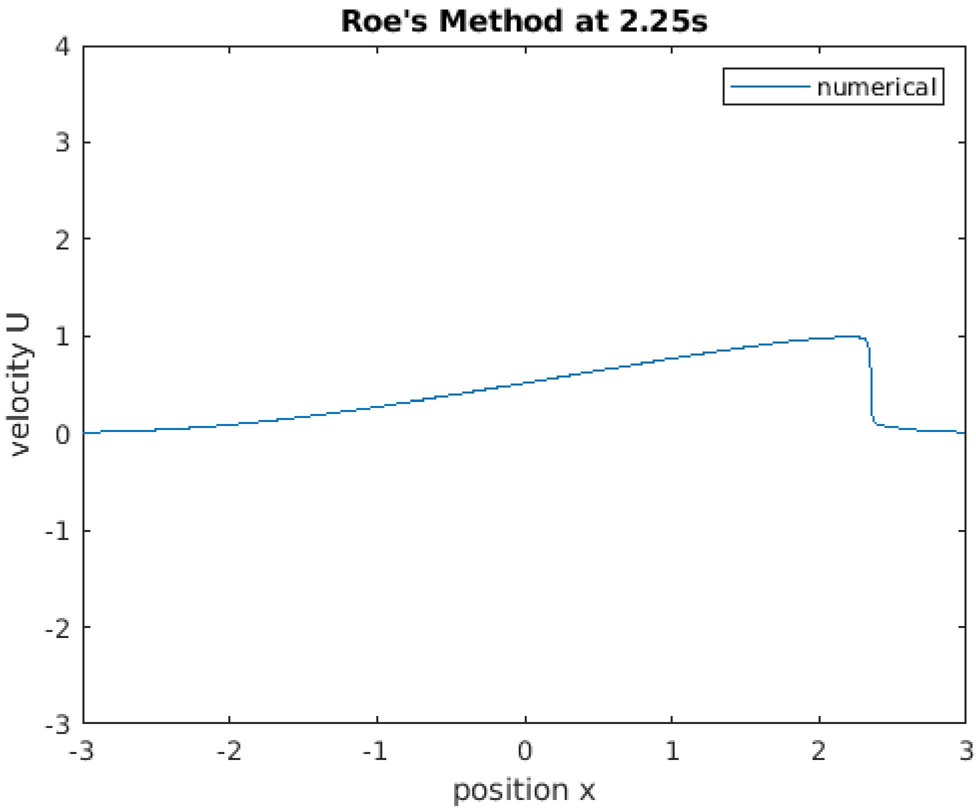

Godunov's Method's at 2.25s
numerical
velocity U
position x

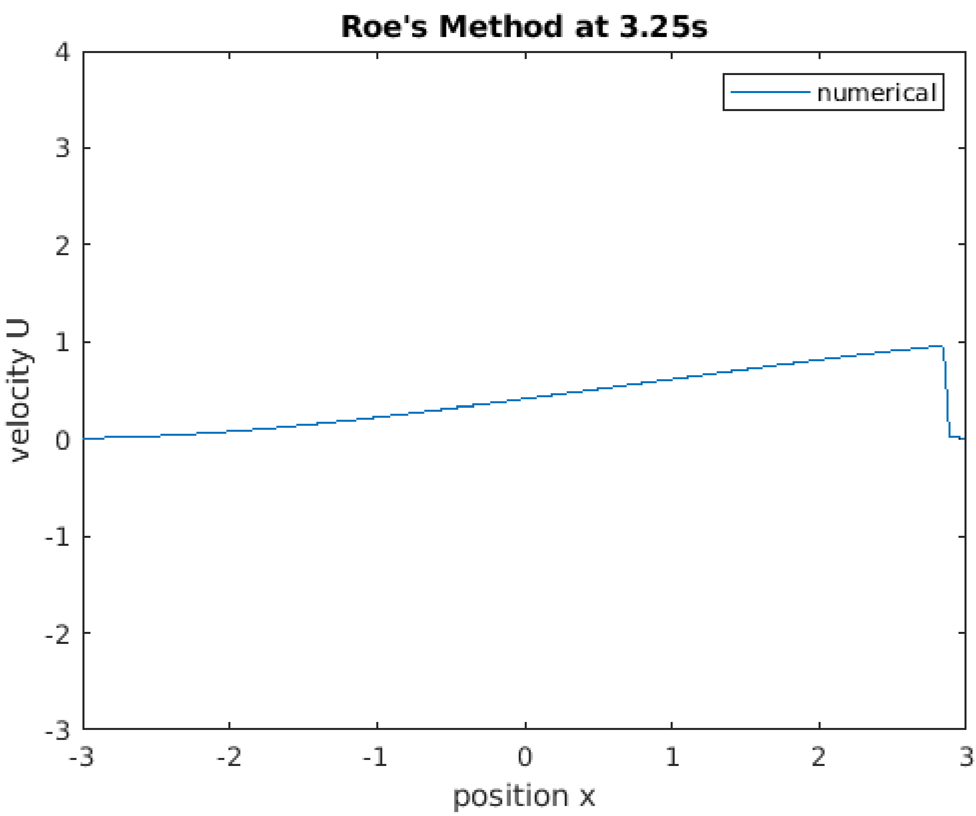

Godunov's Method's at 3.25s
numerical
velocity U
position x

(d) Plots for Roe -

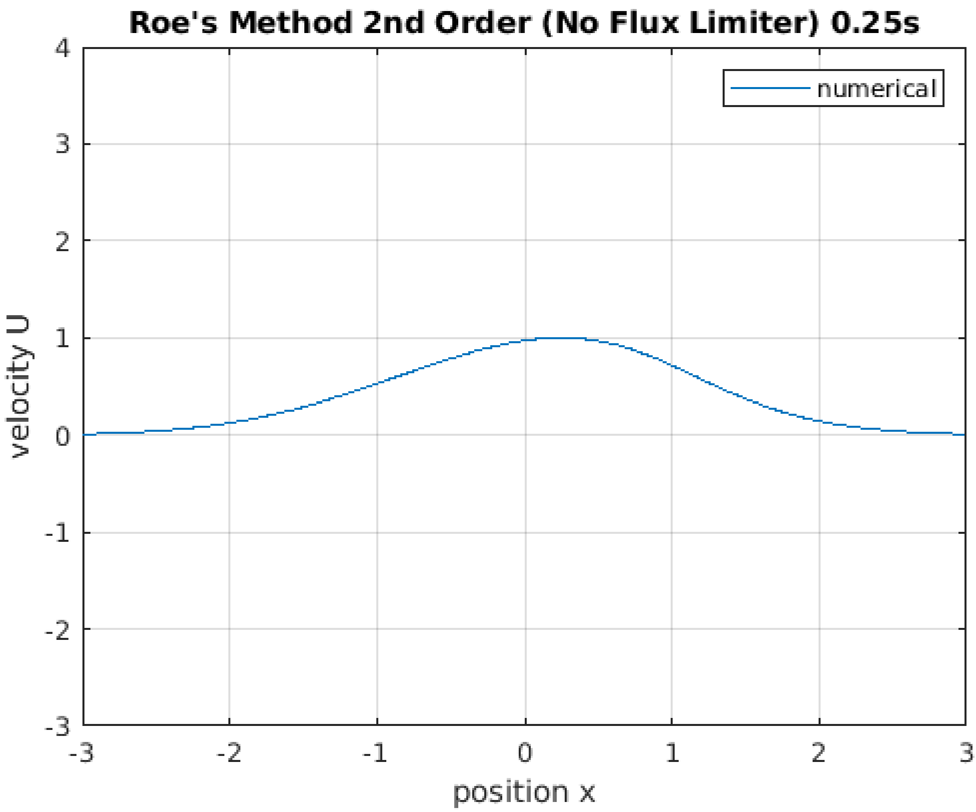


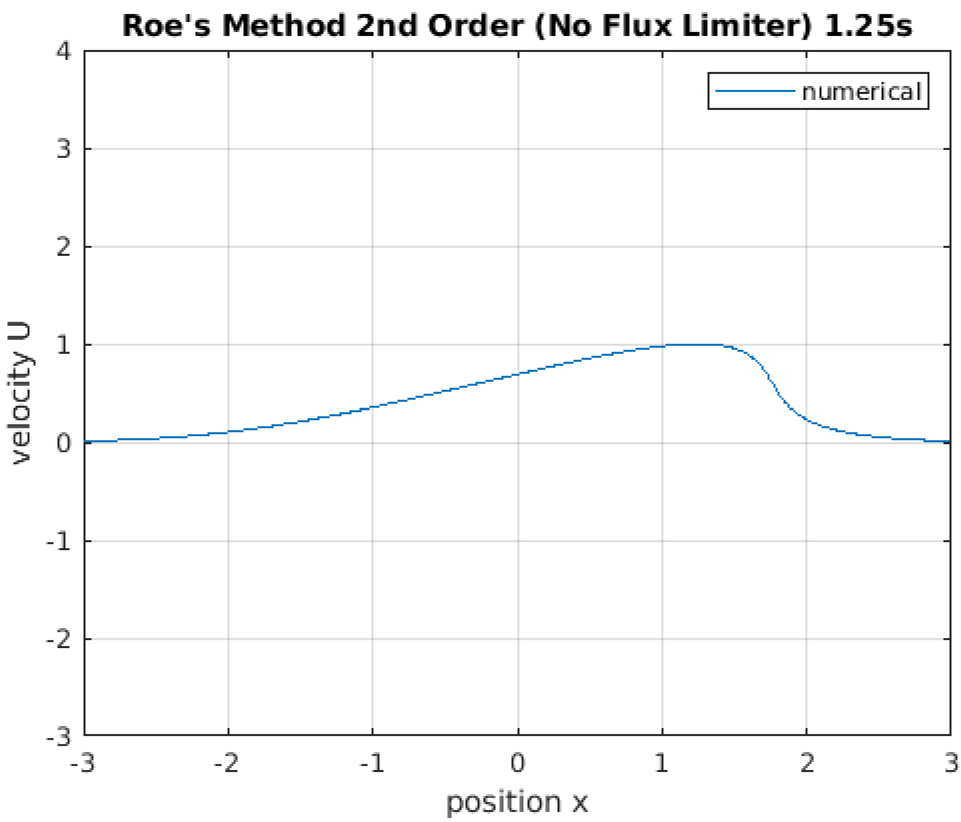



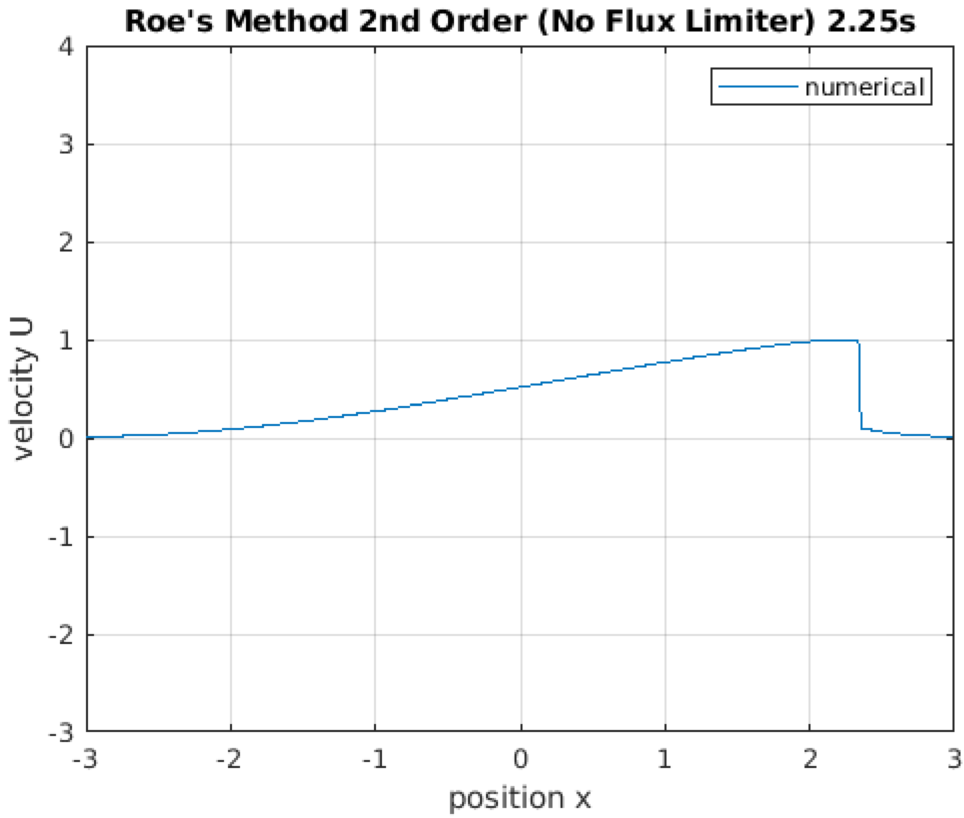

Roe's Method at 2.25s
numerical
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

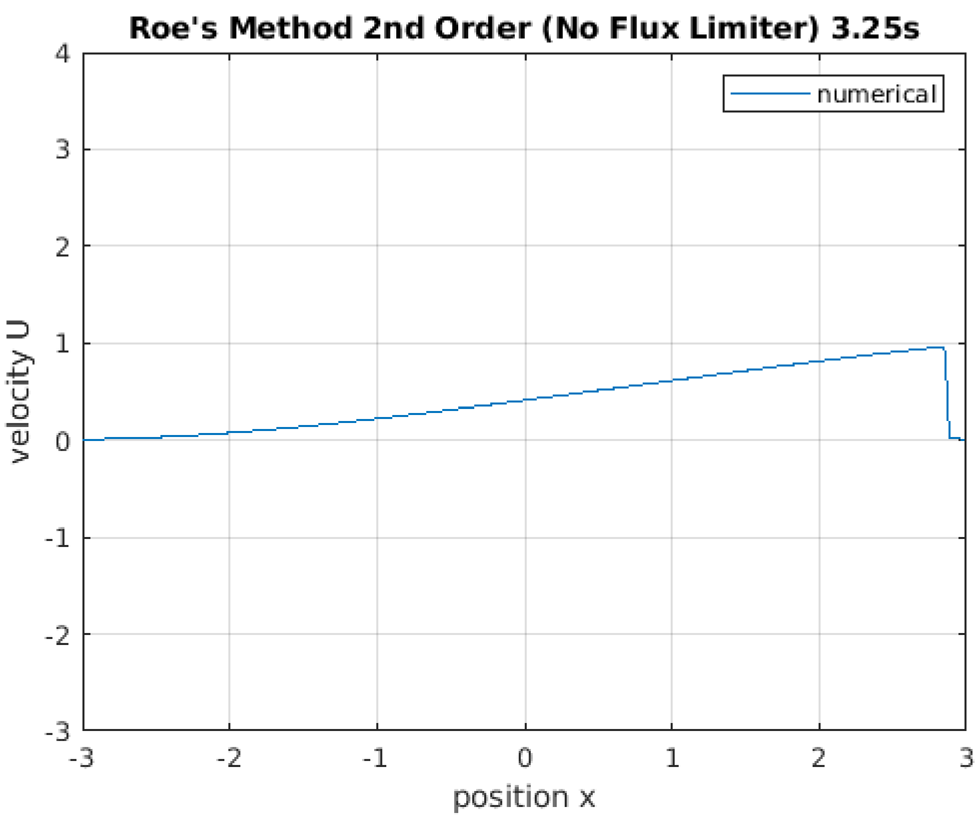

Roe's Method at 3.25s
numerical
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

(f) Roe 2nd order w/o flux limiter

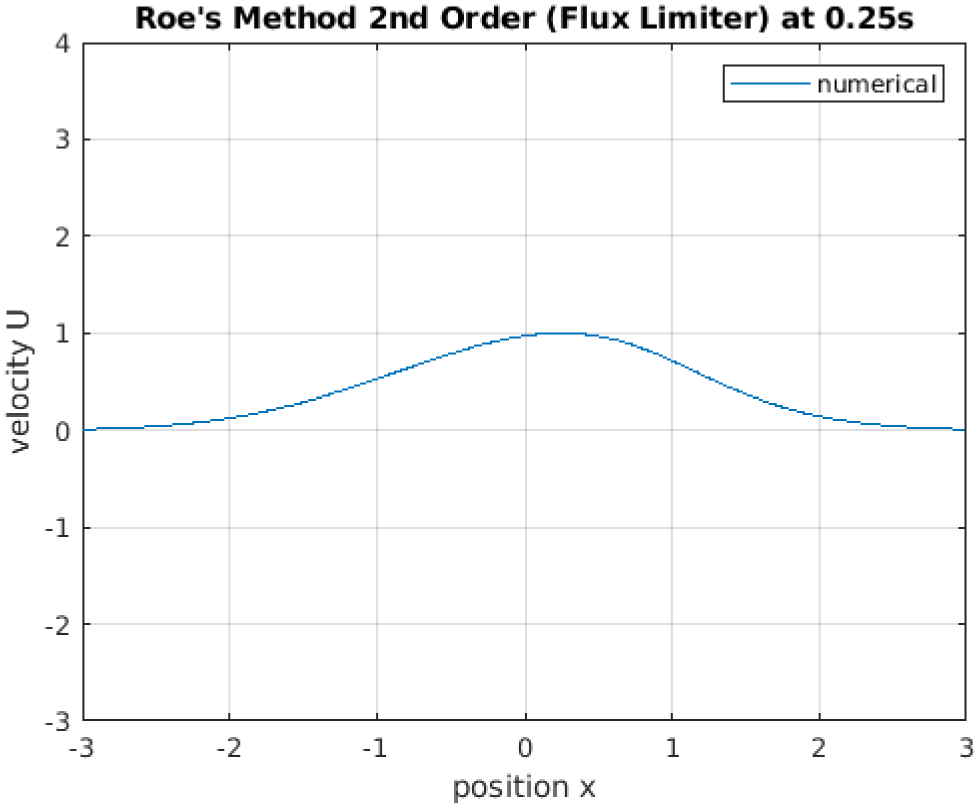


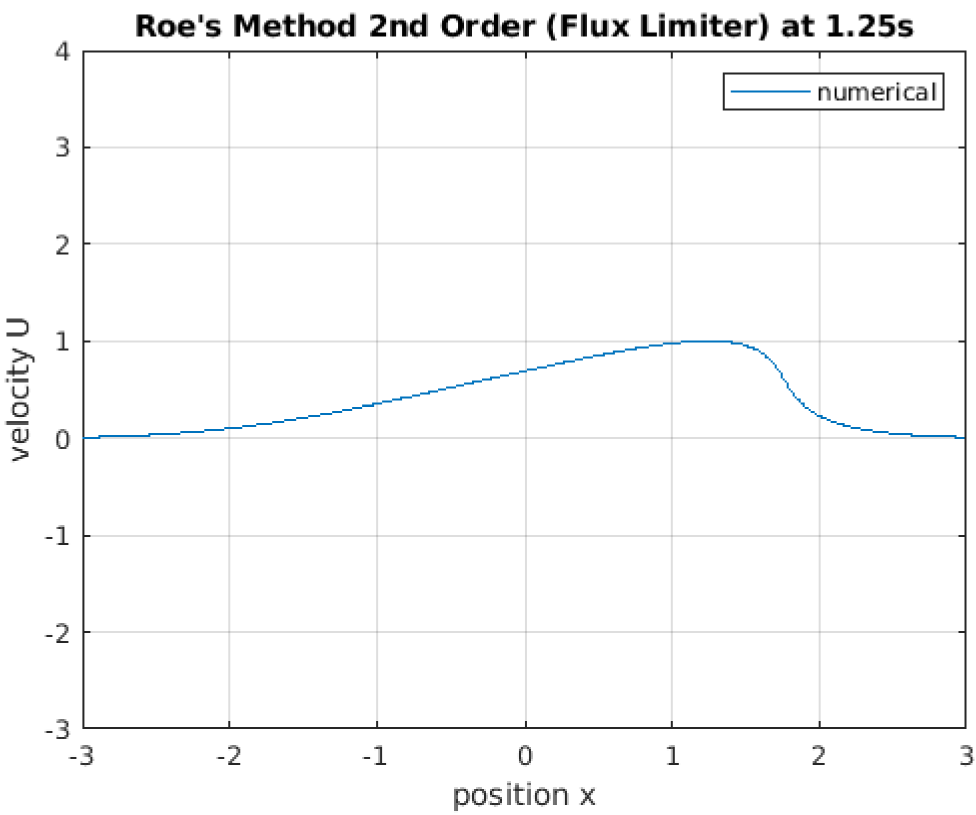



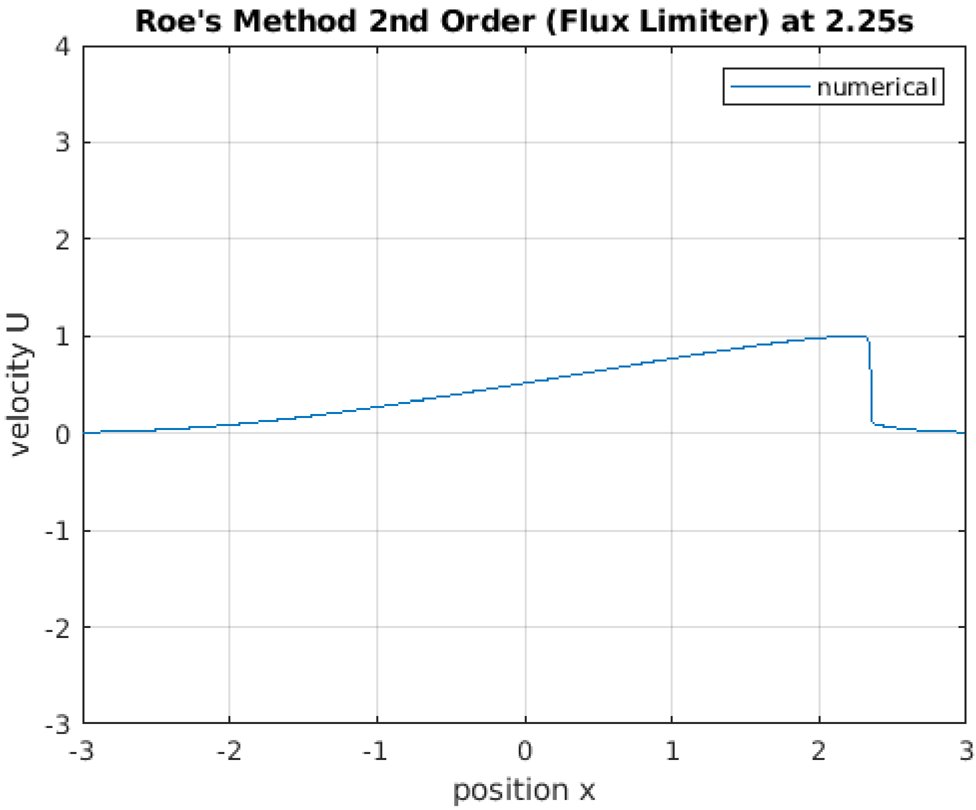

Roe's Method 2nd Order (No Flux Limiter) 2.25s
numerical
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

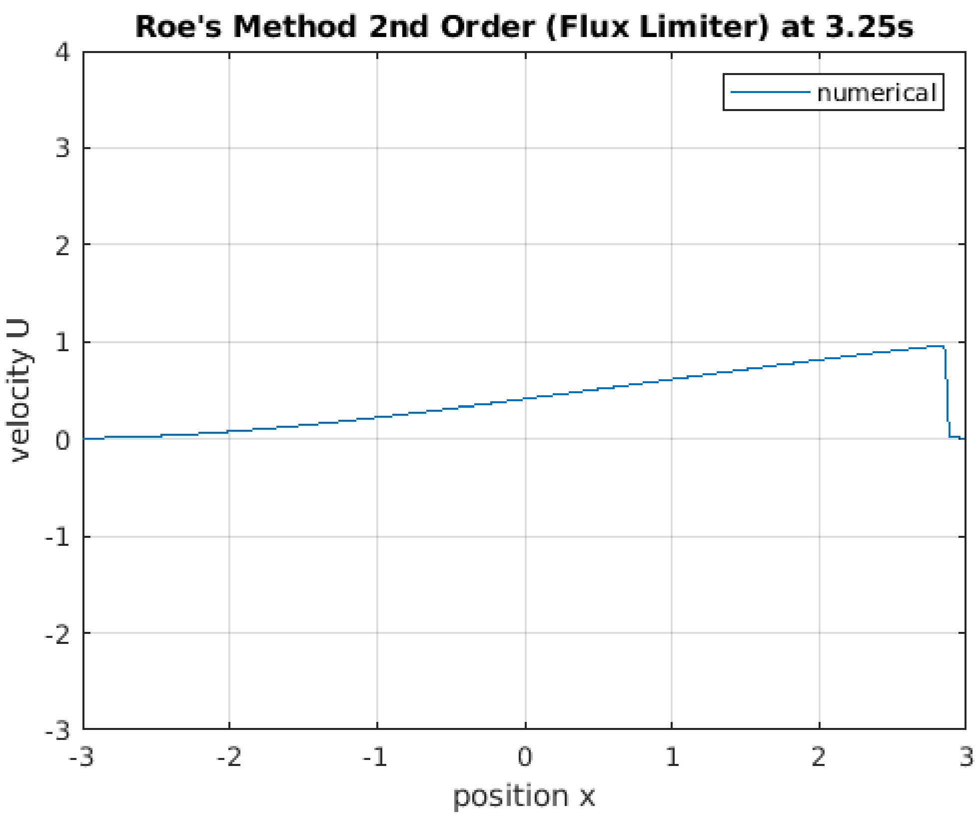

Roe's Method 2nd Order (No Flux Limiter) 3.25s
numerical
velocity U
position x
4
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

(f) Roe $2^{nd}$ order w/ flux limiter




Roe's Method 2nd Order (Flux Limiter) at 2.25s
numerical
velocity U
position x


Roe's Method 2nd Order (Flux Limiter) at 3.25s
numerical
velocity U
position x

(b) Does this initial condition lead to a discontinuity: yes

(c) If so, what value of time: at least from my plots the discontinuity starts around t=2.25(s)