1. $U_i^* = U_i^N - \frac{k}{h}\left(f(U_{i+1}^N) - f(U_i^N)\right)$

$$U_i^{N+1} = \frac{\left[U_i^N - \frac{k}{h}\left(f(U_{i+1}^N) - f(U_i^N)\right)\right] + U_i^N}{2} - \frac{k}{2h}\left[f(U_i^*) - f(U_{i-1}^*)\right]$$

$$U_i^{N+1} = U_i^N - \frac{k}{2h}\left[f(U_{i+1}^N) - f(U_i^N) + f(U_i^*) - f(U_{i-1}^*)\right]$$

$$F_i^L = \left[f(U_{i-1}^*) + f(U_i^N)\right]$$

$$F_i^R = \left[f(U_{i+1}^N) + f(U_i^*)\right]$$

$$\boxed{U_i^{N+1} = U_i^N - \frac{k}{2h}\left[F_i^R - F_i^L\right]}$$

This method is conservative because it finds the solution at the next time step, (U^{N+1}), using the solution from the previous step, (U^N), combined with the difference of the numerical flux terms around the point $[F_i^R - F_i^L]$.

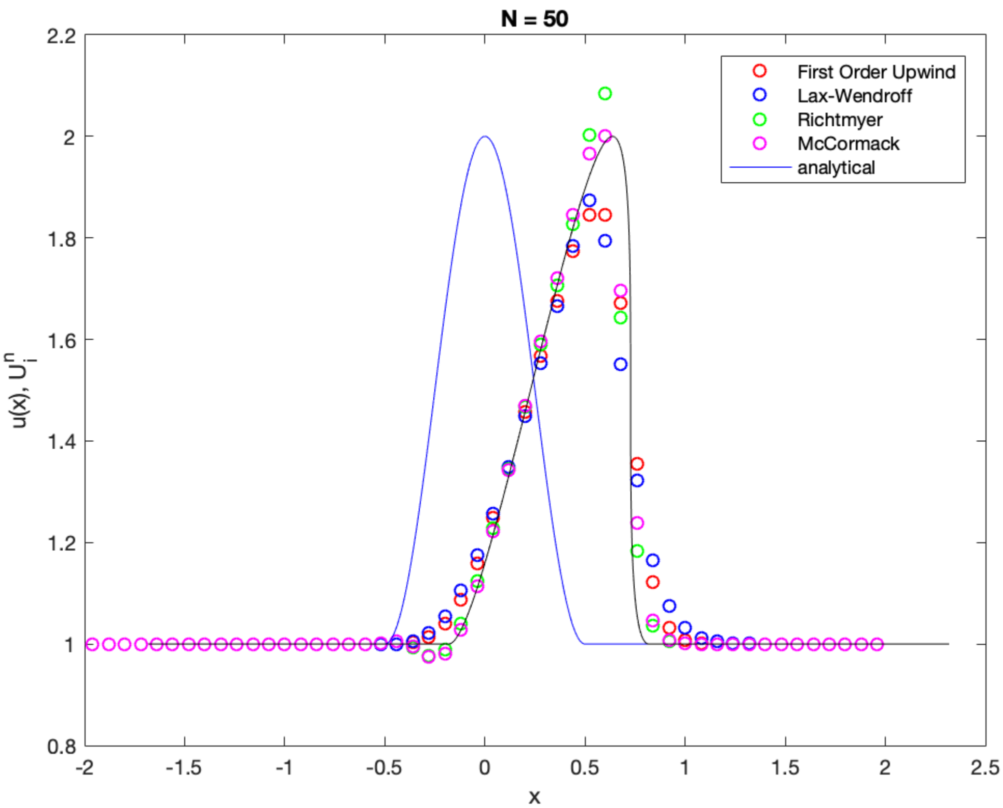
```
function F=fluxevaluate(fh,kh,U,uL,uR,method)


% function F=fluxevaluate(fh,kh,U,uL,uR,method)
%
% fh: function handle to the flux function
% kh: k/h factor
% U:  solution at time t = t_n
% uL: left boundary condition u(-L)=uL
% uR: right boundary condotion u(+L)=uR
% method: one of 'first-order-upwind', 'law-wendroff',
%                 'richtmyer', 'maccormack'

N=length(U);
F=zeros(N+1,1);

%
%
%    F(1)  F(2)   F(3)       F(i-1)  F(i)   F(i+1)            F(N)   F(N+1)
%    |------|------|--- //  ---|-------|------|------|--- // ---|-----|
%       1      2                  i-1      i     i+1               N
%   -L                                                                L
%

switch lower(method)
case 'first-order-upwind'

  % F(i) = F^L_i, the left numerical flux
  for i=2:N
     if ( U(i)>0 ) F(i)=feval(fh,U(i-1)); end;
     if ( U(i)<0 ) F(i)=feval(fh,U(i)); end;
  end

  % BCs

  % left
  if ( U(1)>0 ) F(1)=feval(fh,uL); end;
  if ( U(1)<0 ) F(1)=feval(fh,U(1)); end;

  % right
  if ( U(N)>0 ) F(N+1)=feval(fh,U(N)); end;
  if ( U(N)<0 ) F(N+1)=feval(fh,uR); end;


case 'lax-wendroff'
     U = [uL; U; uR];             % set ghost cells
     N_ = N+2;

     F(1) = 1;
    %F(N_) = 1;

    for i = 2:N_-1
        a = (U(i) + U(i+1)) / 2;
        b = (U(i-1) + U(i))/2;
        F(i) = 0.5*feval(fh,U(i)) + 0.5*feval(fh,U(i-1)) + 0.5*kh*a*feval(fh,U(i)) - 0.5*kh*b*feval(fh,U(i-1));
    end

    if ( U(1)>0 ) F(1)=feval(fh,uL); end;
    if ( U(1)<0 ) F(1)=feval(fh,U(1)); end;

    % right
    if ( U(N)>0 ) F(N+1)=feval(fh,U(N)); end;
    if ( U(N)<0 ) F(N+1)=feval(fh,uR); end;


case 'richtmyer'
```

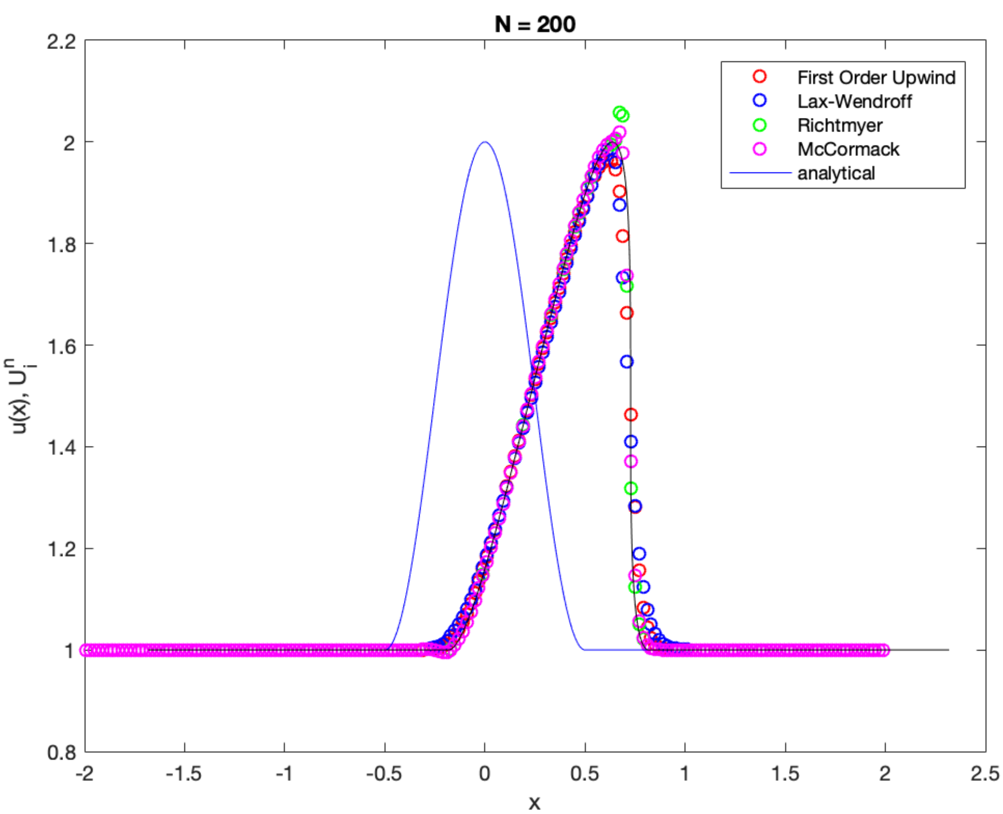
```
    U = [uL; U; uR];
% BCs

% left
    if ( U(1)>0 ) F(1)=feval(fh,uL); end;
    if ( U(1)<0 ) F(1)=feval(fh,U(1)); end;

% right
    if ( U(N)>0 ) F(N+1)=feval(fh,U(N)); end;
    if ( U(N)<0 ) F(N+1)=feval(fh,uR); end;

    for i = 2 : N+1
        a = (U(i+1) + U(i))/2; % U (i+0.5)
        b = (U(i-1) + U(i))/2; % U (i-0.5)


        if U(i) > 0
            F(i) = a - kh * (feval(fh,U(i+1))-feval(fh,U(i))) / 2;
            F(i) = feval(fh,F(i));
        end
        if U(i) < 0
            F(i) = b - kh * (feval(fh,U(i))-feval(fh,U(i-1))) / 2;
            F(i) = feval(fh,F(i));
        end
    end


case 'maccormack'
    U = [uL; U; uR];
% BCs

% left
    if ( U(1)>0 ) F(1)=feval(fh,uL); end;
    if ( U(1)<0 ) F(1)=feval(fh,U(1)); end;

% right
    if ( U(N)>0 ) F(N+1)=feval(fh,U(N)); end;
    if ( U(N)<0 ) F(N+1)=feval(fh,uR); end;

    for i = 2 : N + 1
        if U(i) > 0 % =
            temp = U(i) - kh*(feval(fh,U(i+1))-feval(fh,U(i)));
            F(i) = (feval(fh,U(i+1)) + feval(fh,temp))/2;
        end
        if U(i) < 0 %
            temp = U(i-1) - kh*(feval(fh,U(i)) - feval(fh,U(i-1)));
            F(i) = (feval(fh,U(i)) + feval(fh,temp))/2;
        end
    end

otherwise
  error('method is unknown');

end
```

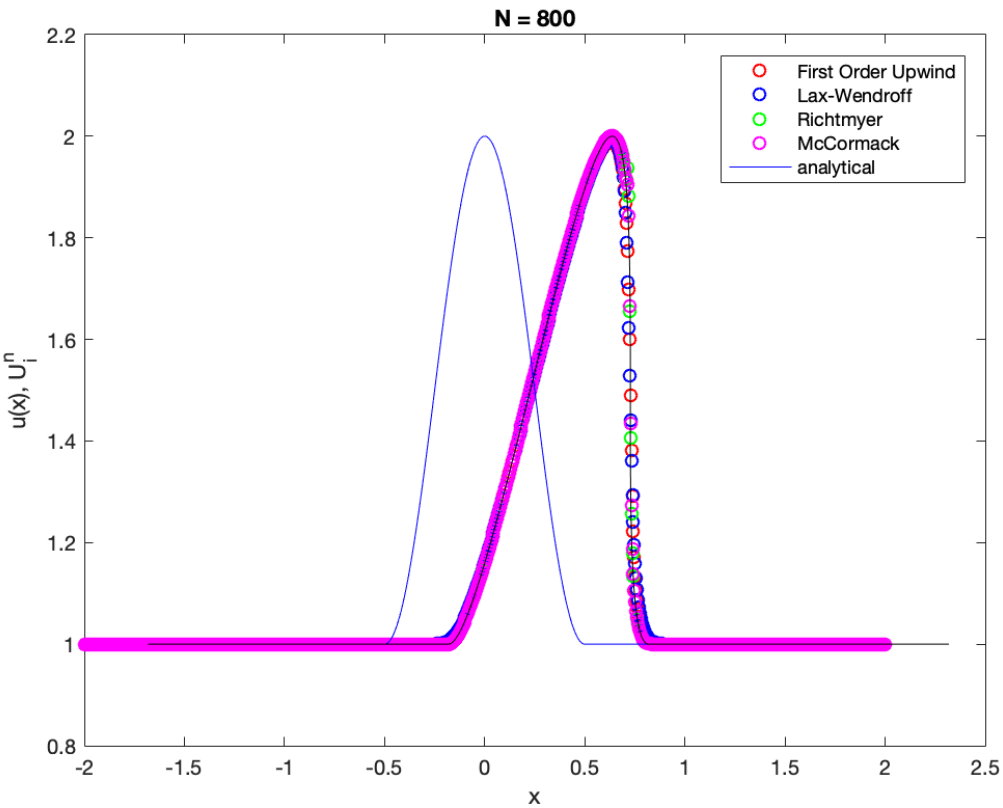
```
Not enough input arguments.
Error in fluxevaluate (line 14)
N=length(U);
```

Published with MATLAB® R2017a

```
clear all; close all; clc
% ready to submit

%method='first-order-upwind';
%method='lax-wendroff';
%method='richtmyer';
%method='maccormack';
%uh=@step; uL=2; uR=1; L=2; N=50; T=2/3; name='step';
uh=@hump; uL=1; uR=1; L=2; N=50; T=1/pi-1e-5; name='hump';


method = {'first-order-upwind' 'lax-wendroff' 'richtmyer' 'maccormack'};
line = {'ro','bo','go','mo'};
N = [50 200 800];

fh=@burgers; % exact flux function f=f(u)



sigma=0.75;

for i = 1:3       % loop through n
    figure
    for j = 1:4   % loop through methods
        [xm,U]=advanceconservative(uh,fh,uL,uR,L,sigma,N(i),T,method{j});
        x_=linspace(-L,L,10000);
        [xi,ui]=burgersanalytical(x_,uh,T);
        plot(xm,U,line{j});
        hold on
    end

    %axis([-0.5,L,0,2.5]);
    plot(xi, feval(uh,xi),'b-')
    hold on
    plot(xi, ui, 'k-')
    hold off

    title(sprintf('N = %d',N(i)));
    xlabel('x'); ylabel('u(x), U^n_i');
    legend('First Order Upwind','Lax-Wendroff','Richtmyer','McCormack','analytical');
end
```


N = 50
First Order Upwind
Lax-Wendroff
Richtmyer
McCormack
analytical
u(x), U_i^n
x
2.2
2
1.8
1.6
1.4
1.2
1
0.8
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5

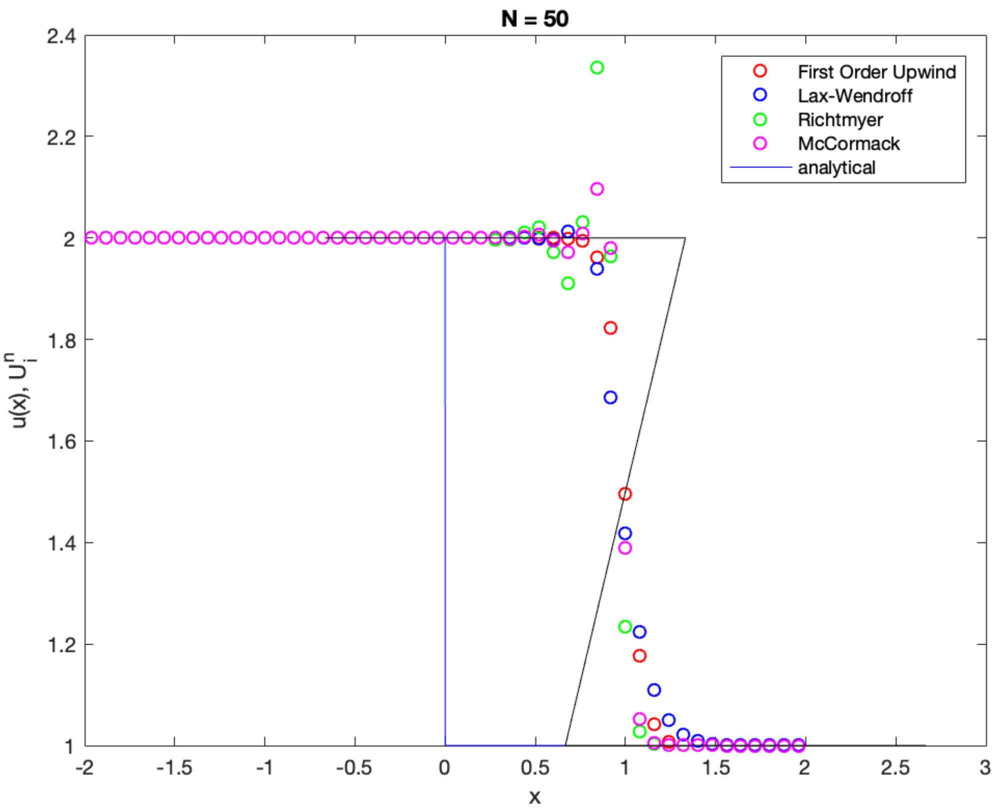

N = 200
First Order Upwind
Lax-Wendroff
Richtmyer
McCormack
analytical
u(x), U_i^n
x
2.2
2
1.8
1.6
1.4
1.2
1
0.8
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5

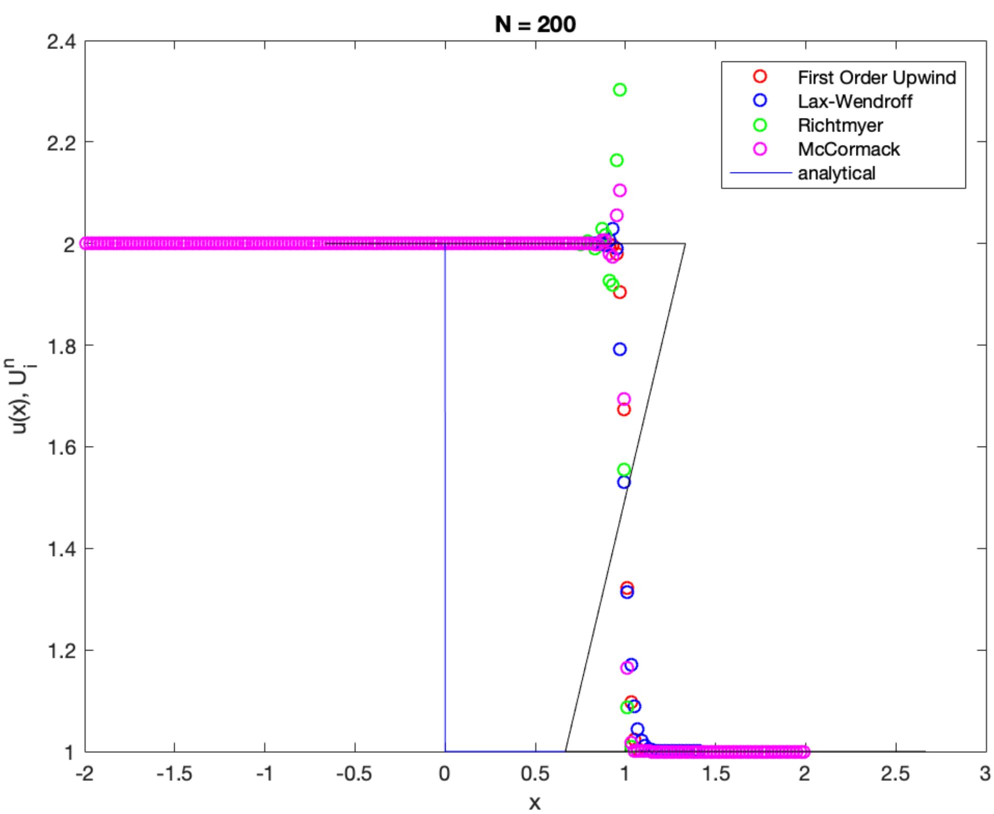


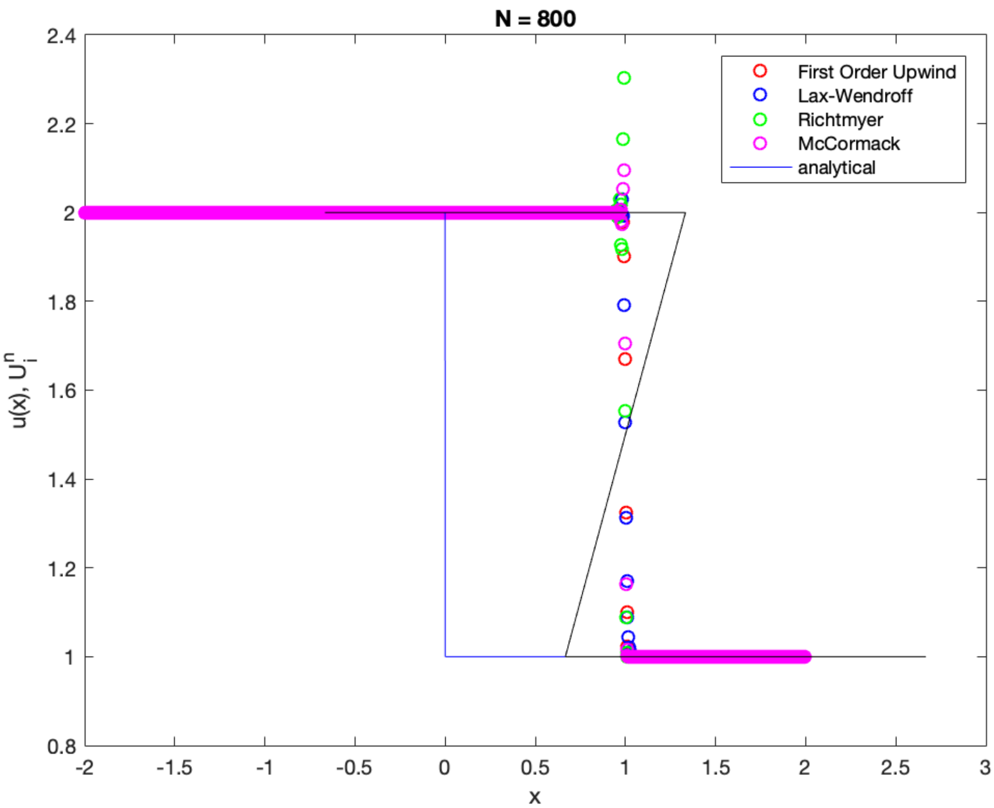
Published with MATLAB® R2017a

(2) As expected, increasing the number of gridpoints decreased the error significantly. Lax-Wendroff was more likely to overshoot the analytical solution, while first-order upwind was more likely to undershoot.

```
clear all; close all; clc
% ready to submit

%method='first-order-upwind';
%method='lax-wendroff';
%method='richtmyer';
%method='maccormack';
uh=@step; uL=2; uR=1; L=2; N=50; T=2/3; name='step';
%uh=@hump; uL=1; uR=1; L=2; N=50; T=1/pi-1e-5; name='hump';


method = {'first-order-upwind' 'lax-wendroff' 'richtmyer' 'maccormack'};
line = {'ro','bo','go','mo'};
N = [50 200 800];

fh=@burgers; % exact flux function f=f(u)

sigma=0.75;


for i = 1:3       % loop through n
    figure
    for j = 1:4   % loop through methods
        [xm,U]=advanceconservative(uh,fh,uL,uR,L,sigma,N(i),T,method{j});
        x_=linspace(-L,L,10000);
        [xi,ui]=burgersanalytical(x_,uh,T);
        plot(xm,U,line{j});
        hold on
    end

    %axis([-0.5,L,0,2.5]);
    plot(xi, feval(uh,xi),'b-')
    hold on
    plot(xi, ui, 'k-')
    hold off

    title(sprintf('N = %d',N(i)));
    xlabel('x'); ylabel('u(x), U^n_i');
    legend('First Order Upwind','Lax-Wendroff','Richtmyer','McCormack','analytical');
end
```


N = 50
u(x), U_i^n
x
First Order Upwind
Lax-Wendroff
Richtmyer
McCormack
analytical


N = 200
u(x), U_i^n
x
First Order Upwind
Lax-Wendroff
Richtmyer
McCormack
analytical



Published with MATLAB® R2017a

(3) Like in the hump case, increasing the number of gridpoints decreased the error significantly. Again Lax-Wendroff was more likely to overshoot the analytical solution. Interestingly, L-W showed signs of oscillatory behavior around the step.