1a.) $x_1 = 0 \quad x_5 = 2\pi$

$$u_1^{n+1} = u_1^n - \frac{\sigma}{2}\left(u_2^{n+1} - u_0^{n+1}\right)$$
$$\hookrightarrow = u_1^n - \frac{\sigma}{2}\left(u_2^{n+1} - u_4^{n+1}\right) \rightsquigarrow$$
$$u_2^{n+1} = u_2^n - \frac{\sigma}{2}\left(u_3^{n+1} - u_1^{n+1}\right)$$
$$u_3^{n+1} = u_3^n - \frac{\sigma}{2}\left(u_4^{n+1} - u_2^{n+1}\right)$$
$$u_4^{n+1} = u_4^n - \frac{\sigma}{2}\left(u_5^{n+1} - u_3^{n+1}\right)$$
$$u_5^{n+1} = u_5^n - \frac{\sigma}{2}\left(u_2^{n+1} - u_4^{n+1}\right)$$

$\frac{\sigma}{2} u_2^{N+1} \qquad u_5^{N+1} - \frac{\sigma}{2} u_4^{N+1} = \dots$

$A\,U^{N+1} = b$

$$u_1^{n+1} + \frac{\sigma}{2}\left[u_2^{n+1} - u_4^{n+1}\right] = u_1^n$$
$$u_2^{n+1} + \frac{\sigma}{2}\left[u_3^{N+1} - u_1^{N+1}\right] = u_1^n$$

$$\begin{bmatrix} 1 & \sigma/2 & 0 & -\sigma/2 & 0 \\ -\frac{\sigma}{2} & 1 & \sigma/2 & 0 & 0 \\ & -\sigma/2 & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \begin{bmatrix} u_1^{N+1} \\ u_2^{N+1} \\ \\ \dots \\ u_5^{N+1} \end{bmatrix} = \begin{bmatrix} u_1^N \\ u_2^N \\ \dots \\ u_5^N \end{bmatrix}$$

$\underline{A U^{N+1} = b}$

$$\begin{bmatrix} 1 & \sigma/2 & 0 & -\sigma/2 & 0 \\ -\sigma/2 & 1 & \sigma/2 & 0 & 0 \\ 0 & -\sigma/2 & 1 & \sigma/2 & 0 \\ 0 & 0 & -\sigma/2 & 1 & \sigma/2 \\ 0 & \sigma/2 & 0 & -\sigma/2 & 1 \end{bmatrix} \begin{bmatrix} u_1^{N+1} \\ u_2^{N+1} \\ u_3^{N+1} \\ u_4^{N+1} \\ u_5^{N+1} \end{bmatrix} = \begin{bmatrix} u_1^N \\ u_2^N \\ u_3^N \\ u_4^N \\ u_5^N \end{bmatrix}.$$

10) $u_i^{n+1} = u_i^n - \frac{\sigma}{2}\left(u_{i+1}^n - u_{i-1}^n\right) + \frac{\sigma^2}{2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right)$ $\sigma = \frac{k\epsilon}{h}$

$i=1$ $u_1^{n+1} = u_1^n - \frac{\sigma}{2}\left(u_2^n - \cancel{u_0^n}\right) + \frac{\sigma^2}{2}\left(u_2^n - 2u_1^n + \cancel{u_0^n}\right)$ $h = \frac{k}{\sigma}$ $k = \sigma h$

$\rightarrow$ $= u_1^n - \frac{\sigma}{2}\left(u_2^n - u_4^n\right) + \frac{\sigma^2}{2}\left(u_2^n - 2u_1^n + u_4^n\right)$

$i=2$ $u_2^{n+1} = u_2^n - \frac{\sigma}{2}\left(u_3^n - u_1^n\right) + \frac{\sigma^2}{2}\left(u_3^n - 2u_2^n + u_1^n\right)$ $\frac{k^2}{h^2}$

$i=3$ $u_3^{n+1} = u_3^n - \frac{\sigma}{2}\left(u_4^n - u_2^n\right) + \frac{\sigma^2}{2}\left(u_4^n - 2u_3^n + u_2^n\right)$

$i=4$: $u_4^{n+1} = u_4^n - \frac{\sigma}{2}\left(u_5^n - u_3^n\right) + \frac{\sigma^2}{2}\left(u_5^n - 2u_4^n + u_3^n\right)$ $\frac{u_{xxx}}{6}\left(\frac{\sigma^3 h^3}{k} - \frac{\sigma h^3}{k}\right)$

$i=5$ $u_5^{n+1} = u_5^n - \frac{\sigma}{2}\left(\cancel{u_6^n} - u_4^n\right) + \frac{\sigma^2}{2}\left(\cancel{u_6^n} - 2u_5^n + u_4^n\right)$

$\hookrightarrow$ $u_5^{n+1} = u_5^n - \frac{\sigma}{2}\left(u_2^n - u_4^n\right) + \frac{\sigma}{2}\left(u_2^n - 2u_5^n + u_4^n\right)$ $\frac{u_{xxxx}}{24}\left(\frac{\sigma^2 h^4}{k} - \frac{\sigma^4 h^4}{k}\right)$

error: $L_h(x,t) = -\frac{h^2}{6}\frac{h}{k}\sigma\left(1-\sigma^2\right)u_{xxx} + \frac{h^3}{24}\frac{h}{k}\sigma^2\left(1-\sigma^2\right)u_{xxxx}$

$u_{i-1} = u_i - u_x h + u_{xx}\frac{h^2}{2} - u_{xxx}\frac{h^3}{6} + u_{xxxx}\frac{h^4}{24}$, $u^{n+1} = u^n + u_t k + u_{tt}\frac{k^2}{2} + u_{ttt}\frac{k^3}{6} + u_{tttt}\frac{k^4}{24}$

$u_{i+1} = u_i + u_x h + u_{xx}\frac{h^2}{2} + u_{xxx}\frac{h^3}{6} + u_{xxxx}\frac{h^4}{24}$

$u_t = -u_x$
$u_{tt} = u_{xx}$
$u_{ttt} = -u_{xxx}$
$u_{tttt} = u_{xxxx}$

$u_i^{n+1} - u_i^n + \frac{\sigma}{2}\left(u_{i+1}^n - u_{i-1}^n\right) - \frac{\sigma^2}{2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) = 0$

$\div k$:

$\frac{1}{k}u_i^{n+1} - \frac{1}{k}u_i^n + \frac{\sigma}{2}\frac{1}{k}\left(u_{i+1}^n - u_{i-1}^n\right) - \frac{\sigma^2}{2}\frac{1}{k}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) = 0$

$\frac{1}{k}\left(u + u_t k + u_{tt}\frac{k^2}{2} + u_{ttt}\frac{k^3}{6} + u_{tttt}\frac{k^4}{24}\right) - \frac{\sigma^2}{2}\frac{1}{k}\left(\cancel{-2u_i^n + 2u_i^n} + u_{xx}h^2 + \frac{1}{12}u_{xxxx}h^4\right) + \frac{\sigma}{2}\frac{1}{k}\left(2u_x h + \frac{1}{3}u_{xxx}h^3\right) - \frac{1}{k}u = 0$

$\frac{1}{k}\left(u - u_x k + u_{xx}\frac{k^2}{2} - u_{xxx}\frac{k^3}{6} + u_{xxxx}\frac{k^4}{24}\right) - \frac{\sigma^2}{2}\frac{1}{k}\left(u_{xx}h^2 + \frac{1}{12}h^4 u_{xxxx}\right) + \frac{\sigma}{2}\frac{1}{k}\left(2u_x h + \frac{1}{3}u_{xxx}h^3\right) - \frac{1}{k}u = 0$

$\frac{1}{k}\left(u - u_x k + u_{xx}\frac{k^2}{2} - u_{xxx}\frac{k^3}{6} + u_{xxxx}\frac{k^4}{24}\right) + \frac{1}{2h}\left(2u_x h + \frac{1}{3}u_{xxx}h^3\right) - \frac{k}{2h^2}\left(u_{xx}h^2 + \frac{1}{12}h^4 u_{xxxx}\right) - \frac{1}{k}u = 0$

$\frac{1}{k}\left[u - u_x k + u_{xx}\frac{k^2}{2} - u_{xxx}\frac{k^3}{6} + u_{xxxx}\frac{k^4}{24}\right] + u_x + u_{xxx}\frac{h^2}{6} - \frac{k}{2}u_{xx} - \frac{k}{24}h^2 u_{xxxx}$

$\boxed{\text{When } \sigma, \frac{h}{k} \text{ are constant, } L_h(x,t) \text{ has order } O(h^2)}$

$\frac{1}{k}u - \cancel{u_x} + \cancel{u_{xx}\frac{k}{2}} - u_{xxx}\frac{k^2}{6} + u_{xxxx}\frac{k^3}{24} + \cancel{u_x} + u_{xxx}\frac{h^2}{6} - \cancel{u_{xx}\frac{k}{2}} - u_{xxxx}\frac{kh^2}{24}$

$\frac{1}{k}\left[u - \frac{u_{xxx}}{6}\left[k^2 - h^2\right] + \frac{u_{xxxx}}{24}\left[k^3 - kh^2\right]\right] = 0$

$L \rightarrow L(x,t) = \frac{1}{k}\left[\frac{u_{xxx}k}{6}\left[k^2 - h^2\right] - \frac{u_{xxxx}}{24}k\left[k^3 - kh^2\right]\right]$

$\boxed{L(x,t) = -\frac{h^2}{6}\frac{h}{k}\sigma\left(1-\sigma^2\right)u_{xxx} + \frac{h^3}{24}\frac{h}{k}\sigma^2\left(1-\sigma^2\right)u_{xxxx}}$ + h.o.t.

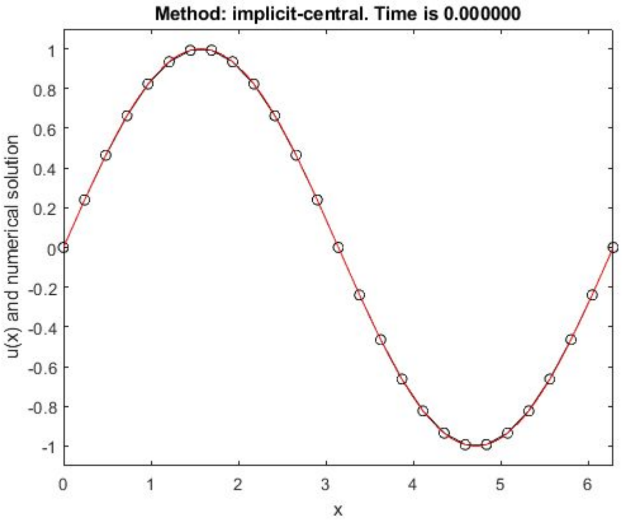
$L(x,t) = \left(k^3 - kh^2\right)\frac{u_{xxx}}{6k} - \left(k^4 - k^2h^2\right)\frac{u_{xxxx}}{24k} = \left(\sigma^3 h^3 - \sigma h^3\right)\frac{u_{xxx}}{6k} - \left(\sigma^4 h^4 - \sigma^2 h^4\right)\frac{u_{xxxx}}{24k}$

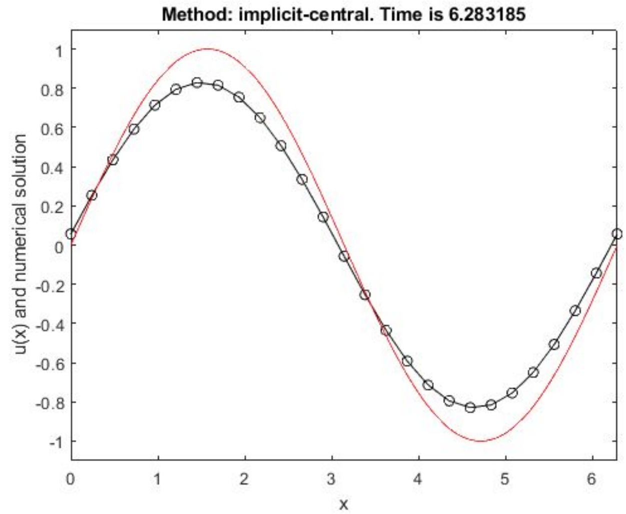
```
function out=wave_solve(c,L,n,sigma,T,M,u0,method)

%
% --inputs--
% c:        advective speed
% L:        domain size [0,L]
% n:        number of interior grid points
% sigma:    Courant number
% T:        final time
% M:        number of solutions recorded between [0,T]
% u0:       function that prescribes the initial conditions u0(x)
% method:   integration method, one of:
%           'forward-upwind'
%           'implicit-central'
%           'beam-warming'
%           'lax-wendroff'
%         ...
%
% --outputs--
% out.h   grid spacing
% out.k   time step size
% out.l   number of time steps taken
% out.x:  spatial locations so that out.x(1)=0 and out.x(end)=L
% out.TT: out.TT(1)=0 and out.TT(end)=T with
%         length(out.TT)=M+2;
% out.U:  numerical solution as matrix
%         out.U(:,j) is the numerical solution at time out.TT(j)
%         with j=1,\dots,M+2
%         size(out.U,1)=length(out.x)
%         size(out.U,2)=length(out.TT)
%

% set output to empty
out=[];

% work on grid
h=L/(n+1); % grid spacing recovered from the number of interior points
out.h = h; % store it

out.x=[0:h:L]; % actual grid array, including x=0 and x=L
N=length(out.x); % number of overall points

% time outputs
out.TT=linspace(0,T,M+2);

% build the matrix for the updates
switch lower(method)

case 'forward-upwind'

  if ( c<0 )
      error('please specify a positive advective speed');
  end

  A = -diag(ones(N,1),0) - ...
      -diag(ones(N-1,1),-1);
  A(1,n+1)=1; % periodic boundary on U(1)=U_0

case 'implicit-central'
  if ( c<0 )
      error('please specify a positive advective speed');
  end

  A = diag(zeros(N,1),0) + diag(ones(N-1,1),1) - diag(ones(N-1,1),-1);
  % create a tridiagonal matrix with -1, 0, 1
  A = .5*A;
```

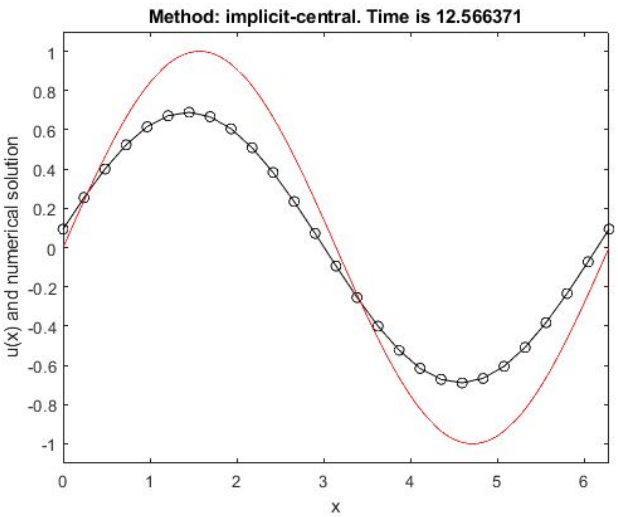
```
  C = diag(ones(N,1),0); % create a diagonal matrix with 1

  % we have to create two separate matrices for implicit central due to it
  % being an implicit method


  % set boundary conditions based on finite difference method equation
  A(1,N-1) = -.5;
  A(N,2) = .5;


case 'beam-warming'
  if ( c<0 )
      error('please specify a positive advective speed');
  end


  A = zeros(N) + diag(ones(N,1),0) * 3 + diag(ones(N-1,1),-1) * -4 + ...
      diag(ones(N-2,1),-2);
  % create a slightly shifted tridiagonal matrix with 1,-4,3
  A(1,N-2) = 1;
  A(1,N-1) = -4;
  A(2,N-1) = 1;
  % set boundary conditions based on finite difference method equation

  B = zeros(N) + diag(ones(N,1),0) * 1 + diag(ones(N-1,1),-1) * -2 + ...
      diag(ones(N-2,1),-2);
  B(1,N-2) = 1;
  B(1,N-1) = -2;
  B(2,N-1) = 1;
  % create a second tridiagonal matrix. We have to do this due to the
  % nature of the beam warming equation having different coefficients




case 'lax-wendroff'
  % create a tridiagonal matrix with -1,0,1
  A = zeros(N) + diag(ones(N-1,1),1) - diag(ones(N-1,1),-1);
  A(1,N-1) = -1;
  A(N,2) = 1;

  % tridiagonal matrix with 1,-2,1
  B = zeros(N) -2* diag(ones(N,1),0) + diag(ones(N-1,1),1) + ...
      diag(ones(N-1,1),-1);
  B(1,N-1) = 1;
  B(N,2) = 1;



otherwise
  error('method is unknown');
end

% time step size recovered from Courant number
k=sigma*h/c;
out.k = k; % store it

% initial conditions
U_=u0(out.x)'; t=0; j=1;

% store initial conditions
out.U(:,j)=U_; j=j+1;

% integrate in time
l=0;
while t<out.TT(end)
```

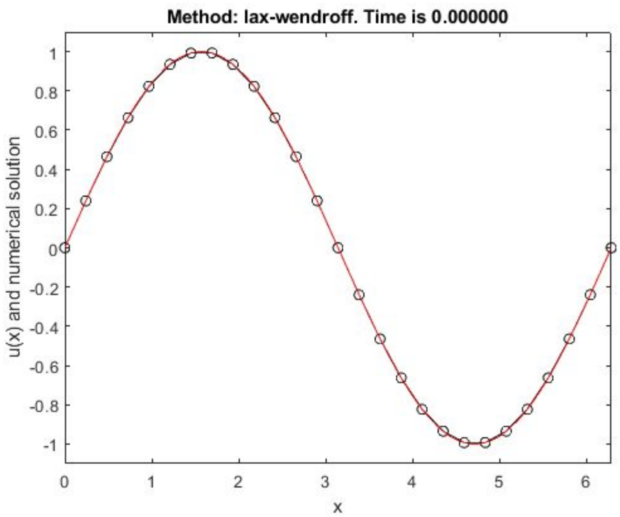
```
    % pick the smallest between the time step
    % and the time step to get to the next out.TT(j)
    k_=min([k,(out.TT(j)-t)]);
    sigma_=k_*c/h;

    fprintf('Time: %f; Sigma = %f; Time step = %f\n',t,sigma_,k_);

    % zero the update
    dU_=zeros(size(U_));

    switch lower(method)

    case {'forward-upwind'}

       dU_=sigma_*A*U_; % Euler fwd step

    case {'implicit-central'}

       u_next = ((sigma_*A)+C) \ (U_);
       dU_ = -sigma_*A*u_next;
       % we have to use a temporary variable u_next due to the implicit
       % nature of this method. First we must compute our coefficient matrix
       % divided by U_, then we plug in the result into U_

    case 'beam-warming'

       dU_ = (-(sigma_ / 2) * A + (sigma_^2 / 2) * B) * U_;
       % multiply the coefficient matrices by sigma/2, then multiply by U_
       % to find the majority of the rhs of the beam warming equation

    case 'lax-wendroff'

       dU_ = (-(sigma_ / 2) * A + (sigma_^2 / 2) * B) * U_;
       % multiply the coefficient matrices by sigma/2, sigma^2/2 then
       % multiply by U_ to find the majority of the rhs of the lax-wendroff
       % equation

    otherwise
      error('method is unknown');
    end

    % update
    U_=U_+dU_;

    % advance time to reflect update
    t=t+k_;
    l=l+1; % step counter

    % store
    if ( t==out.TT(j) )
      out.TT(j)=t; % adjust recorded time
      out.U(:,j)=U_; j=j+1;
    end

end

out.l=l; % number of steps
```

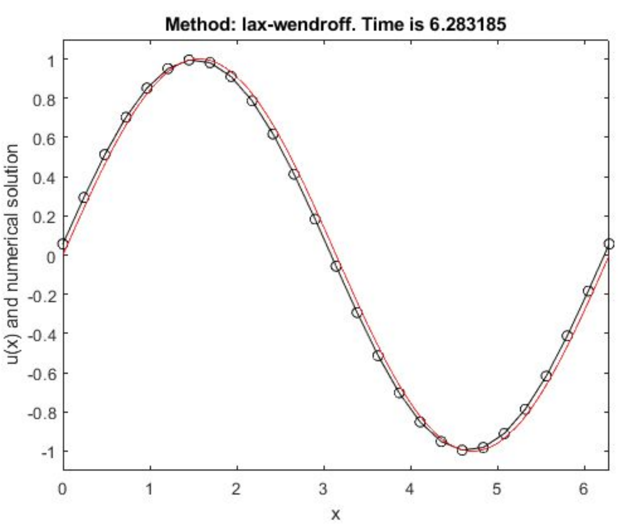
```
Not enough input arguments.

Error in wave_solve (line 37)
h=L/(n+1); % grid spacing recovered from the number of interior points
```

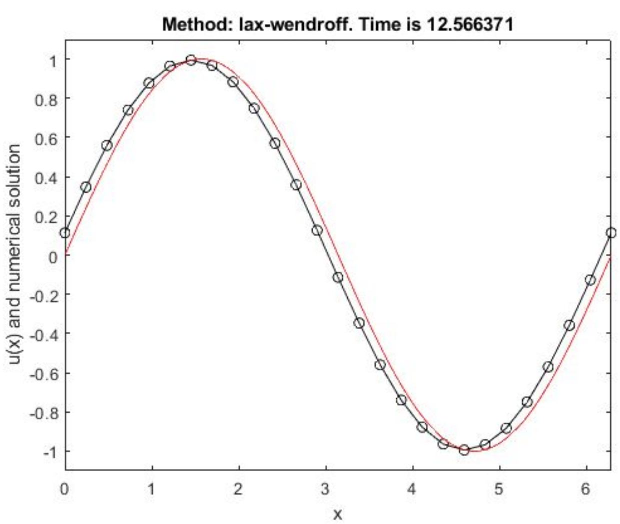

Method: implicit-central. Time is 0.000000
u(x) and numerical solution
x

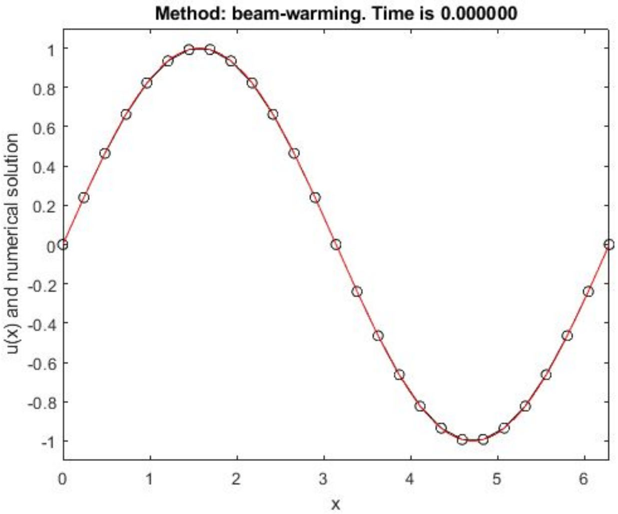

Method: implicit-central. Time is 6.283185
u(x) and numerical solution
x

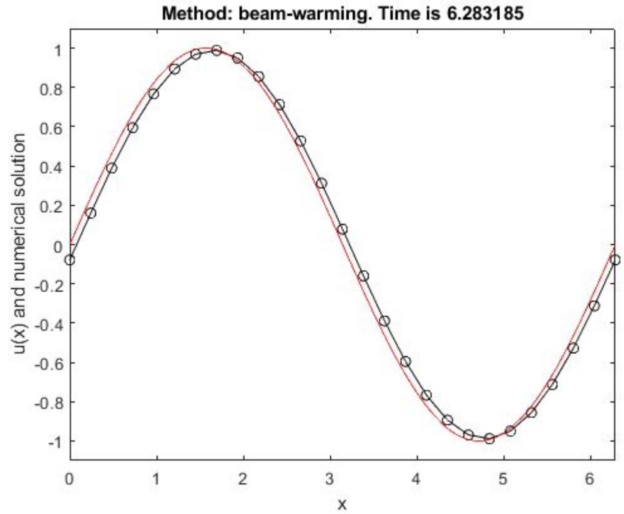

Method: implicit-central. Time is 12.566371
u(x) and numerical solution
x

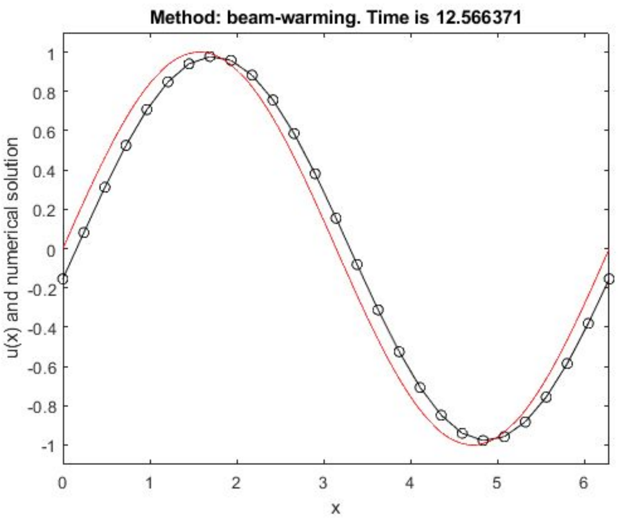

Method: lax-wendroff. Time is 0.000000
u(x) and numerical solution
x


Method: lax-wendroff. Time is 6.283185
u(x) and numerical solution
x


Method: lax-wendroff. Time is 12.566371
u(x) and numerical solution
x

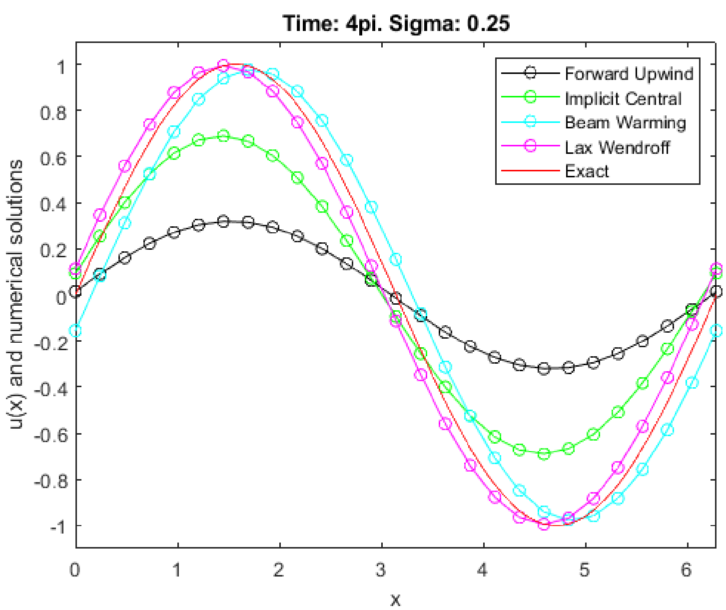

Method: beam-warming. Time is 0.000000
u(x) and numerical solution
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6

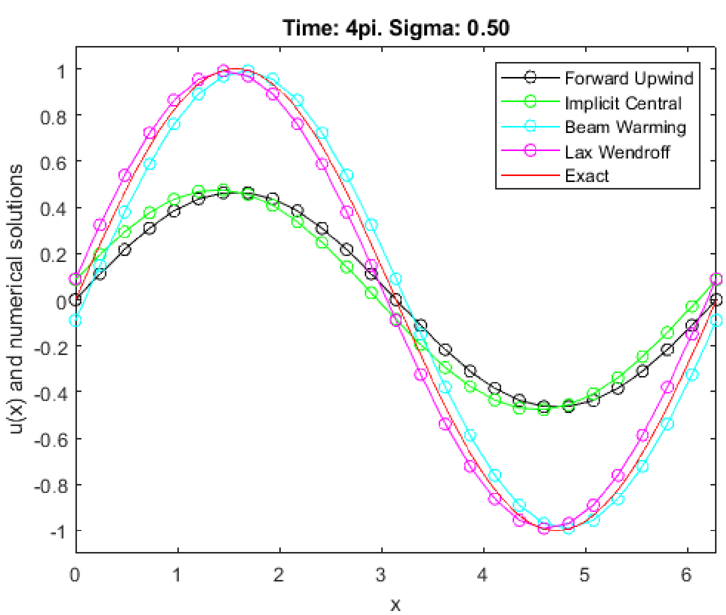

Method: beam-warming. Time is 6.283185
u(x) and numerical solution
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6

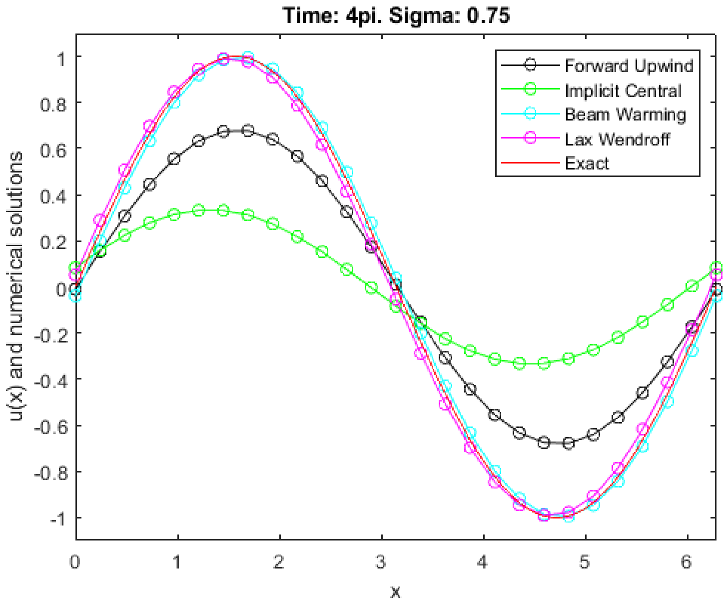

Method: beam-warming. Time is 12.566371
u(x) and numerical solution
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6

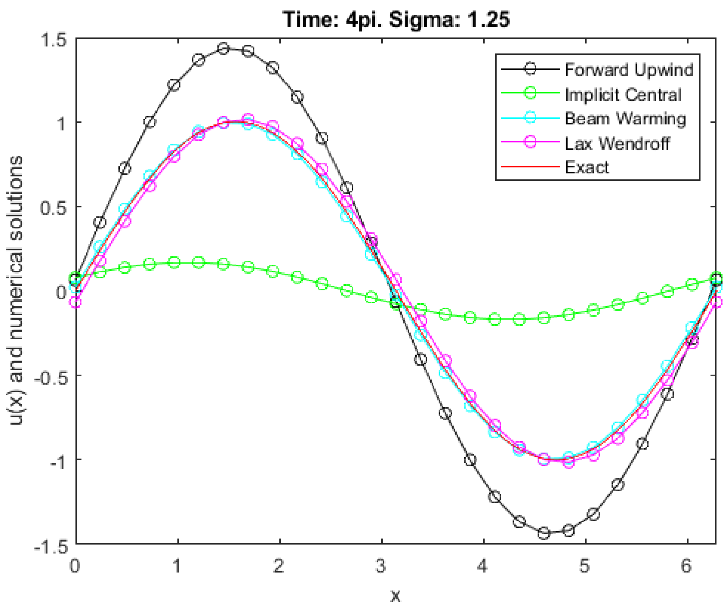
```
clear all; close all; clc;

c=1;        % advective speed
L=2*pi;     % computational domain [0,L]
T=2*2*pi;   % end time
M=0;        % intermediate solutions

fexact='exact.dat';

sigma= 0.25; % Courant number
%sigma = {.25,.5,.75,1.25};
n=25;       % number of interior points

%method='forward-upwind';
%method='implicit-central';
%method='beam-warming';
%method='lax-wendroff';
method = {'forward-upwind','implicit-central','beam-warming', ...
    'lax-wendroff'};


% initial conditions
u0 = @(x) sin(x);  % anonymous function

% solve
%out=wave_solve(c,L,n,sigma,T,M,u0,method);

% plot
xx=linspace(0,L,1000);

figure
sigma = .25;
colors = {'ko-','go-','co-','mo-'};
for k = 1:4
    clear out
    out=wave_solve(c,L,n,sigma,T,M,u0,method{k});

    plot(out.x,out.U(:,2),colors{k});
    hold on
    axis([0,L,-1.1,1.1]);
    xlabel('x');
    ylabel('u(x) and numerical solutions');
    title(sprintf('Time: 4pi. Sigma: %.2f',sigma));
end
plot(xx,u0(xx-out.TT(2)),'r-');
legend('Forward Upwind','Implicit Central','Beam Warming',...
    'Lax Wendroff','Exact')


figure
sigma = .5;
colors = {'ko-','go-','co-','mo-'};
for k = 1:4
    clear out
    out=wave_solve(c,L,n,sigma,T,M,u0,method{k});

    plot(out.x,out.U(:,2),colors{k});
    hold on
    axis([0,L,-1.1,1.1]);
    xlabel('x');
    ylabel('u(x) and numerical solutions');
    title(sprintf('Time: 4pi. Sigma: %.2f',sigma));
end
plot(xx,u0(xx-out.TT(2)),'r-');
legend('Forward Upwind','Implicit Central','Beam Warming',...
    'Lax Wendroff','Exact')


figure
sigma = .75;
colors = {'ko-','go-','co-','mo-'};
for k = 1:4
    clear out
    out=wave_solve(c,L,n,sigma,T,M,u0,method{k});

    plot(out.x,out.U(:,2),colors{k});
    hold on
    axis([0,L,-1.1,1.1]);
    xlabel('x');
    ylabel('u(x) and numerical solutions');
    title(sprintf('Time: 4pi. Sigma: %.2f',sigma));
end
plot(xx,u0(xx-out.TT(2)),'r-');
legend('Forward Upwind','Implicit Central','Beam Warming',...
```

```
    'Lax Wendroff','Exact')


figure
sigma = 1.25;
colors = {'ko-','go-','co-','mo-'};
for k = 1:4
    clear out
    out=wave_solve(c,L,n,sigma,T,M,u0,method{k});

    plot(out.x,out.U(:,2),colors{k});
    hold on
    axis([0,L,-1.5,1.5]);
    xlabel('x');
    ylabel('u(x) and numerical solutions');
    title(sprintf('Time: 4pi. Sigma: %.2f',sigma));
end
plot(xx,u0(xx-out.TT(2)),'r-');
legend('Forward Upwind','Implicit Central','Beam Warming',...
    'Lax Wendroff','Exact')



%dump
%fout=sprintf('%s_n%g_sigma%f.dat',method,n,sigma);
%dlmwrite(fout,[out.x',out.U],'delimiter',' ','precision','%e');
%dlmwrite(fexact,[xx',exact],'delimiter',' ','precision','%e');
```



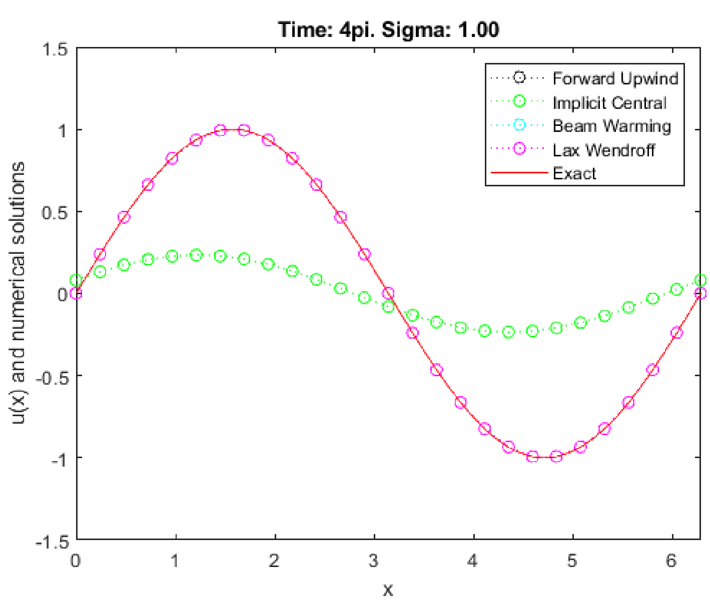






Published with MATLAB® R2018b

```
clear all; close all; clc;

c=1;        % advective speed
L=2*pi;     % computational domain [0,L]
T=2*2*pi;   % end time
M=0;        % intermediate solutions

fexact='exact.dat';

sigma= 0.25; % Courant number
%sigma = {.25,.5,.75,1.25};
n=25;       % number of interior points

%method='forward-upwind';
%method='implicit-central';
%method='beam-warming';
%method='lax-wendroff';
method = {'forward-upwind','implicit-central','beam-warming', ...
    'lax-wendroff'};


% initial conditions
u0 = @(x) sin(x);  % anonymous function

% solve
%out=wave_solve(c,L,n,sigma,T,M,u0,method);

% plot
xx=linspace(0,L,1000);



figure
sigma = 1;
colors = {'ko:','go:','co:','mo:'};
for k = 1:4
    clear out
    out=wave_solve(c,L,n,sigma,T,M,u0,method{k});

    plot(out.x,out.U(:,2),colors{k});
    hold on
    axis([0,L,-1.5,1.5]);
    xlabel('x');
    ylabel('u(x) and numerical solutions');
    title(sprintf('Time: 4pi. Sigma: %.2f',sigma));
end
plot(xx,u0(xx-out.TT(2)),'r-');
legend('Forward Upwind','Implicit Central','Beam Warming',...
    'Lax Wendroff','Exact')




%dump
%fout=sprintf('%s_n%g_sigma%f.dat',method,n,sigma);
%dlmwrite(fout,[out.x',out.U],'delimiter',' ','precision','%e');
%dlmwrite(fexact,[xx',exact],'delimiter',' ','precision','%e');
```



For $\sigma = 1.00$, the graph is nearly useless. All the explicit methods lie completely on top of the exact solution, which is to be expected. In all the explicit methods, we see a $[1-\sigma]$ or $[1-\sigma^2]$ term show up in the truncation error formula. With $\sigma = 1$, it's clear that truncation error is, and should be, 0.