In [1]:

```
Autosave disabled
```

**Chapter 31-7: Hyperbolic PDEs Using Finite Difference Method.**

In mathematics, a hyperbolic partial differential equation of order $n$ is a partial differential equation (PDE) that, roughly speaking, has a well-posed initial value problem for the first $n-1$ derivatives. More precisely, the Cauchy problem can be locally solved for arbitrary initial data along any non-characteristic hypersurface. Many of the equations of mechanics are hyperbolic, and so the study of hyperbolic equations is of substantial contemporary interest. The model hyperbolic equation is the wave equation. In one spatial dimension, this is

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

The equation has the property that, if $u$ and its first time derivative are arbitrarily specified initial data on the line $t = 0$ (with sufficient smoothness properties), then there exists a solution for all time $t$.

The solutions of hyperbolic equations are "wave-like". If a disturbance is made in the initial data of a hyperbolic differential equation, then not every point of space feels the disturbance at once. Relative to a fixed time coordinate, disturbances have a finite propagation speed. They travel along the characteristics of the equation. This feature qualitatively distinguishes hyperbolic equations from elliptic partial differential equations and parabolic partial differential equations. A perturbation of the initial (or boundary) data of an elliptic or parabolic equation is felt at once by essentially all points in the domain.

A finite difference is a mathematical expression of the form $f(x+b) - f(x+a)$. If a finite difference is divided by $b-a$, one gets a difference quotient. The approximation of derivatives by finite differences plays a central role in finite difference methods for the numerical solution of differential equations, especially boundary value problems.

The difference operator, commonly denoted $\Delta$, is the operator that maps a function $f$ to the function $\Delta[f]$ defined by

$$\Delta[f](x) = f(x+1) - f(x)$$

A difference equation is a functional equation that involves the finite difference operator in the same way as a differential equation involves derivatives. There are many similarities between difference equations and differential equations, specially in the solving methods. Certain recurrence relations can be written as difference equations by replacing iteration notation with finite differences.

From Wikipedia, the free encyclopedia

The Lax–Wendroff method, named after Peter Lax and Burton Wendroff, is a numerical method for the solution of hyperbolic partial differential equations, based on finite differences. It is second-order accurate in both space and time. This method is an example of explicit time integration where the function that defines the governing equation is evaluated at the current time.

6. Solve Burgers equation and provide a plot of the resulting solution.

In [ ]:

```
clear all
close all

%-----
% animation of the solution to burgers equation
% using different finite-difference methods
%------

r=0.50;

N=2*2*32;
a=-2;b= 2;
```

```
Dx=(b-a)/N;

Dt=r*Dx;

method=1;  % Lax
method=2;  % Lax_Wendroff

%---
% initial condition
%----

for i=1:N+1
 x(i)=a+(i-1)*Dx;
 f(i) = exp(-x(i)*x(i));
 q(i) = 0.5*f(i)^2;
end


%---------
for step=1:30000

for i=2:N
 if(method==1)%--- Lax
  fnew(i) = 0.5*(1+r*f(i-1))*f(i-1)+0.5*(1-r*f(i+1))*f(i+1);
 elseif(method==2)%--- Lax-Wendroff
   A = f(i-1)+f(i);
   B = f(i-1)+2*f(i)+f(i+1);
   C = f(i)+f(i+1);
   fnew(i) = f(i) + 0.5*r*(q(i-1)-q(i+1)) + 0.25*r*r*(A*q(i-1)-B*q(i)+C*q(i+1));
 end
%---
end

f(2:N)=fnew(2:N);

for i=1:N+1
 q(i) = 0.5*f(i)^2;
end

if(step==1)
  Handle1 = plot(x,f,'o-');
  set(Handle1, 'erasemode', 'xor');
  set(gca,'fontsize',15)
  axis([a b 0.0 1.2])
  xlabel('x','fontsize',15)
  ylabel('f','fontsize',15)
else
  set(Handle1,'XData',x,'YData',f);
  pause(0.1)
  drawnow
end

end
```

7. Use the Lax-Wendroff scheme to solve the hyperbolic equation

$$ut + ux = 0 \qquad 0 \le x \le 1, \quad 0 < t < 0.5$$

with proper boundary condition at $x = 0$ and initial condition:

$$u(x, 0) = \exp(-c(x - 0.5)2) \qquad 0 \le x \le 1$$

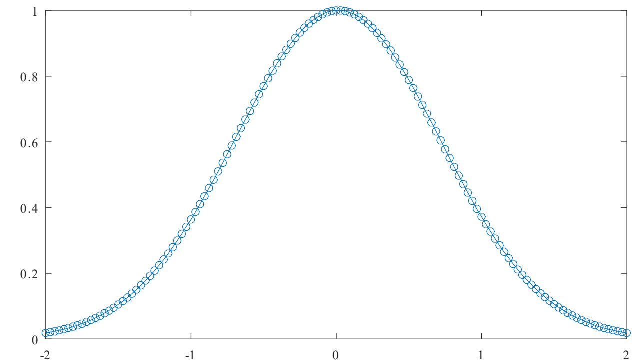
such that the analytic solution is a smooth Gaussian wave

$$u(x, t) = \exp(-c(x - t - 0.5)2)$$

where $c > 0$ is a constant determining the narrowness of the wave.

(The larger $c$ is, the narrower the wave will be.)

The close correlation shown by the tracks of the two marker symbols testifies that in this case the two methods employed produce essentially identical results.




The example shown above uses the Lax-Wendroff scheme to solve the hyperbolic equation ut + ux =0 0 ≤ x ≤ 1, 0 <t< 0.5, with proper boundary condition at x = 0 and initial condition: u(x, 0) = exp(−c(x − 0.5)2), 0 ≤ x ≤ 1, such that the analytic solution is a smooth Gaussian wave u(x, t) = exp(−c(x − t − 0.5)2), where c > 0 is a constant determining the narrowness of the wave. The larger c is, the narrower the wave will be.

Following is another (Matlab) example of Lax-Wendroff on a hyperbolic equation, using finite differences.

Show an example using the Lax-Wendroff scheme to solve the hyperbolic equation

$$ut + ux = 00 \leq x \leq 1, 0 < t < 0.5$$

with proper boundary condition at x = 0, and initial condition:

$$u(x, 0) = exp(-c(x - 0.5)2), 0 \leq x \leq 1$$

such that the analytic solution is a smooth Gaussian wave

$$u(x, t) = exp(-c(x - t - 0.5)2)$$

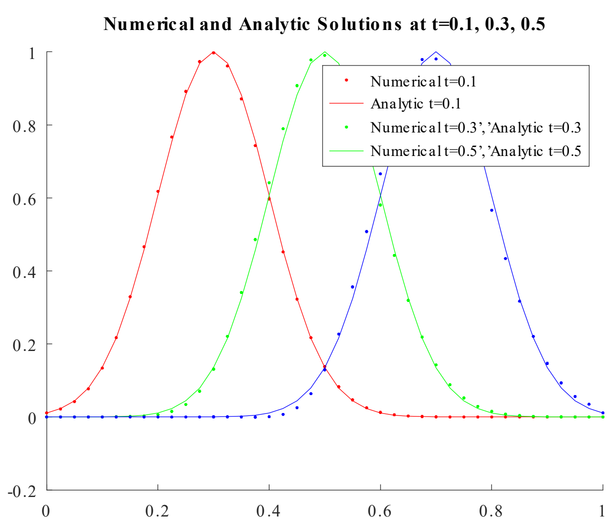
where c > 0 is a constant determining the narrowness of the wave. (The larger c is, the narrower the wave will be.)

In [ ]:

```
%--------------------------------------------------------------
% hyper1d.m:
% use Lax-Wendroff scheme to solve the hyperbolic equation
% u_t(x,t) + u_x(x,t) = 0, xl < x < xr, 0 < t < tf
% u(x, 0) = f(x), xl < x < xr
% u(0, t) = g(t), 0 < t < tf
%% A special case is choosing f and g properly such that the
% The analytic solution is:
% u(x,t)= f(x-t)=e^(-10(x-t-0.2)^2)
%--------------------------------------------------------------
clear all; % clear all variables in memory
xl=0; xr=1; % x domain [xl,xr]
J = 40; % J: number of division for x
dx = (xr-xl) / J; % dx: mesh size
tf = 0.5; % final simulation time
Nt = 50; % Nt: number of time steps
dt = tf/Nt;
c = 50; % parameter for the solution
mu = dt/dx;
if mu > 1.0 % make sure dt satisy stability condition
  error('mu should < 1.0!')
end
% Evaluate the initial conditions
x = xl : dx : xr; % generate the grid point
f = exp(-c*(x-0.2).^2); % dimension f(1:J+1)
% store the solution at all grid points for all time steps
u = zeros(J+1,Nt);
% Find the approximate solution at each time step
  for n = 1:Nt
```

```
  t = n*dt; % current time
  gl = exp(-c*(xl-t-0.2)^2); % BC at left side
  gr = exp(-c*(xr-t-0.2)^2); % BC at right side
  if n==1 % first time step
    for j=2:J % interior nodes
    u(j,n) = f(j) - 0.5*mu*(f(j+1)-f(j-1)) + ...
                  0.5*mu^2*(f(j+1)-2*f(j)+f(j-1));
    end
    u(1,n) = gl; % the left-end point
    u(J+1,n) = gr; % the right-end point
  else
    for j=2:J % interior nodes
      u(j,n) = u(j,n-1) - 0.5*mu*(u(j+1,n-1)-u(j-1,n-1)) + ...
                  0.5*mu^2*(u(j+1,n-1)-2*u(j,n-1)+u(j-1,n-1));
    end
    u(1,n) = gl; % the left-end point
    u(J+1,n) = gr; % the right-end point
  end
  % calculate the analytic solution
  for j=1:J+1
    xj = xl + (j-1)*dx;
    u_ex(j,n)=exp(-c*(xj-t-0.2)^2);
  end
end

% plot the analytic and numerical solution at different times
figure;
hold on;
n=10;
plot(x,u(:,n),'r.',x,u_ex(:,n),'r-'); % r for red
n=30;
plot(x,u(:,n),'g.',x,u_ex(:,n),'g-');
n=50;
plot(x,u(:,n),'b.',x,u_ex(:,n),'b-');
legend('Numerical t=0.1','Analytic t=0.1',...
      'Numerical t=0.3','Analytic t=0.3',...
      'Numerical t=0.5','Analytic t=0.5');
```

In the following plot, the analytical track does not adhere as closely as before to the numerical track.

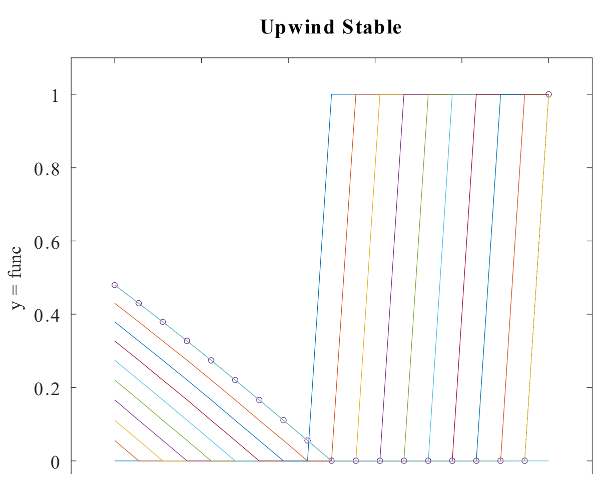


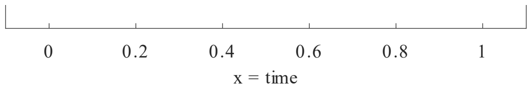

6. Develop an upwind scheme (within finite differences) for the equation $u_t + au_x = 0$.

Below is a plot of the initial and consecutive approximation of the upwinding method for an advection equation. The time step is Δt = h and the scheme is stable.

In [ ]:

```
clear; close all

 a = 0;  b=1; tfinal = 0.5

 m = 20;
 m=18;

 h = (b-a)/m;
 k = h; mu = k/h;

 t = 0;
 n = fix(tfinal/k);
 y1 = zeros(m+1,1); y2=y1; x=y1;

 for i=1:m+1,
   x(i) = a + (i-1)*h;
   y1(i) = uexact(t,x(i));
   y2(i) = 0;
 end

figure(1); plot(x,y1); hold
axis([-0.1 1.1 -0.1 1.1]);
title('Upwind Stable');
xlabel('x = time','fontsize',10)
ylabel('y = func','fontsize',10)


 t = 0;
 for j=1:n,

   y1(1)=bc(t); y2(1)=bc(t+k);
%  for i=2:m+1
%    y2(i) = y1(i) - mu*(y1(i)-y1(i-1) );
   for i=2:m
%    y2(i) = y1(i) - mu*(y1(i+1)-y1(i-1))/2;
     y2(i) = 0.5*(y1(i+1)+y1(i-1)) - mu*(y1(i+1)-y1(i-1))/2;
   end
   i = m+1;
   y2(i) = y1(i) - mu*(y1(i)-y1(i-1) );

   t = t + k;
   y1 = y2;

   plot(x,y2); pause(0.5)

 end

 plot(x,y2,'o', "markersize", 3)

 u_e = zeros(m+1,1);
 for i=1:m+1
   u_e(i) = uexact(t,x(i));
 end

 max(abs(u_e-y2))

 plot(x,y2,':',x,u_e)

figure(2); plot(x,u_e-y2)
```






Below is a plot of the initial and consecutive approximation of the upwinding method for the same advection equation. The time step is Δt = 1.5h and the scheme is unstable, which leads to a blowing-up quantity.

In [ ]:

```
clear; close all

 a = 0;  b=1; tfinal = 0.5; % Input the domain and final time.

 m = 20; h = (b-a)/m; k = h; mu = 1.5*k/h; % Set mesh and time step.

 t = 0; n = fix(tfinal/k);  % Find the number of time steps.
 y1 = zeros(m+1,1); y2=y1; x=y1;

 figure(1);  hold
 %axis([-0.1 1.1 -0.1 1.1]);

 for i=1:m+1,
   x(i) = a + (i-1)*h;
   y1(i) = uexact(t,x(i));   % Initial data
   y2(i) = 0;
 end

 % Time marching

 for j=1:n,
   y1(1)=bc(t); y2(1)=bc(t+k);
   for i=2:m+1
     y2(i) = y1(i) - mu*(y1(i)-y1(i-1) );
   end
   t = t + k;
   y1 = y2;
   plot(x,y2); pause(0.5);  % Add the solution plot to the history.
 end

 u_e = zeros(m+1,1);
 for i=1:m+1
   u_e(i) = uexact(t,x(i));
 end

 max(abs(u_e-y2))

 plot(x,y2,'o',"markersize", 3,x,u_e )
 title('Upwind Unstable');
xlabel('x = time','fontsize',10)
ylabel('y = u_e','fontsize',10)
```