《计算流体力学基础》第三次作业

姓名：杨阳
学号：150*******
专业：流体力学

采用不同的数值格式计算如下一阶波动方程的解：

$$\frac{\partial u}{\partial t}+\frac{\partial u}{\partial x}=0$$

初始条件为：

$$u(x,0)=\sin(2\pi x)$$

计算区域取为$0<x<3$，采用周期边界条件：

1. 验证格式的稳定性条件，观察到发散现象；

2. 验证格式的精度阶数；

3. 观察数值解的耗散以及相位的超前和滞后。

问题的求解：

首先这个问题存在一个解析解，可以验证这个解析解的形式是：

$$u(x,t)=\sin\left[2\pi(x-t)\right]$$

这个解满处初始条件和边界条件。

差分格式列表

格式名称	格式表达式	格式精度
一阶迎风	$\frac{u_j^{n+1}-u_j^n}{\Delta t}=\frac{u_j^n-u_{j-1}^n}{\Delta x}$	一阶
隐式 FTCS	$\frac{u_j^{n+1}-u_j^n}{\Delta t}=\frac{u_{j+1}^{n+1}-u_{j-1}^{n+1}}{2\Delta x}$	一阶
Lax	$\frac{u_j^{n+1}-\frac{1}{2}\left(u_{j+1}^n+u_{j-1}^n\right)}{\Delta t}=\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}$	一阶
Lax-Wendroff	$\frac{u_j^{n+1}-u_j^n}{\Delta t}+\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}=\frac{1}{2}\Delta t\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{\Delta x^2}$	二阶
Leap-frog	$\frac{u_j^{n+1}-u_j^{n-1}}{2\Delta t}=\frac{u_{j+1}^n-u_{j-1}^n}{2\Delta x}$	二阶

为了方便绘图展示结果，并且计算量并不大，所以采用 MATLAB^R 软件完成程

序设计的计算。

1. 验证格式的稳定性条件，观察到发散现象；

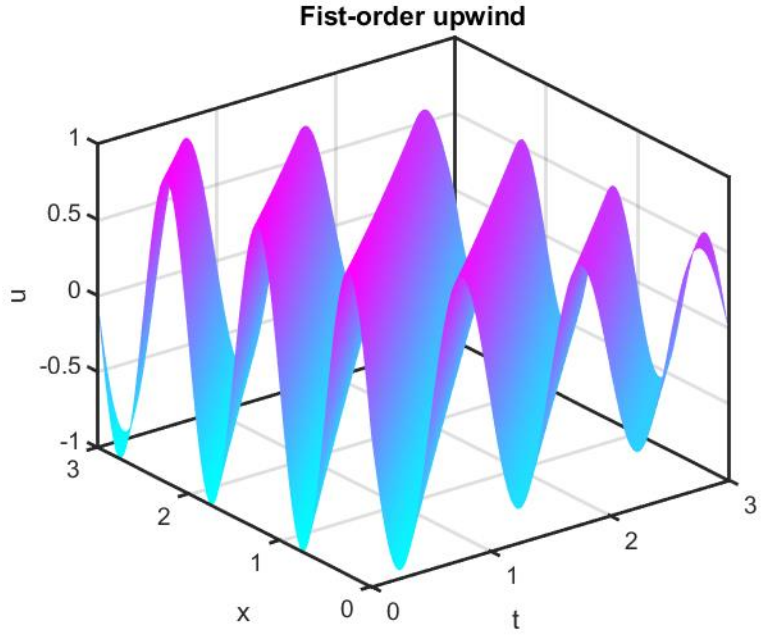
在所选择的显式差分格式中稳定性条件都是$c<0$。而隐式格式都是无条件稳定的，但是为了能够取得较好的时间上的分辨率对时间步长的选择仍然存在一定的限制。其中的c定义为：

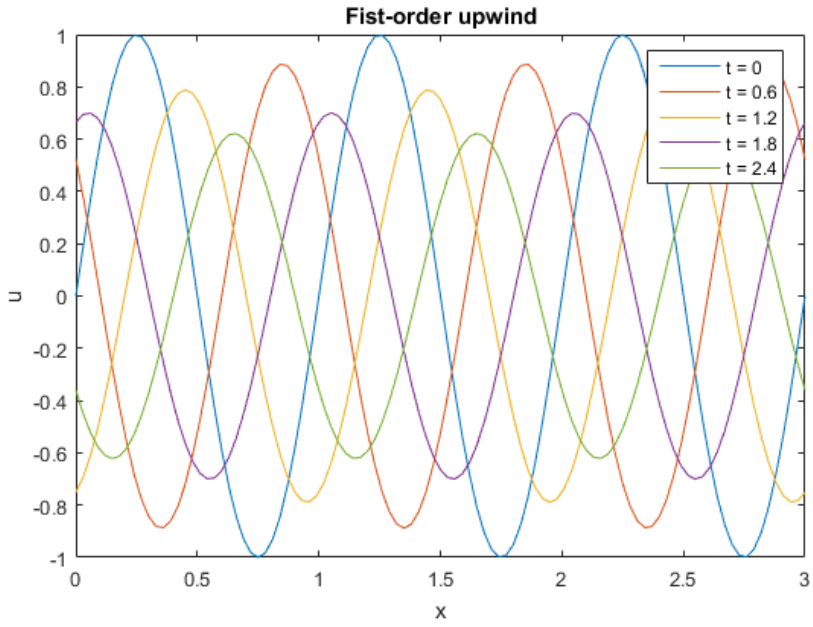
$$c=\frac{\Delta t}{\Delta x}$$

（1）一阶迎风格式

稳定情况$c=0.666667$

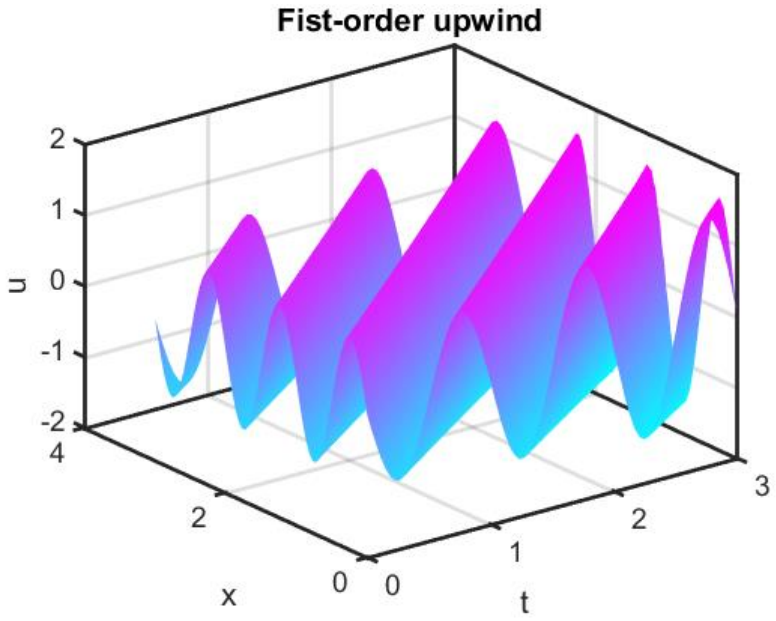
计算结果分别以曲面和曲线的形式展示：





从结果中可以看出这种格式是一种具有耗散性的格式，随着时间的推进解得幅值逐渐减小。

不稳定情况 $c=1.250000$：

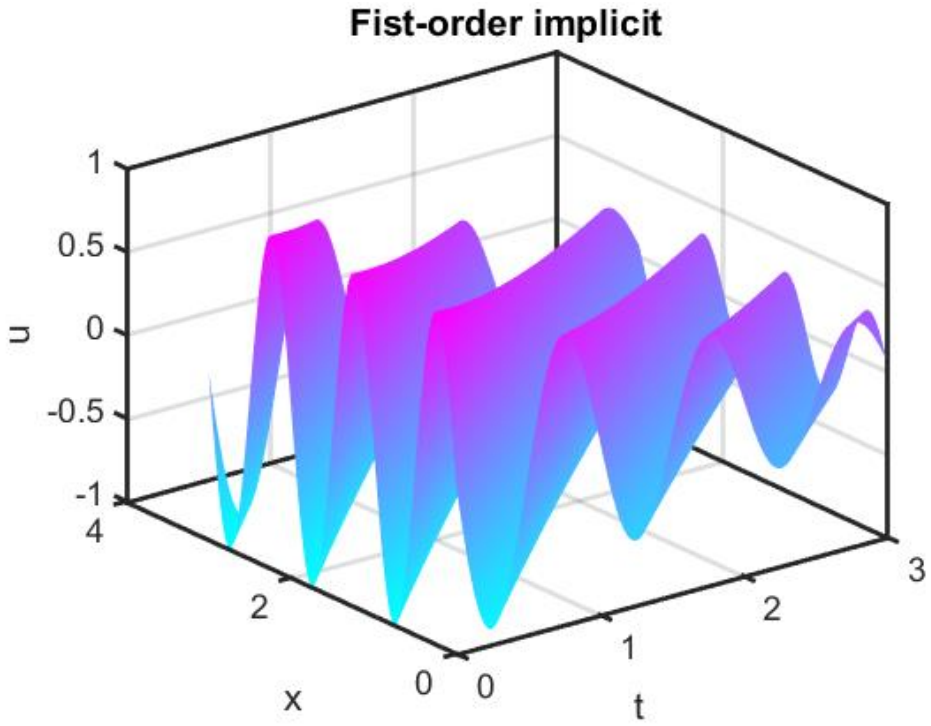


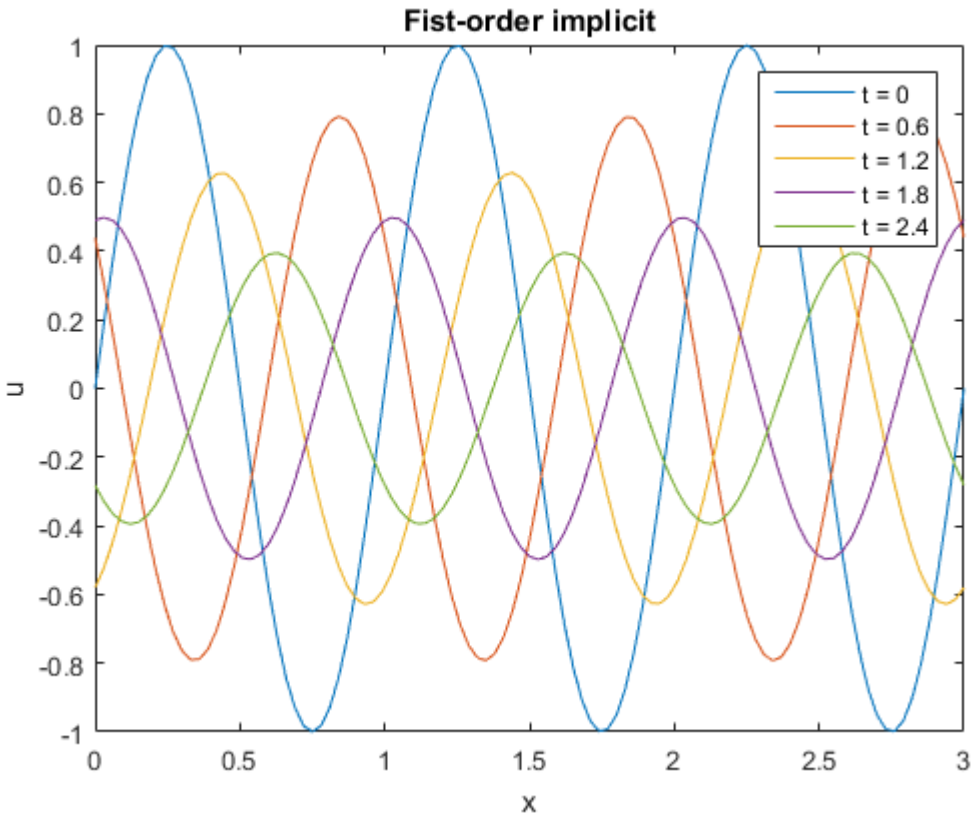
不稳定情况下解的幅值随着时间的推进不断增加，随着时间的增加因为解是不断

的增长的所以一定会发散。

（2）隐式 FTCS

稳定情况 $c = 0.666667$

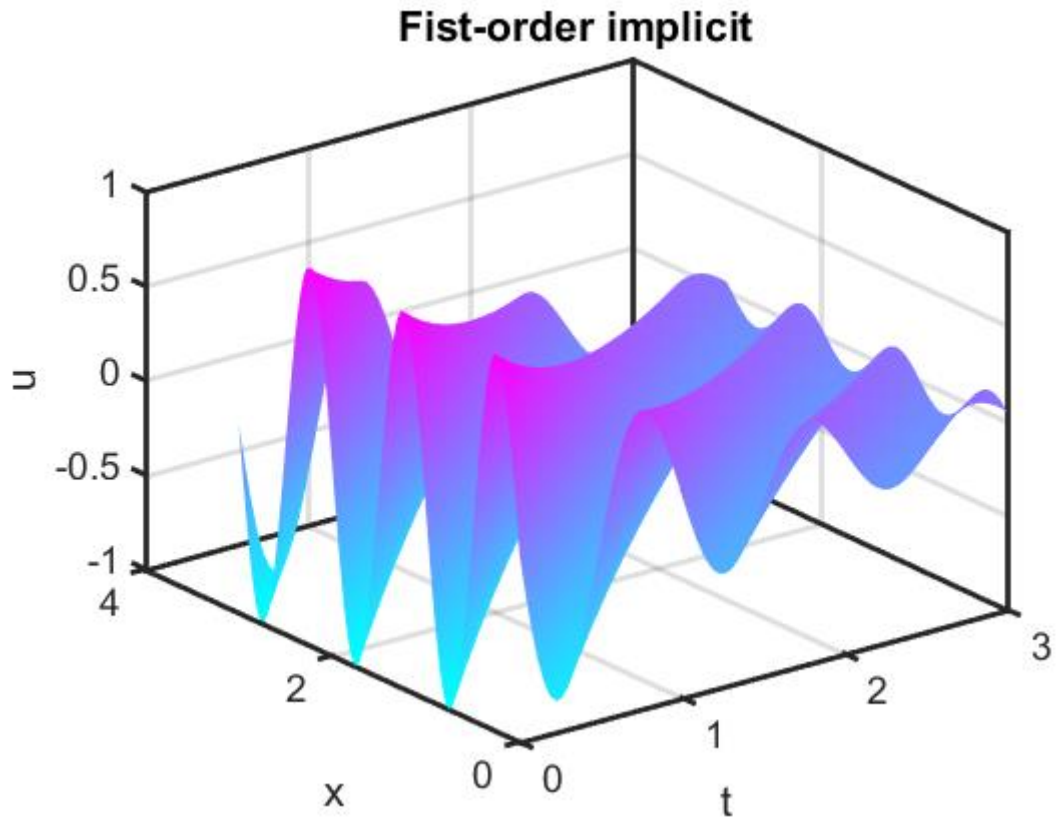


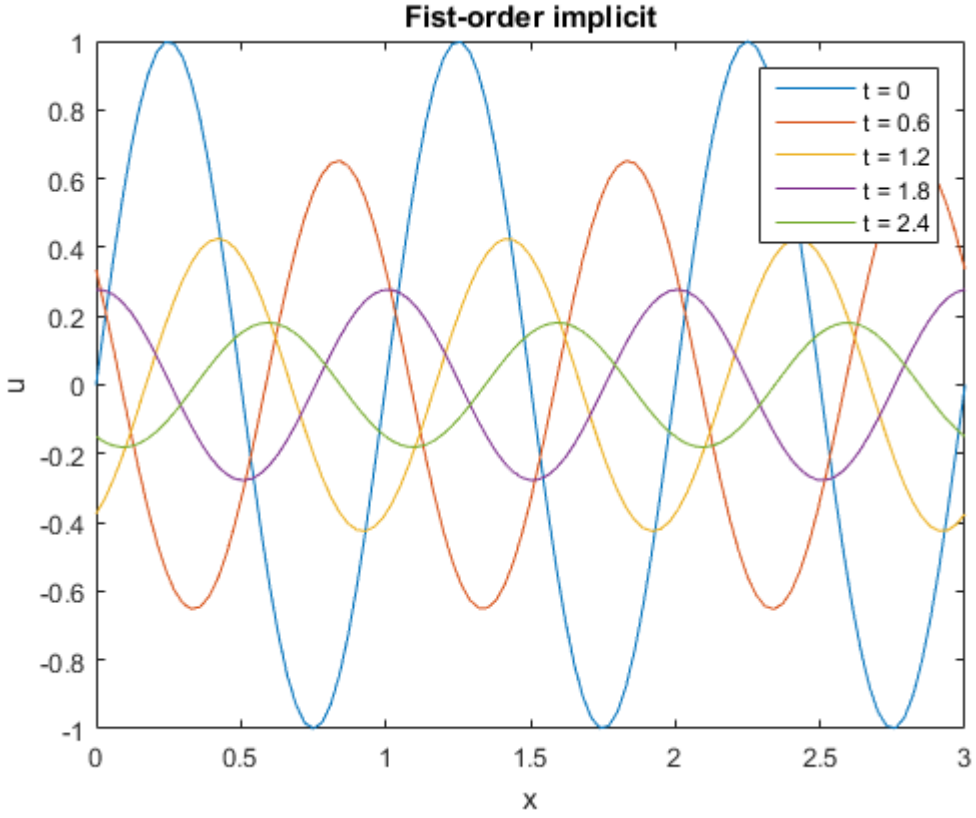


隐式格式是无条件稳定的，因为数值依赖域始终包含了物理依赖域，从结果中看

到隐式 FTCS 格式也是一种具有耗散性的格式，并且数值耗散还比较严重。

稳定情况 $c=1.250000$

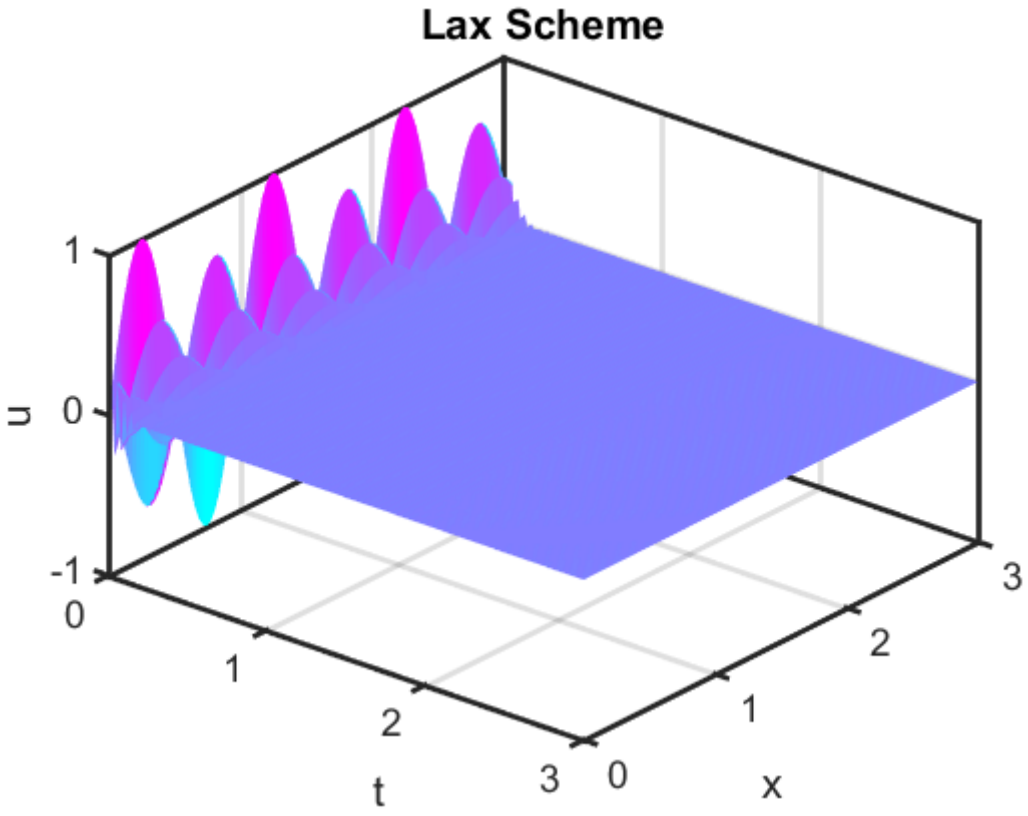




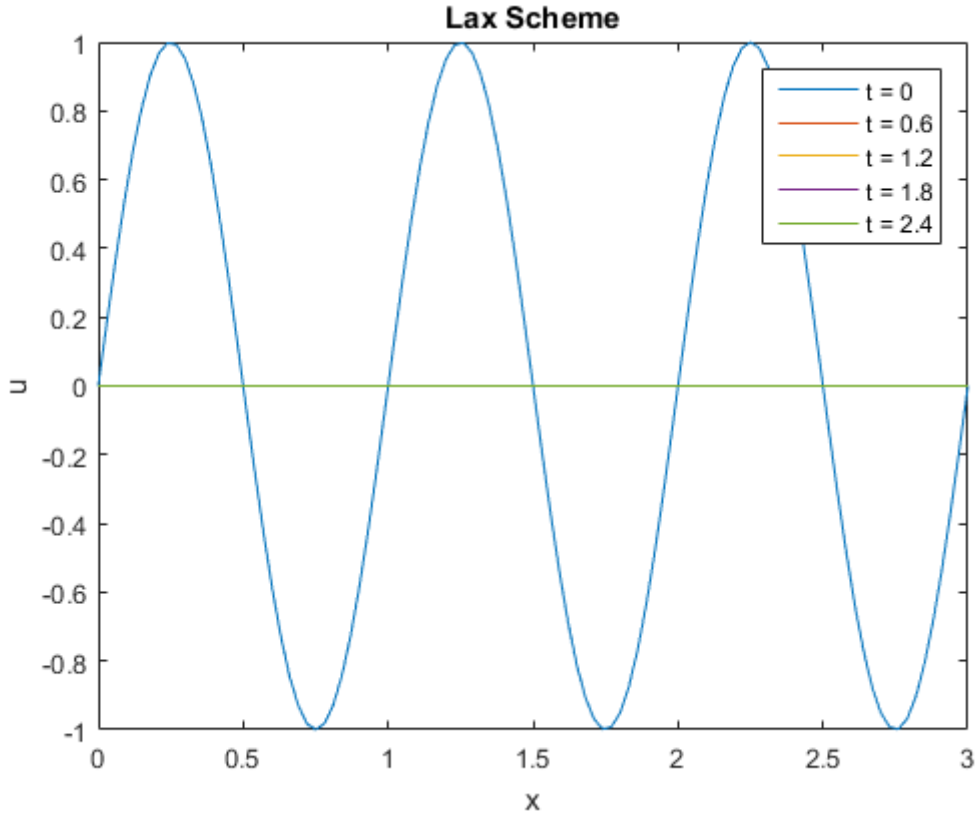
对于其他显式格式，当我们取 c>1 时就会出现发散，但是对于隐式格式 c 如何去取都不会导致结果发散，但是在这里我们看到 c 的值会影响到格式的数值粘性，c 越大格式的数值粘性越大。当我们把 c 增大后显然解随时间增加幅值衰减的更快了。这个结果可以直接从差分方程的修正方程中看出来，因为误差项中的二阶项系数为 c，若不改变空间步长显然增加 c 将带来格式数值粘性的增加。c 的增加也意味着时间步长的增加，因此虽然隐式格式是无条件稳定的但是时间步长仍不能取得过大。

（3）Lax 格式

稳定情况 $c=0.666667$

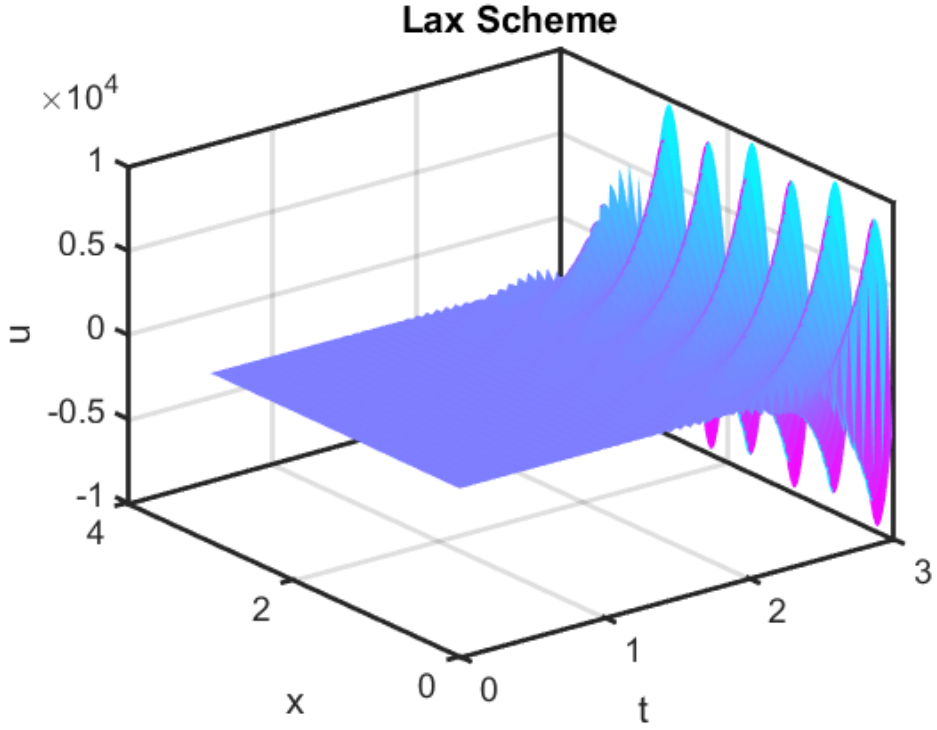


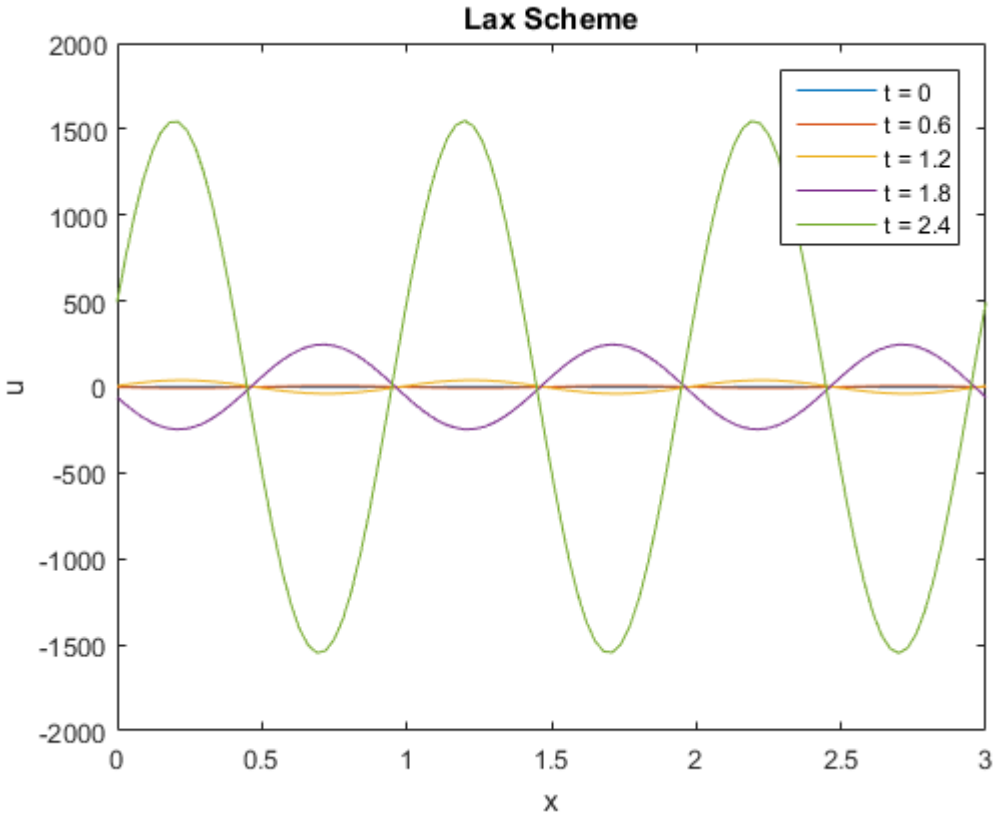
从时间方向看这里的解是稳定的。



Lax 格式具有很大的数值粘性，随着时间步的推进很快解就衰减了，在这里计算中解得衰减甚至是数量级的衰减，所以很快就看不到解了。

不稳定情况 $c=1.111111$

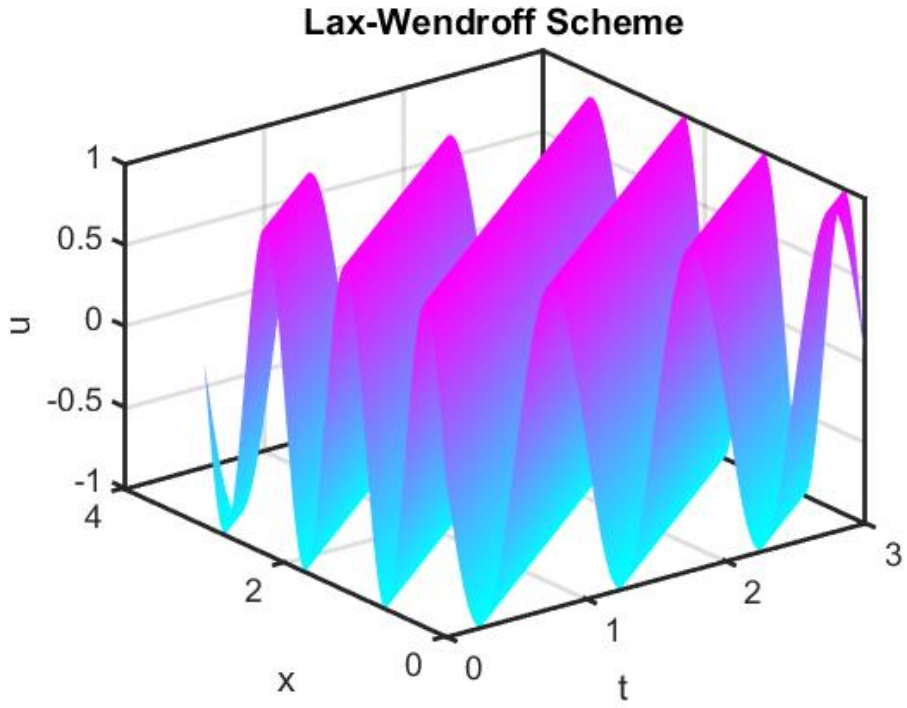


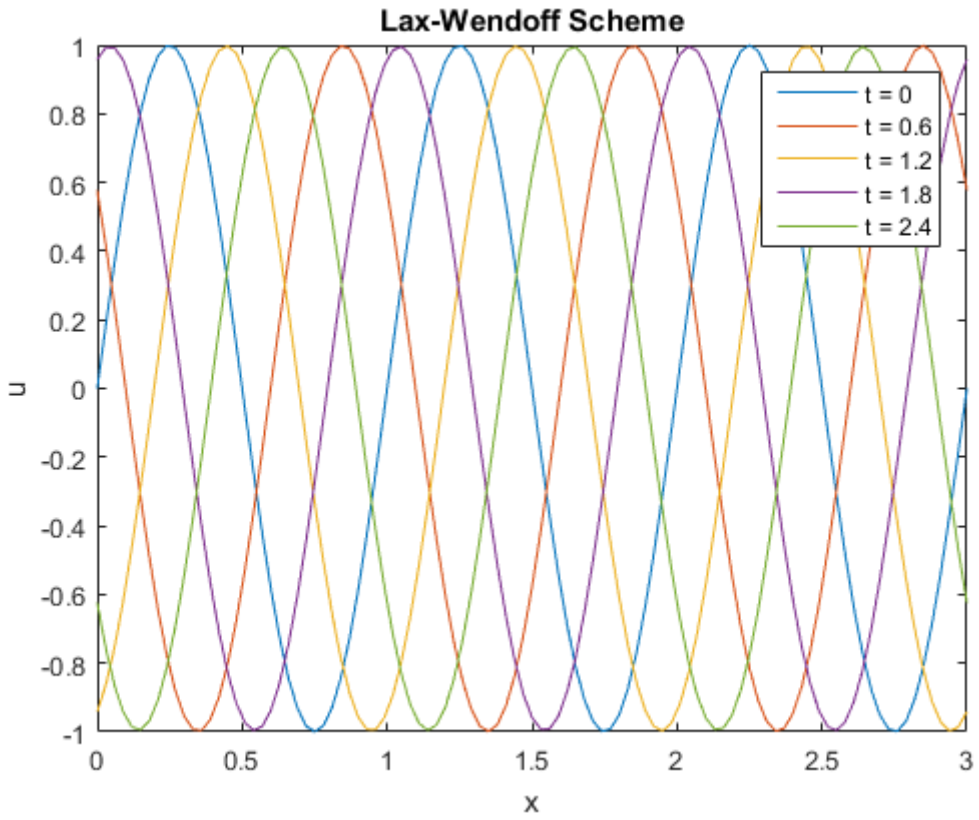


当我们将 c 取为大于 1 的数的时候，格式是不稳定的，解得幅值迅速的增加，最后发散。

（4）Lax-Wendroff 格式

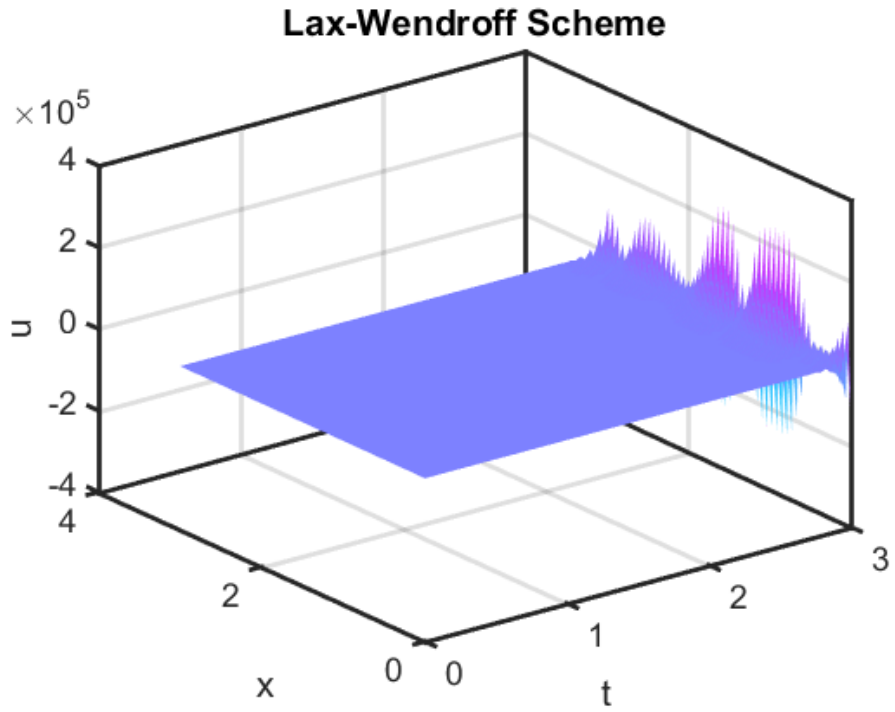
稳定情况 $c = 0.666667$

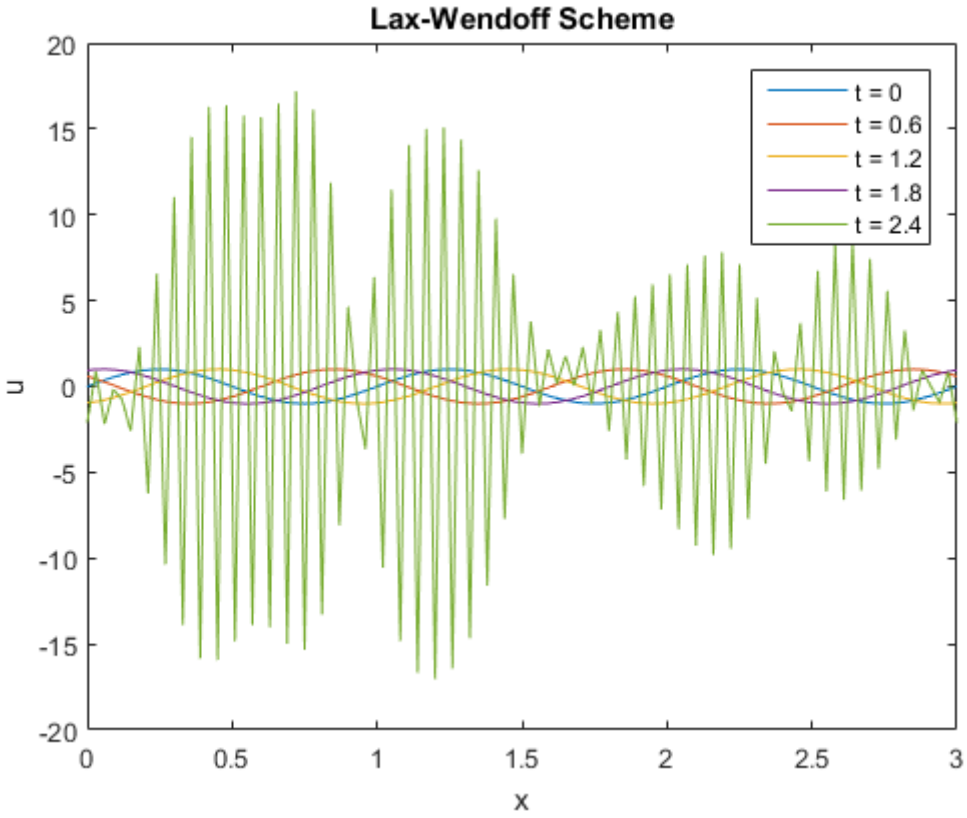




从解得结果中来看，Lax-Wendroff 格式能够较好的保持解的幅值，因为这个格式具有二阶精度。当 c 是小于 1 的值时格式稳定。

不稳定情况 $c=1.176471$

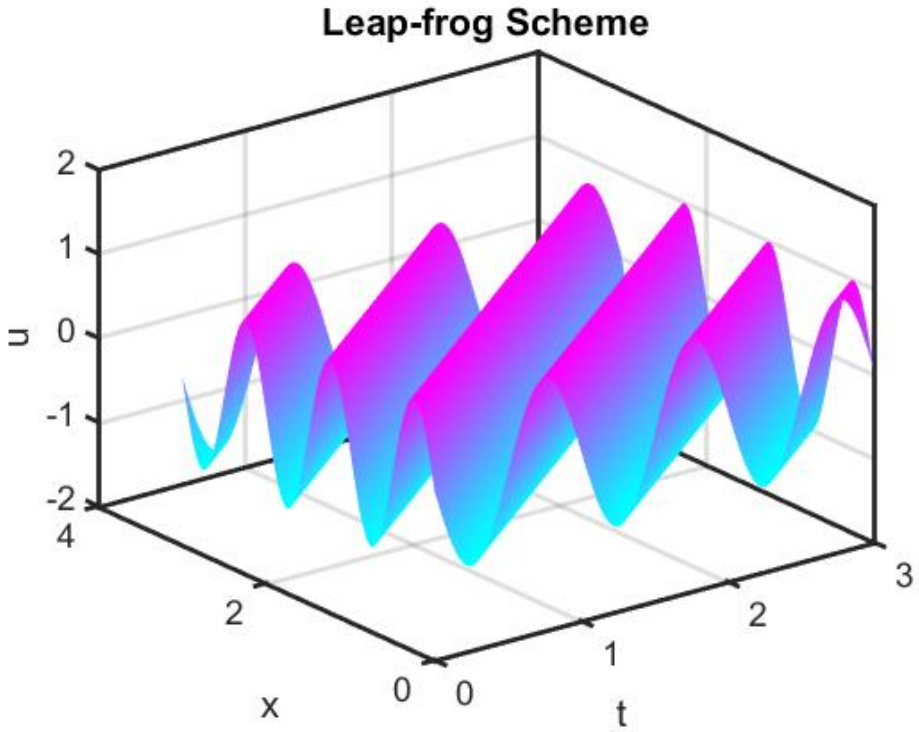


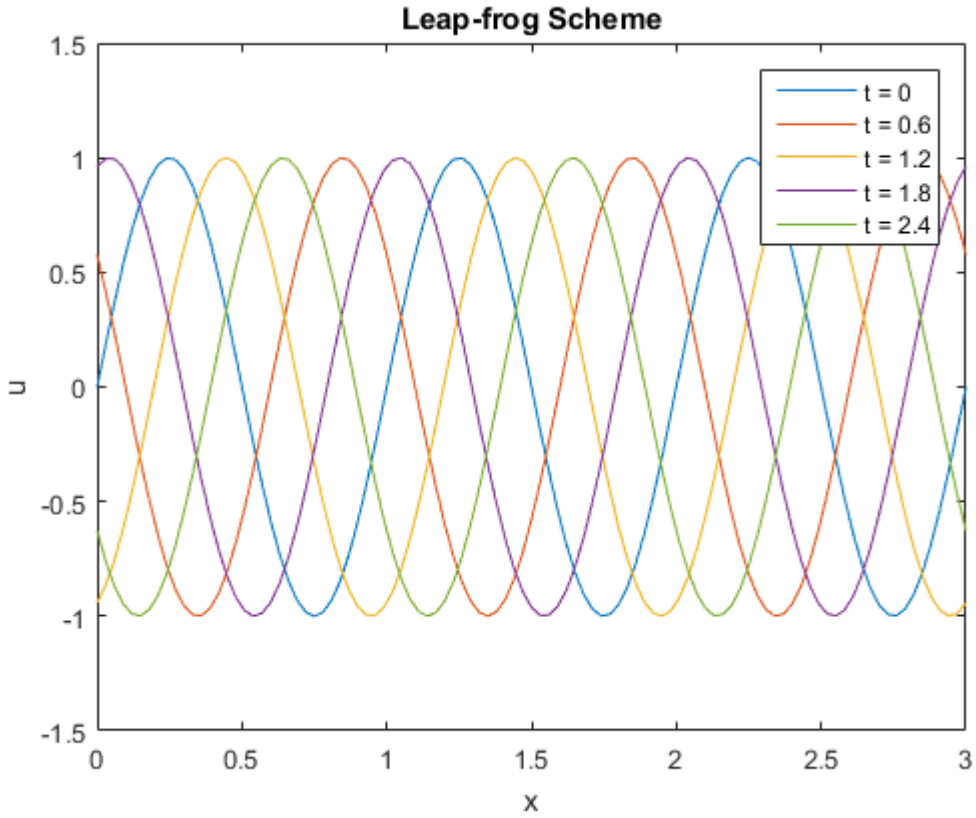


当 c 大于 1 时 Lax-Wendroff 格式得到的解得振幅会逐渐增加最后快速发散，并且看到这个发散的解具有高次谐波。

（5）Leap-frog 格式

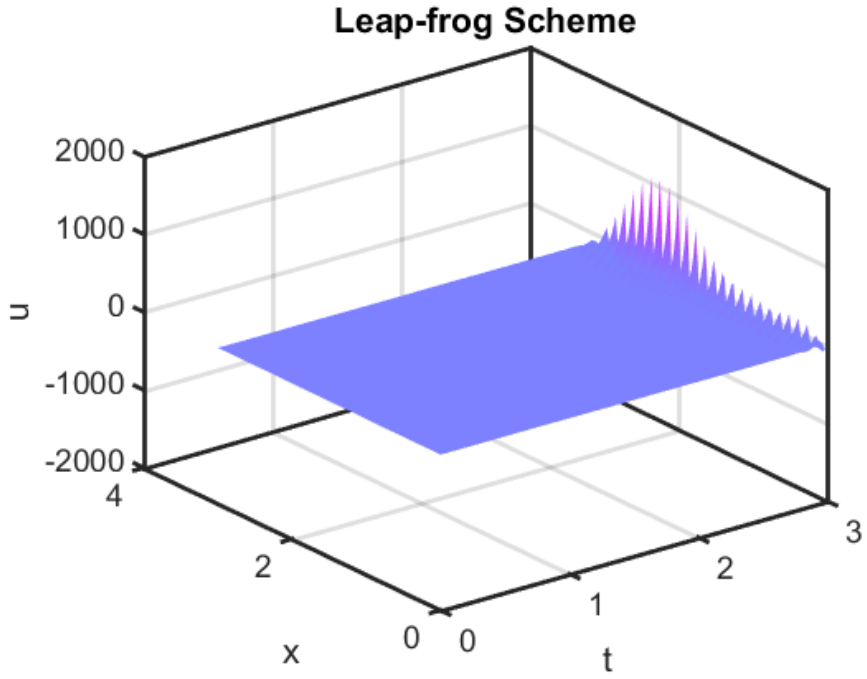
稳定情况 $c = 0.666667$

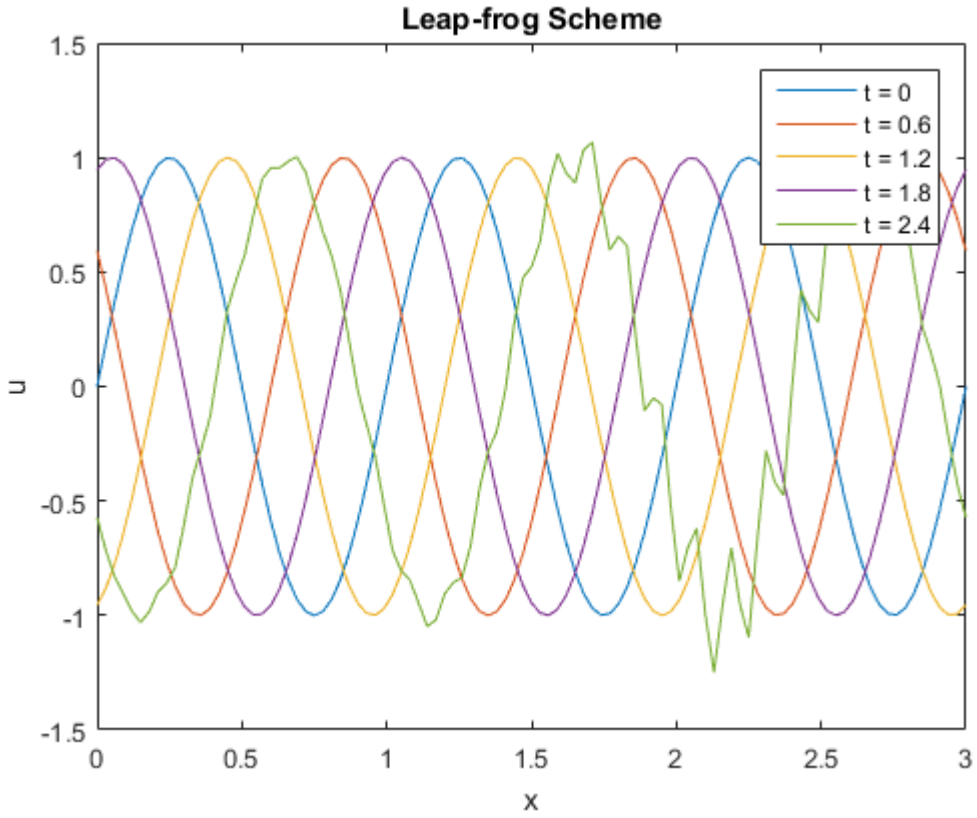




对于蛙跳格式而言赋予恰当的初值是很重要的，在这里我们有精确解而实际情况往往不好赋予两个时间层上的初值，不过可以使用其他方法先去初步计算再使用蛙跳格式。如果初值赋予的不合理奇偶时间层上的解会有差距然后逐渐抹平差距。蛙跳格式能够保持解的幅值，而事实上蛙跳格式就是没有耗散的具有二阶精度的格式。

不稳定情况 $c=1.111111$





当 c 大于 1 时蛙跳格式是不稳定的，解中会出现高次谐波并且幅值被放大。

2. 验证格式的精度阶数；

在这个问题中我们知道精确解的形式，同时可以在数值计算中固定住 Courant 数 c，然后改变空间网格步长并且在时间步上只推进几步后和精确解比对的方式来事后估计格式的精度，以验证格式的精度。

根据数值格式精度的事后估计方法有：

$$\|u_e - u_h\| = Ch^p \qquad \|u_e - u_{2h}\| = C(2h)^p = 2^p Ch^p$$

$$\log_2\left(\frac{\|u_e - u_{2h}\|}{\|u_e - u_h\|}\right) = \log_2\left(2^p\right) = p$$

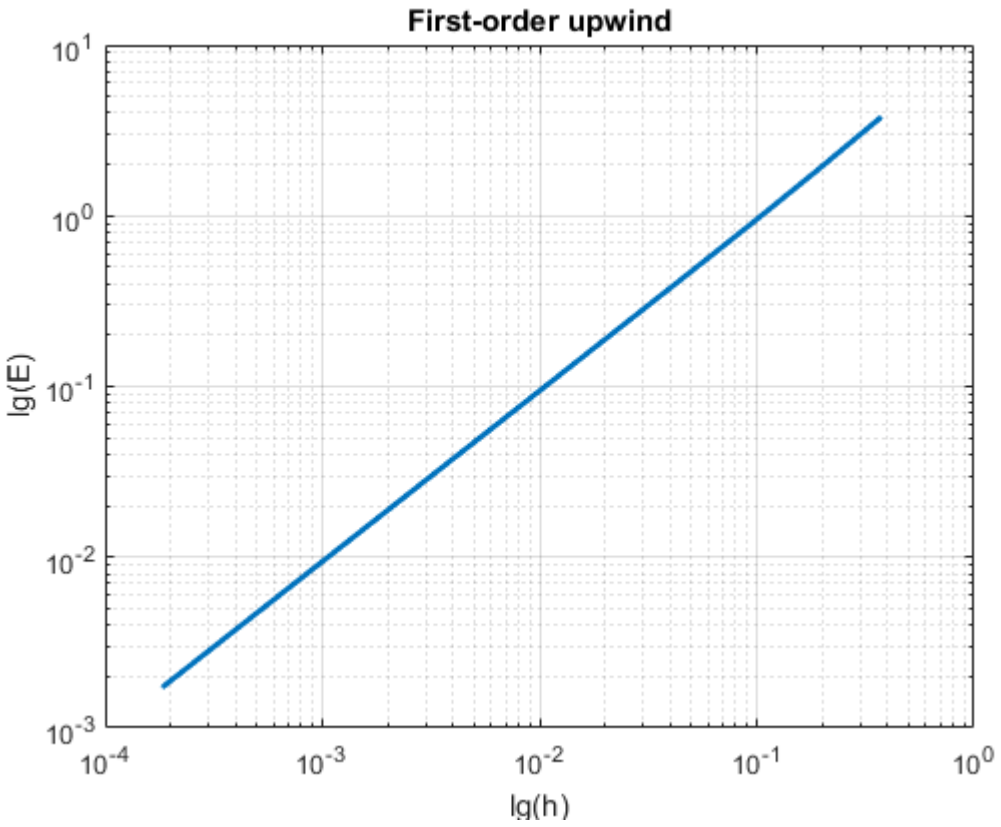
用这样的方法就能估计出数值格式精度 P，但是由于舍入误差的影响实际上 P 并不是精确的，但可以用回归分析去估计。

如果我们令

$$E_i = \sum_i \|u_e - u_{h_i}\|_1 / N$$

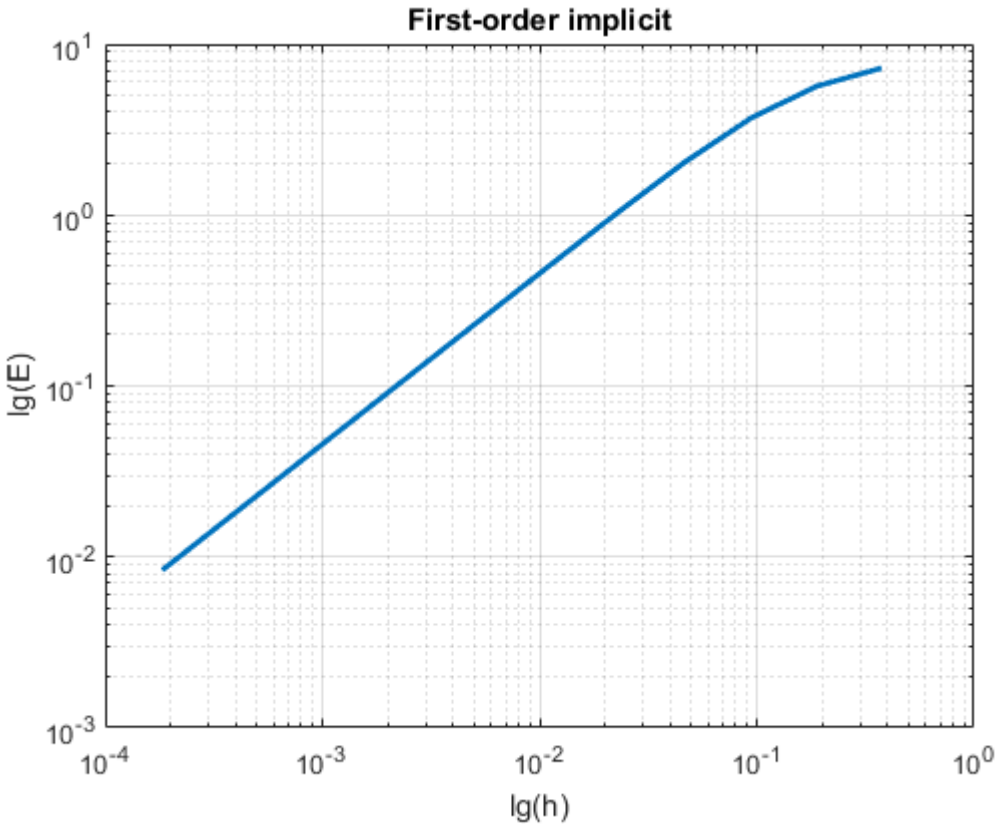
那么通过做出 $\lg(h_i)-\lg(E_i)$ 图并分析曲线大概的斜率就可以知道格式精度的阶数 P。

（1）一阶迎风格式
计算结果绘图如下，采用双对数坐标图。



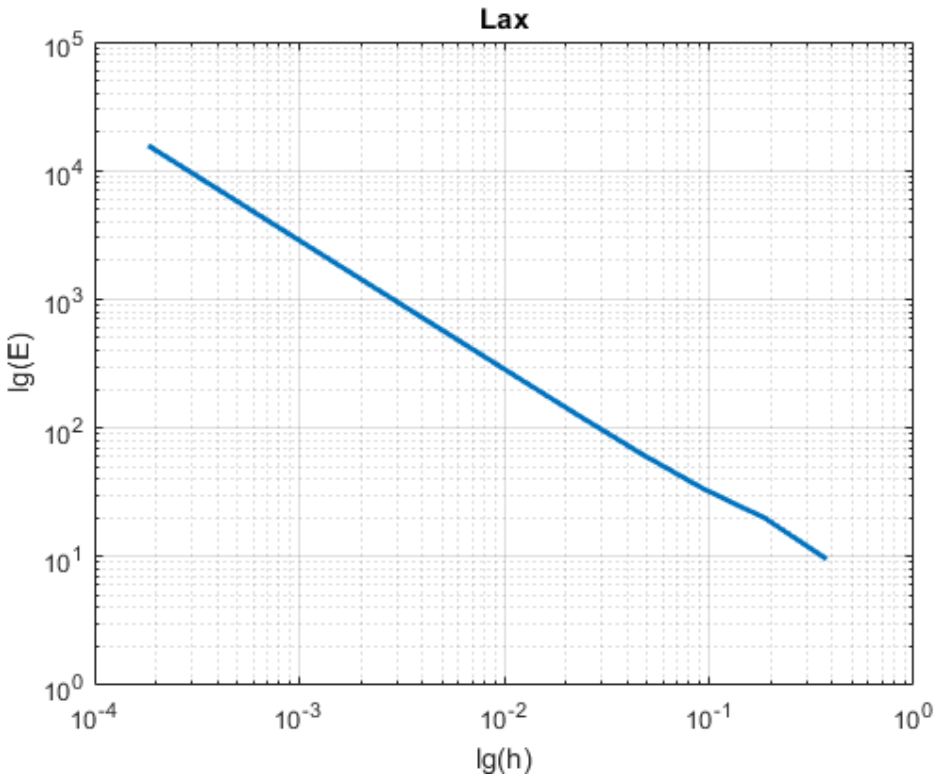
在双对数坐标图中可以很清楚的看到曲线的斜率是 1，所以一阶迎风差分格式的精度是一阶的，这与预期符合的较好。

（2）隐式 FTCS 格式



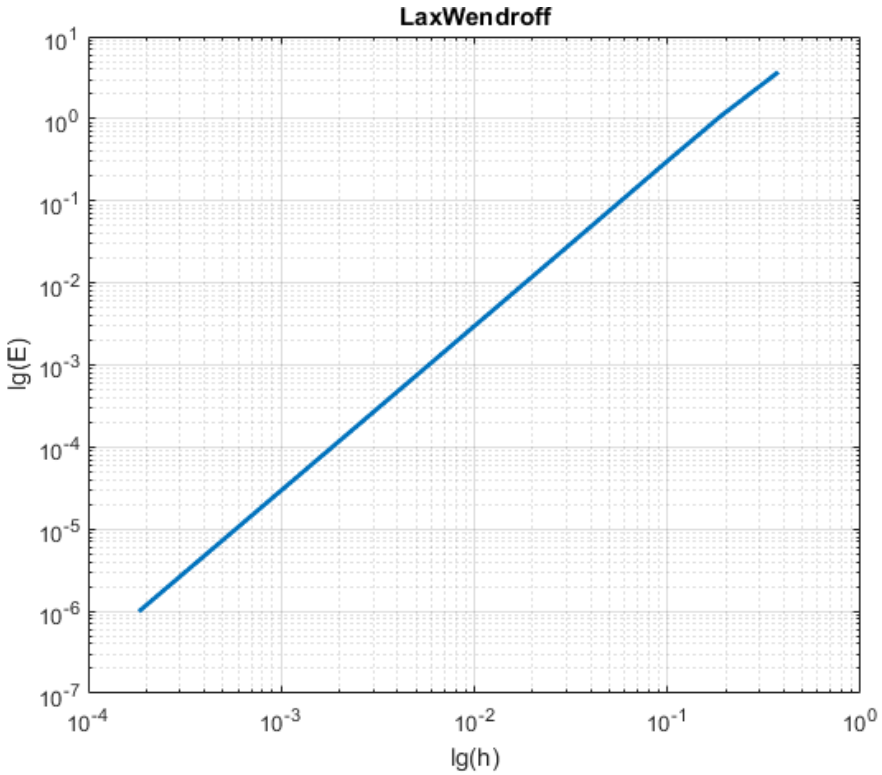
从双对数坐标途中也可以大致看出曲线的斜率是 1，所以隐式 FTCS 格式的精度也是 1 阶的和预期的结果是一致的。

（3）Lax 格式



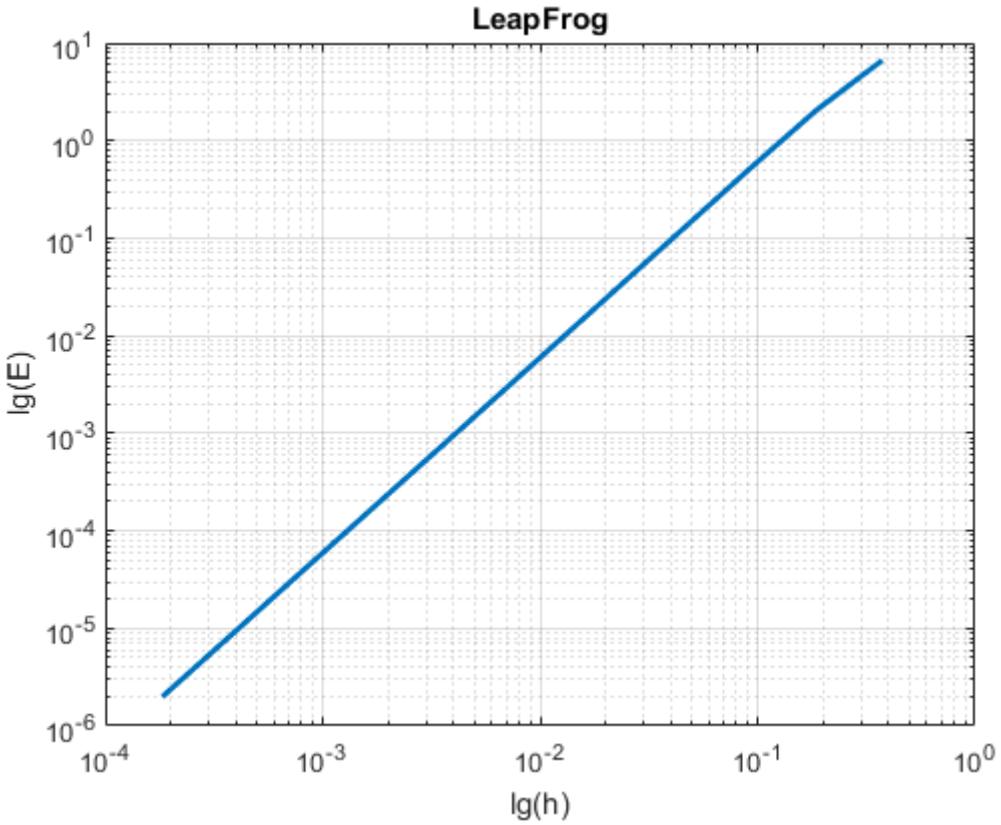
Lax 格式是具有一阶精度的格式。

（4）Lax-Wendroff 格式



从双对数坐标图中可以看出，曲线的斜率为 2 因为横坐标跨一格的时候纵坐标垮了两格，所以 Lax-Wendroff 格式是一种具有二阶精度的格式。

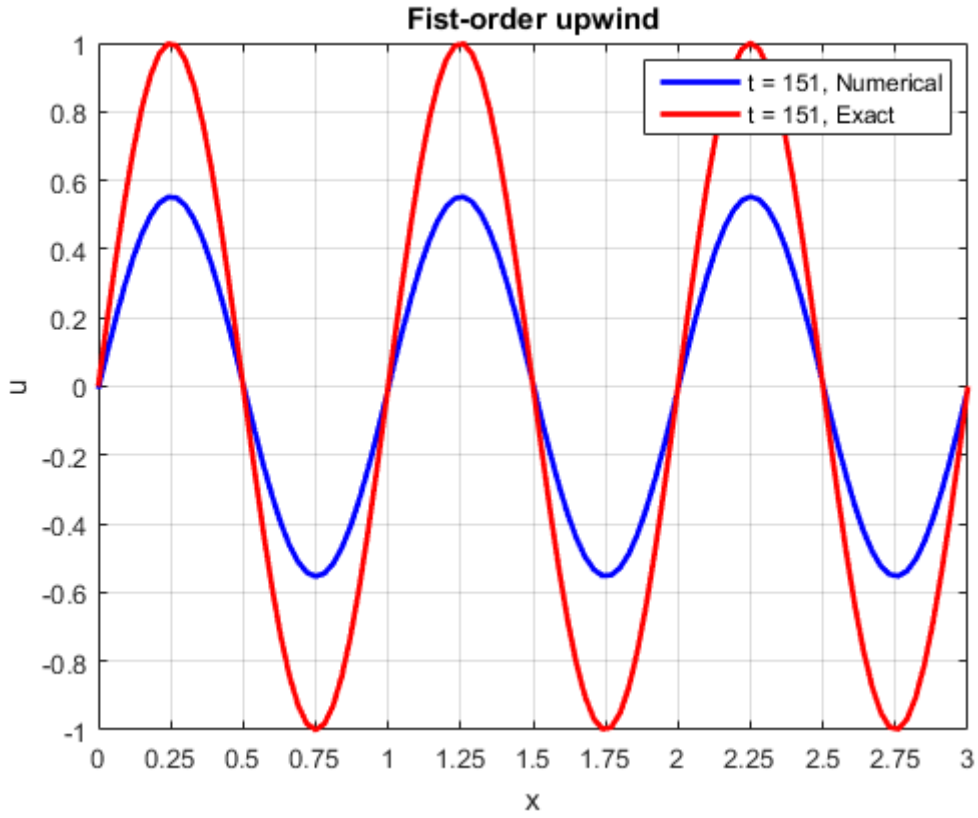
（5）Leap-frog 格式



从双对数坐标途中看到曲线的斜率是 2，所以蛙跳格式也是一种具有二阶精度的格式，验证了理论预期。

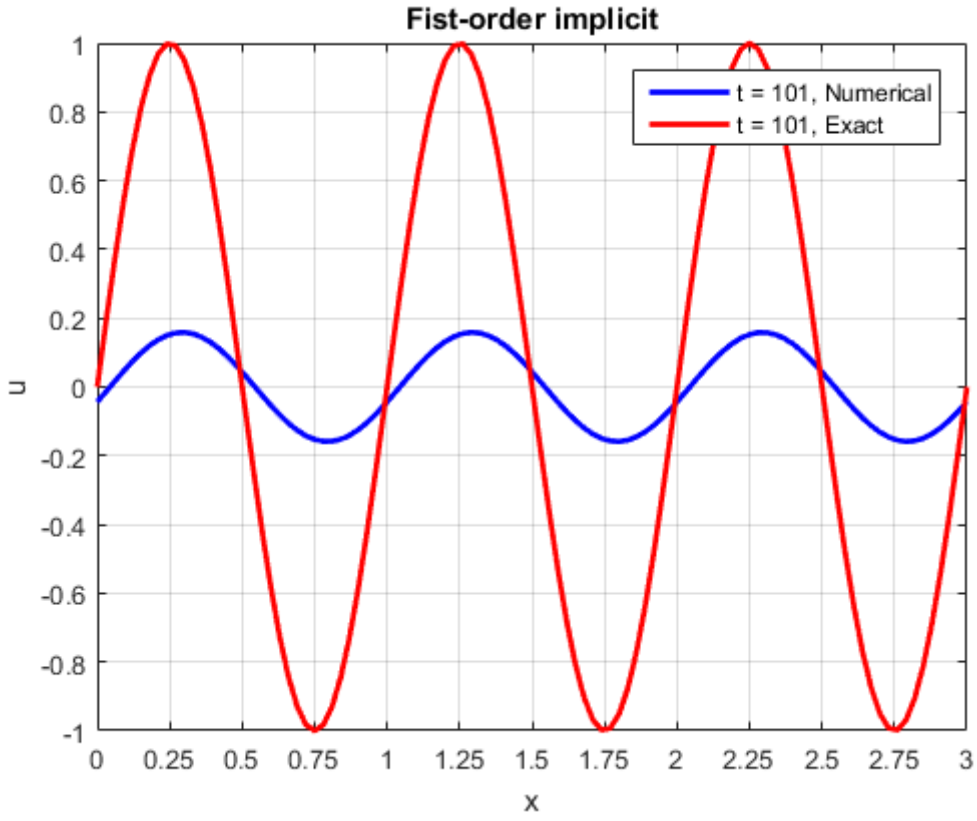
3.观察数值解的耗散以及相位的超前和滞后。

（1）一阶迎风格式



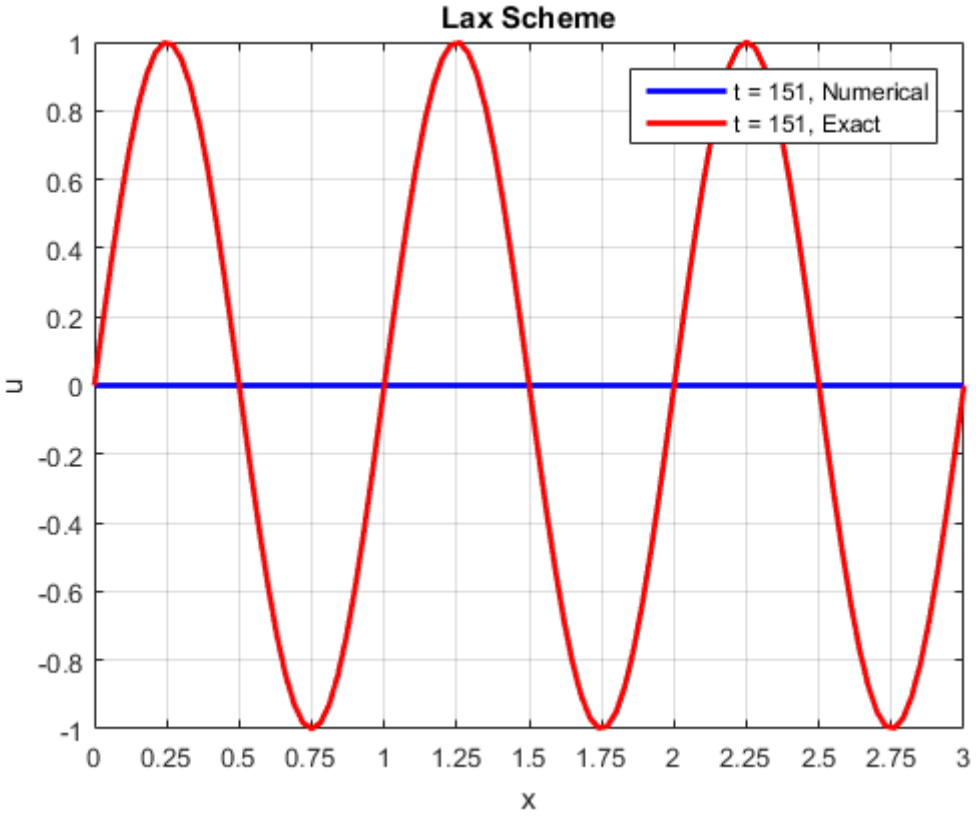
在 151 时间步的推进后对比精确解和数值解可以看到一阶迎风格式是一种耗散为主的格式，没有观察到明显的色散，理论解与数值解的相位基本一致。而数值解得振幅较理论界衰减的明显。

（2）隐式 FTCS



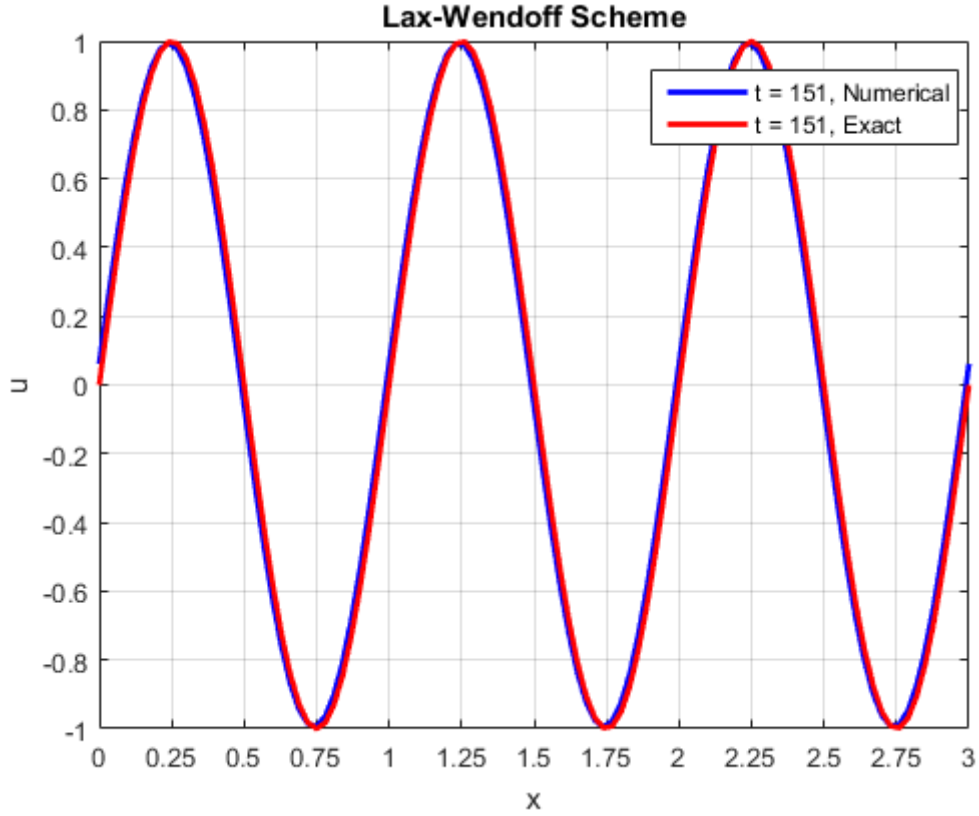
从结果中看到，数值解的振幅较理论解有很明显的衰减，并且数值解的相位也与理论解不同步，数值解的相位较理论解超前。所以隐式 FTCS 格式是一种既有耗散又有色散的格式而且还比较严重，所以这个格式不适合实际计算采用。

（3）Lax 格式



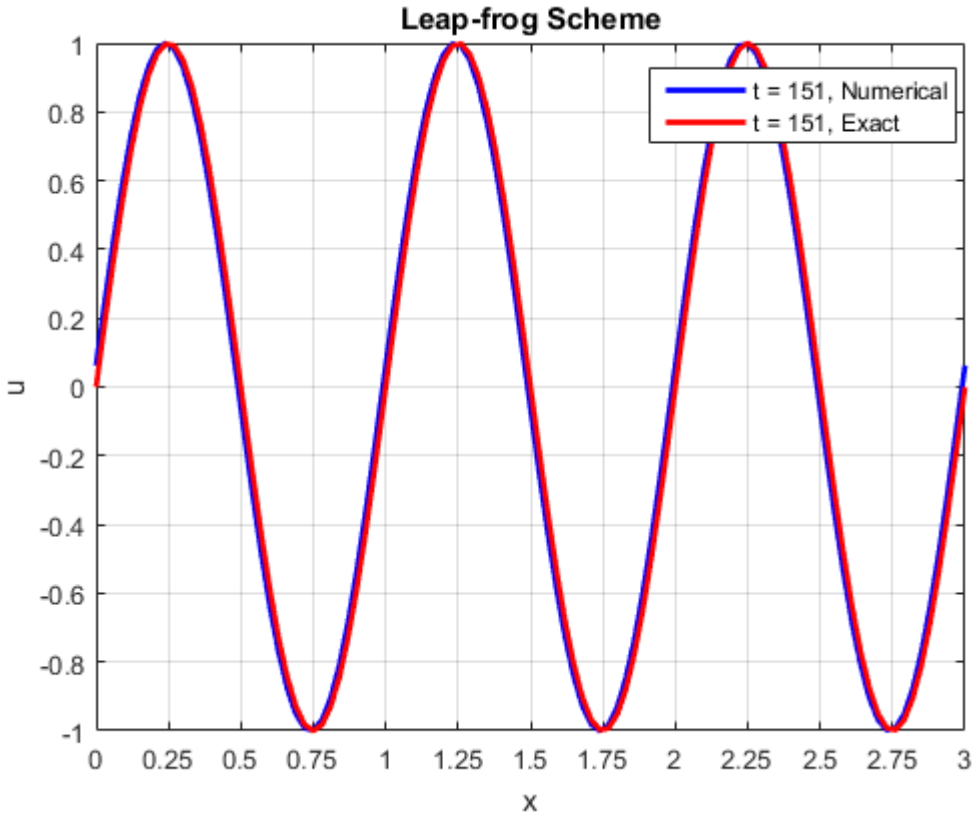
从结果中看出 Lax 格式具有极强的数值粘性，数值解相对于理论解振幅上有数量级的差距，所以 Lax 格式也是一中不可以实际计算采用的格式。

（4）Lax-Wendroff 格式



从结果中看出，理论解与数值解的振幅还是比较接近，但是数值解的相位略微滞后于精确解，属于是一种慢格式。Lax-Wendroff 格式是一种二阶精度的格式所以精确度相比于一阶格式有较大的提高。

（5）Leap-frog 格式



计算的结果验证了蛙跳格式是一种没有耗散的格式，数值解的振幅于理论解的振幅一致，但是蛙跳格式数值解略微在相位上落后于精确解，验证了蛙跳格式是一种色散型格式，并且因为相位滞后于精确解所以蛙跳格式属于是慢格式。

程序附录

1. 第一问程序

OneDimensionWaveEquationProblem1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
%%%% Program name: OneDimensionWave
Equation
%%%% Program purpose: Compute and Analysis
different numerical scheme
%%%% Aurthor: Yang Yang
%%%% Date: 2015.11.05
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%

%% Fist-order upwind
%\frac{u_{j}^{n+1}-u_{j}^{n}}{\Delta
t}=\frac{u_{j}^{n}-u_{j-1}^{n}}{\Delta x}
% basic parameters
xspan = [0 3];
```

```
tspan = [0 3];

Nx = 100;

Nt = 150;

plots = 5;

% initial variables

delta_x = (xspan(2) - xspan(1))/Nx;

delta_t = (tspan(2) - tspan(1))/Nt;

c = delta_t / delta_x;

fprintf('First-order upwind: c = %f\n',c);

x = zeros(1,Nx+1);

t = zeros(1,Nt+1);

u = zeros(Nt+1,Nx+1);

% initial mesh nodes

x = xspan(1):delta_x:xspan(2);   % x(i)

spatial nodes

t = tspan(1):delta_t:tspan(2);   % t(i)

time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)

initial u using initial condition
```

```
% forward in time step
for n = 2:Nt+1
    for i = 2:Nx
        u(n,i) = u(n-1,i) - c*( u(n-1,i) -
u(n-1,i-1) );        % inner points
    end
    u(n,1) = u(n-1,1) - c*( u(n-1,1) -
u(n-1,Nx) );             % boundary points
    u(n,Nx+1) = u(n-1,Nx+1) - c*( u(n-
1,Nx+1) - u(n-1,Nx) );    % boundary
points
end

% plot results
% surf
f1 = figure('Color',[1 1 1]);
axes1 = axes('Parent',f1);
colormap('cool');
[X,T] = meshgrid(x,t);
surf(T,X,u,'Parent',axes1,'EdgeAlpha',0.8,
'EdgeColor','none');
box on;
```

```
grid on;
xlabel('t');
ylabel('x');
zlabel('u');
set(axes1,'FontSize',14,'LineWidth',2);
title('Fist-order upwind');

% curves
figure('Color',[1 1 1]);


Nt_temp = 1;
name_cell = cell(1,plots);
for i = 1:plots
    delta_Nt = floor((Nt+1) / plots);
    plot(x,u(Nt_temp,:));
    hold on
    name_cell{i} = ['t =
',num2str(t(Nt_temp))];
    Nt_temp = Nt_temp + delta_Nt;
end
hold off
legend(name_cell);
```

```
xlabel('x');
ylabel('u');
title('Fist-order upwind')

%% Implicit Scheme
% \frac{u_{j}^{n+1}-u_{j}^{n}}{\Delta
t}=\frac{u_{j+1}^{n+1}-u_{j-
1}^{n+1}}{2\Delta x}

% basic parameters
xspan = [0 3];
tspan = [0 3];
Nx = 100;
Nt = 150;
plots = 5;

% initial variables
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = (tspan(2) - tspan(1))/Nt;
c = delta_t / delta_x;

fprintf('Implicit Scheme: c = %f\n',c);
x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
```

```
u = zeros(Nt+1,Nx+1);

% initial mesh nodes
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition

% assemble matrix
A = zeros(Nx+1,Nx+1);

A(1,Nx) = -1/2*c;
A(1,1) = 1;
A(1,2) = 1/2*c;
for i = 2:Nx
    A(i,i-1) = -1/2*c;
    A(i,i) = 1;
    A(i,i+1) = 1/2*c;
end
A(Nx+1,Nx) = -1/2*c;
A(Nx+1,Nx+1) = 1;
```

```
A(Nx+1,2) = 1/2*c;


% forward in time step
for n = 2:Nt+1
    u(n,:) = (A\(u(n-1,:)'))';    % All
points using Matrix Algebra
end

% plot results
% surf
f1 = figure('Color',[1 1 1]);
axes1 = axes('Parent',f1);
colormap('cool');
[X,T] = meshgrid(x,t);
surf(T,X,u,'Parent',axes1,'EdgeAlpha',0.8,
'EdgeColor','none');
box on;
grid on;
xlabel('t');
ylabel('x');
zlabel('u');
set(axes1,'FontSize',14,'LineWidth',2);
```

```
title('Fist-order implicit')

% curves
figure('Color',[1 1 1]);
colormap jet

Nt_temp = 1;
name_cell = cell(1,plots);
for i = 1:plots
    delta_Nt = floor((Nt+1) / plots);
    plot(x,u(Nt_temp,:));
    hold on
    name_cell{i} = ['t =
',num2str(t(Nt_temp))];
    Nt_temp = Nt_temp + delta_Nt;
end
hold off
legend(name_cell);
xlabel('x');
ylabel('u');
title('Fist-order implicit')

%% Lax Scheme
```

```
% \frac{u_{j}^{n+1}-
\frac{1}{2}\left(  u_{j+1}^{n}+u_{j-
1}^{n}\right)  }{\Delta
t}=\frac{u_{j+1}^{n}-u_{j-1}^{n}}{2\Delta
x}%
% basic parameters
xspan = [0 3];
tspan = [0 3];
Nx = 100;
Nt = 150;
plots = 5;

% initial variables
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = (tspan(2) - tspan(1))/Nt;
c = delta_t / delta_x;

fprintf('Lax Scheme: c = %f\n',c);
x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
u = zeros(Nt+1,Nx+1);

% initial mesh nodes
```

```
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition

% forward in time step
for n = 2:Nt+1
    for i = 2:Nx
        u(n,i) = 0.5*(1-c)*u(n-1,i+1) -
0.5*(1+c)*u(n-1,i-1);      % inner points
    end
    u(n,1) = 0.5*(1-c)*u(n-1,2) -
0.5*(1+c)*u(n-1,Nx);                %
boundary points
    u(n,Nx+1) = 0.5*(1-c)*u(n-1,2) -
0.5*(1+c)*u(n-1,Nx);       % boundary
points
end

% plot results
```

```
% surf
f1 = figure('Color',[1 1 1]);
axes1 = axes('Parent',f1);
colormap('cool');
[X,T] = meshgrid(x,t);
surf(T,X,u,'Parent',axes1,'EdgeAlpha',0.8,
'EdgeColor','none');
box on;
grid on;
xlabel('t');
ylabel('x');
zlabel('u');
set(axes1,'FontSize',14,'LineWidth',2);
title('Lax Scheme')

% curves
figure('Color',[1 1 1]);
colormap jet

Nt_temp = 1;
name_cell = cell(1,plots);
for i = 1:plots
    delta_Nt = floor((Nt+1) / plots);
```

```
    plot(x,u(Nt_temp,:));
    hold on
    name_cell{i} = ['t =
',num2str(t(Nt_temp))];
    Nt_temp = Nt_temp + delta_Nt;
end
hold off
legend(name_cell);
xlabel('x');
ylabel('u');
title('Lax Scheme')

%% Lax-Wendroff Scheme
% \frac{u_{j}^{n+1}-u_{j}^{n}}{\Delta
t}+\frac{u_{j+1}^{n}-u_{j-1}^{n}}{2\Delta
x}=\frac{1}{2}\Delta t\frac{u_{j+1}^{n}-
2u_{j}^{n}+u_{j-1}^{n}}{\Delta x^{2}}
% basic parameters
xspan = [0 3];
tspan = [0 3];
Nx = 100;
Nt = 150;
```

```
plots = 5;

% initial variables
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = (tspan(2) - tspan(1))/Nt;
c = delta_t / delta_x;

fprintf('Lax-Wendroff Scheme: c
= %f\n',c);
x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
u = zeros(Nt+1,Nx+1);

% initial mesh nodes
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition

% forward in time step
for n = 2:Nt+1
```

```
    for i = 2:Nx
        u(n,i) = (1-c^2)*u(n-1,i) +
0.5*c*(c-1)*u(n-1,i+1) + 0.5*c*(c+1)*u(n-
1,i-1);  % inner points
    end
    u(n,1) = (1-c^2)*u(n-1,1) + 0.5*c*(c-
1)*u(n-1,2) + 0.5*c*(c+1)*u(n-1,Nx);      %
boundary points
    u(n,Nx+1) = (1-c^2)*u(n-1,Nx+1) +
0.5*c*(c-1)*u(n-1,2) + 0.5*c*(c+1)*u(n-
1,Nx);      % boundary points
end

% plot results
% surf
f1 = figure('Color',[1 1 1]);
axes1 = axes('Parent',f1);
colormap('cool');
[X,T] = meshgrid(x,t);
surf(T,X,u,'Parent',axes1,'EdgeAlpha',0.8,
'EdgeColor','none');
box on;
```

```
grid on;
xlabel('t');
ylabel('x');
zlabel('u');
set(axes1,'FontSize',14,'LineWidth',2);
title('Lax-Wendroff Scheme');
% curves
figure('Color',[1 1 1]);
colormap jet

Nt_temp = 1;
name_cell = cell(1,plots);
for i = 1:plots
    delta_Nt = floor((Nt+1) / plots);
    plot(x,u(Nt_temp,:));
    hold on
    name_cell{i} = ['t =
',num2str(t(Nt_temp))];
    Nt_temp = Nt_temp + delta_Nt;
end
hold off
legend(name_cell);
```

```
xlabel('x');
ylabel('u');
title('Lax-Wendoff Scheme')

%% Leap-frog Scheme
% \frac{u_{j}^{n+1}-u_{j}^{n-1}}{2\Delta
t}=\frac{u_{j+1}^{n}-u_{j-1}^{n}}{2\Delta
x}
% basic parameters
xspan = [0 3];
tspan = [0 3];
Nx = 100;
Nt = 150;
plots = 5;

% initial variables
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = (tspan(2) - tspan(1))/Nt;
c = delta_t / delta_x;

fprintf('Leap-frog Scheme: c = %f\n',c);
x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
```

```
u = zeros(Nt+1,Nx+1);

% initial mesh nodes
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,1)
initial u using initial condition
u_temp = sin(2*pi*(x+delta_t));  % u(i,0)
visual initial condition

% forward in time step
n = 2;
for i = 2:Nx
    u(n,i) = u_temp(i) - c*( u(n-1,i+1) -
u(n-1,i-1) );        % inner points
end
u(n,1) = u_temp(1) - c*( u(n-1,2) - u(n-
1,Nx) );              % boundary points
u(n,Nx+1) = u_temp(Nx+1) - c*( u(n-1,2) -
u(n-1,Nx) );    % boundary points
```

```
for n = 3:Nt+1
    for i = 2:Nx
        u(n,i) = u(n-2,i) - c*( u(n-1,i+1)
- u(n-1,i-1) );        % inner points
    end
    u(n,1) = u(n-2,1) - c*( u(n-1,2) -
u(n-1,Nx) );             % boundary points
    u(n,Nx+1) = u(n-2,Nx+1) - c*( u(n-1,2)
- u(n-1,Nx) );    % boundary points
end

% plot results
% surf
f1 = figure('Color',[1 1 1]);
axes1 = axes('Parent',f1);
colormap('cool');
[X,T] = meshgrid(x,t);
surf(T,X,u,'Parent',axes1,'EdgeAlpha',0.8,
'EdgeColor','none');
box on;
grid on;
xlabel('t');
```

```
ylabel('x');
zlabel('u');
set(axes1,'FontSize',14,'LineWidth',2);
title('Leap-frog Scheme');
% curves
figure('Color',[1 1 1]);
colormap jet

Nt_temp = 1;
name_cell = cell(1,plots);
for i = 1:plots
    delta_Nt = floor((Nt+1) / plots);
    plot(x,u(Nt_temp,:));
    hold on
    name_cell{i} = ['t =
',num2str(t(Nt_temp))];
    Nt_temp = Nt_temp + delta_Nt;
end
hold off
legend(name_cell);
xlabel('x');
ylabel('u');
```

```
title('Leap-frog Scheme');
%% end
```

2. 第二问程序

```
FOUW.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%%%% Program name: FOUW.m
%%%% Program purpose: Analysis Fist-order
upwind scheme's accurate
%%%% Aurthor: Yang Yang
%%%% Date: 2015.11.05
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%\frac{u_{j}^{n+1}-u_{j}^{n}}{\Delta
t}=\frac{u_{j}^{n}-u_{j-1}^{n}}{\Delta x}

%% basic parameters
```

```
% computational parameters
xspan = [0 3];
Nt = 1;
c = 0.5;
% mesh parameters
Nx_0 = 8;
RefineTimes = 11;

MeanErro = zeros(1,RefineTimes+1);
h = zeros(1,RefineTimes+1);

fprintf('First-order upwind: c = %f\n',c);
%% Compute solution in Initial mesh
Nx = Nx_0;
h(1) = (xspan(2) - xspan(1)) / Nx;
fprintf('Current mesh nodes
amount: %d\n',Nx)

% initial variables
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = c*delta_x;
tspan = [0, Nt*delta_t];

x = zeros(1,Nx+1);
```

```
t = zeros(1,Nt+1);
u = zeros(Nt+1,Nx+1);
u_real = zeros(1,Nx+1);

% initial mesh nodes
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition

%forward in time step
for n = 2:Nt+1
    for i = 2:Nx
        u(n,i) = u(n-1,i) - c*( u(n-1,i) -
u(n-1,i-1) );        % inner points
    end
    u(n,1) = u(n-1,1) - c*( u(n-1,1) -
u(n-1,Nx) );             % boundary points
    u(n,Nx+1) = u(n-1,Nx+1) - c*( u(n-
1,Nx+1) - u(n-1,Nx) );    % boundary
```

```
points
end

u_real(:) = sin(2*pi*(x - delta_t*Nt));
MeanErro(1) = norm(u(Nt+1,:)-u_real,1);

%% Compute solution and refine mesh
parfor k = 1:RefineTimes
    xspan = [0 3];
    Nt = 1;
    N_step = 2^(k);
    Nx = Nx_0*N_step;
    h(k+1) = (xspan(2) - xspan(1)) / Nx;
    fprintf('Current mesh nodes
amount: %d\n',Nx)

    % initial variables
    delta_x = (xspan(2) - xspan(1))/Nx;
    delta_t = c*delta_x;
    tspan = [0, Nt*delta_t];


    x = zeros(1,Nx+1);
    t = zeros(1,Nt+1);
```

```
    u = zeros(Nt+1,Nx+1);
    u_real = zeros(1,Nx+1);

    % initial mesh nodes
    x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
    t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

    u(1,:) = sin(2*pi*(x));          %
u(i,0) initial u using initial condition

    %forward in time step
    for n = 2:Nt+1
      for i = 2:Nx
          u(n,i) = u(n-1,i) - c*( u(n-1,i)
- u(n-1,i-1) );        % inner points
      end
      u(n,1) = u(n-1,1) - c*( u(n-1,1) -
u(n-1,Nx) );             % boundary points
      u(n,Nx+1) = u(n-1,Nx+1) - c*( u(n-
1,Nx+1) - u(n-1,Nx) );    % boundary
points
```

```
    end

     u_real(:) = sin(2*pi*(x -
delta_t*Nt));
     MeanErro(k+1) = norm(u(Nt+1,:)-
u_real,1);
end

%% release memory
clear u u_real;

%% plot
figure('Color',[1 1 1]);
loglog(h,MeanErro,'linewidth',2);
grid on;
xlabel('lg(h)');
ylabel('lg(E)');
title('First-order upwind');
```

FOIM.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%%%% Program name: FOIM.m
%%%% Program purpose: Analysis Fist-order
implicit scheme's accurate
%%%% Aurthor: Yang Yang
%%%% Date: 2015.11.05
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%%%% Fist-order implicit
% \frac{u_{j}^{n+1}-u_{j}^{n}}{\Delta
t}=\frac{u_{j+1}^{n+1}-u_{j-
1}^{n+1}}{2\Delta x}

%% basic parameters
% computational parameters
xspan = [0 3];
Nt = 1;
c = 1.1;
% mesh parameters
Nx_0 = 8;
```

```
RefineTimes = 11;

MeanErro = zeros(1,RefineTimes+1);
h = zeros(1,RefineTimes+1);

fprintf('Fist-order implicit: c
= %f\n',c);
%% Compute solution in Initial mesh
Nx = Nx_0;
h(1) = (xspan(2) - xspan(1)) / Nx;
fprintf('Current mesh nodes
amount: %d\n',Nx)

% initial variables
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = c*delta_x;
tspan = [0, Nt*delta_t];

x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
u = zeros(Nt+1,Nx+1);
u_real = zeros(1,Nx+1);

% initial mesh nodes
```

```
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition

% assemble matrix
A = zeros(Nx+1,Nx+1);

A(1,Nx) = -1/2*c;
A(1,1) = 1;
A(1,2) = 1/2*c;
for i = 2:Nx
    A(i,i-1) = -1/2*c;
    A(i,i) = 1;
    A(i,i+1) = 1/2*c;
end
A(Nx+1,Nx) = -1/2*c;
A(Nx+1,Nx+1) = 1;
A(Nx+1,2) = 1/2*c;
```

```
% forward in time step
for n = 2:Nt+1
    u(n,:) = (A\(u(n-1,:)'))';    % All
points using Matrix Algebra
end

u_real(:) = sin(2*pi*(x - delta_t*Nt));
MeanErro(1) = norm(u(Nt+1,:)-u_real,1);

%% Compute solution and refine mesh
parfor k = 1:RefineTimes
    xspan = [0 3];
    Nt = 1;
    N_step = 2^(k);
    Nx = Nx_0*N_step;
    h(k+1) = (xspan(2) - xspan(1)) / Nx;
    fprintf('Current mesh nodes
amount: %d\n',Nx)

    % initial variables
    delta_x = (xspan(2) - xspan(1))/Nx;
    delta_t = c*delta_x;
    tspan = [0, Nt*delta_t];
```

```
    x = zeros(1,Nx+1);
    t = zeros(1,Nt+1);
    u = zeros(Nt+1,Nx+1);
    u_real = zeros(1,Nx+1);

    % initial mesh nodes
    x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
    t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

    u(1,:) = sin(2*pi*(x));          %
u(i,0) initial u using initial condition

    % assemble matrix
    A = zeros(Nx+1,Nx+1);

    A(1,Nx) = -1/2*c;
    A(1,1) = 1;
    A(1,2) = 1/2*c;
    for i = 2:Nx
        A(i,i-1) = -1/2*c;
        A(i,i) = 1;
```

```
        A(i,i+1) = 1/2*c;
    end
    A(Nx+1,Nx) = -1/2*c;
    A(Nx+1,Nx+1) = 1;
    A(Nx+1,2) = 1/2*c;

    % forward in time step
    for n = 2:Nt+1
        u(n,:) = (A\(u(n-1,:)'))';    % All
points using Matrix Algebra
    end

      u_real(:) = sin(2*pi*(x -
delta_t*Nt));
      MeanErro(k+1) = norm(u(Nt+1,:)-
u_real,1);
end

%% release memory
clear u u_real;

%% plot
figure('Color',[1 1 1]);
```

```
loglog(h,MeanErro,'linewidth',2);
grid on;
xlabel('lg(h)');
ylabel('lg(E)');
title('First-order implicit');
```

Lax.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%%%% Program name: Lax.m
%%%% Program purpose: Analysis Lax
scheme's accurate
%%%% Aurthor: Yang Yang
%%%% Date: 2015.11.05
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
% \frac{u_{j}^{n+1}-
\frac{1}{2}\left( u_{j+1}^{n}+u_{j-
```

```
1}^{n}\right)  }{\Delta
t}=\frac{u_{j+1}^{n}-u_{j-1}^{n}}{2\Delta
x}%

%% basic parameters
% computational parameters
xspan = [0 3];
Nt = 1;
c = 0.5;
% mesh parameters
Nx_0 = 8;
RefineTimes = 11;

MeanErro = zeros(1,RefineTimes+1);
h = zeros(1,RefineTimes+1);

fprintf('Lax: c = %f\n',c);
%% Compute solution in Initial mesh
Nx = Nx_0;
h(1) = (xspan(2) - xspan(1)) / Nx;
fprintf('Current mesh nodes
amount: %d\n',Nx)

% initial variables
```

```
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = c*delta_x;
tspan = [0, Nt*delta_t];

x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
u = zeros(Nt+1,Nx+1);
u_real = zeros(1,Nx+1);

% initial mesh nodes
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition

% forward in time step
for n = 2:Nt+1
    for i = 2:Nx
        u(n,i) = 0.5*(1-c)*u(n-1,i+1) -
0.5*(1+c)*u(n-1,i-1);      % inner points
    end
```

```
    u(n,1) = 0.5*(1-c)*u(n-1,2) -
0.5*(1+c)*u(n-1,Nx);                % 
boundary points
    u(n,Nx+1) = 0.5*(1-c)*u(n-1,2) -
0.5*(1+c)*u(n-1,Nx);        % boundary
points
end

u_real(:) = sin(2*pi*(x - delta_t*Nt));
MeanErro(1) = norm(u(Nt+1,:)-u_real,1);

%% Compute solution and refine mesh
parfor k = 1:RefineTimes
    xspan = [0 3];
    Nt = 1;
    N_step = 2^(k);
    Nx = Nx_0*N_step;
    h(k+1) = (xspan(2) - xspan(1)) / Nx;
    fprintf('Current mesh nodes
amount: %d\n',Nx)

    % initial variables
    delta_x = (xspan(2) - xspan(1))/Nx;
```

```
    delta_t = c*delta_x;
    tspan = [0, Nt*delta_t];

    x = zeros(1,Nx+1);
    t = zeros(1,Nt+1);
    u = zeros(Nt+1,Nx+1);
    u_real = zeros(1,Nx+1);

    % initial mesh nodes
    x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
    t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

    u(1,:) = sin(2*pi*(x));          %
u(i,0) initial u using initial condition

% forward in time step
for n = 2:Nt+1
    for i = 2:Nx
        u(n,i) = 0.5*(1-c)*u(n-1,i+1) -
0.5*(1+c)*u(n-1,i-1);      % inner points
    end
```

```
    u(n,1) = 0.5*(1-c)*u(n-1,2) -
0.5*(1+c)*u(n-1,Nx);                %
boundary points
    u(n,Nx+1) = 0.5*(1-c)*u(n-1,2) -
0.5*(1+c)*u(n-1,Nx);        % boundary
points
end

     u_real(:) = sin(2*pi*(x -
delta_t*Nt));
     MeanErro(k+1) = norm(u(Nt+1,:)-
u_real,1);
end

%% release memory
clear u u_real;

%% plot
figure('Color',[1 1 1]);
loglog(h,MeanErro,'linewidth',2);
grid on;
xlabel('lg(h)');
ylabel('lg(E)');
```

```
title('Lax');
```

```
Lax-Wendroff.m
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% Program name: Lax-Wendroff.m
%%%% Program purpose: Analysis Lax-Wendroff Scheme scheme's accurate
%%%% Aurthor: Yang Yang
%%%% Date: 2015.11.05
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% \frac{u_{j}^{n+1}-u_{j}^{n}}{\Delta t}+\frac{u_{j+1}^{n}-u_{j-1}^{n}}{2\Delta x}=\frac{1}{2}\Delta t\frac{u_{j+1}^{n}-2u_{j}^{n}+u_{j-1}^{n}}{\Delta x^{2}}

%% basic parameters
% computational parameters
```

```
xspan = [0 3];
Nt = 1;
c = 0.5;
% mesh parameters
Nx_0 = 8;
RefineTimes = 11;

MeanErro = zeros(1,RefineTimes+1);
h = zeros(1,RefineTimes+1);

fprintf('Lax-Wendroff: c = %f\n',c);
%% Compute solution in Initial mesh
Nx = Nx_0;
h(1) = (xspan(2) - xspan(1)) / Nx;
fprintf('Current mesh nodes
amount: %d\n',Nx)

% initial variables
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = c*delta_x;
tspan = [0, Nt*delta_t];

x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
```

```
u = zeros(Nt+1,Nx+1);
u_real = zeros(1,Nx+1);

% initial mesh nodes
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition

% forward in time step
for n = 2:Nt+1
    for i = 2:Nx
        u(n,i) = (1-c^2)*u(n-1,i) +
0.5*c*(c-1)*u(n-1,i+1) + 0.5*c*(c+1)*u(n-
1,i-1);  % inner points
    end
    u(n,1) = (1-c^2)*u(n-1,1) + 0.5*c*(c-
1)*u(n-1,2) + 0.5*c*(c+1)*u(n-1,Nx);      %
boundary points
    u(n,Nx+1) = (1-c^2)*u(n-1,Nx+1) +
```

```
0.5*c*(c-1)*u(n-1,2) + 0.5*c*(c+1)*u(n-
1,Nx);      % boundary points
end

u_real(:) = sin(2*pi*(x - delta_t*Nt));
MeanErro(1) = norm(u(Nt+1,:)-u_real,1);

%% Compute solution and refine mesh
parfor k = 1:RefineTimes
    xspan = [0 3];
    Nt = 1;
    N_step = 2^(k);
    Nx = Nx_0*N_step;
    h(k+1) = (xspan(2) - xspan(1)) / Nx;
    fprintf('Current mesh nodes
amount: %d\n',Nx)

    % initial variables
    delta_x = (xspan(2) - xspan(1))/Nx;
    delta_t = c*delta_x;
    tspan = [0, Nt*delta_t];


    x = zeros(1,Nx+1);
```

```
    t = zeros(1,Nt+1);
    u = zeros(Nt+1,Nx+1);
    u_real = zeros(1,Nx+1);

    % initial mesh nodes
    x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
    t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

    u(1,:) = sin(2*pi*(x));          %
u(i,0) initial u using initial condition

% forward in time step
for n = 2:Nt+1
    for i = 2:Nx
        u(n,i) = (1-c^2)*u(n-1,i) +
0.5*c*(c-1)*u(n-1,i+1) + 0.5*c*(c+1)*u(n-
1,i-1);  % inner points
    end
    u(n,1) = (1-c^2)*u(n-1,1) + 0.5*c*(c-
1)*u(n-1,2) + 0.5*c*(c+1)*u(n-1,Nx);      %
boundary points
```

```
    u(n,Nx+1) = (1-c^2)*u(n-1,Nx+1) +
0.5*c*(c-1)*u(n-1,2) + 0.5*c*(c+1)*u(n-
1,Nx);      % boundary points
end

     u_real(:) = sin(2*pi*(x -
delta_t*Nt));
     MeanErro(k+1) = norm(u(Nt+1,:)-
u_real,1);
end

%% release memory
clear u u_real;

%% plot
figure('Color',[1 1 1]);
loglog(h,MeanErro,'linewidth',2);
grid on;
xlabel('lg(h)');
ylabel('lg(E)');
title('LaxWendroff');
```

LeapFrog.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%%%% Program name: LeapFrog.m
%%%% Program purpose: Analysis Leap-frog
scheme's accurate
%%%% Aurthor: Yang Yang
%%%% Date: 2015.11.05
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
% \frac{u_{j}^{n+1}-u_{j}^{n-1}}{2\Delta
t}=\frac{u_{j+1}^{n}-u_{j-1}^{n}}{2\Delta
x}

%% basic parameters
% computational parameters
xspan = [0 3];
Nt = 1;
c = 0.5;
% mesh parameters
```

```
Nx_0 = 8;
RefineTimes = 11;

MeanErro = zeros(1,RefineTimes+1);
h = zeros(1,RefineTimes+1);

fprintf('LeapFrog: c = %f\n',c);
%% Compute solution in Initial mesh
Nx = Nx_0;
h(1) = (xspan(2) - xspan(1)) / Nx;
fprintf('Current mesh nodes
amount: %d\n',Nx)

% initial variables
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = c*delta_x;
tspan = [0, Nt*delta_t];

x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
u = zeros(Nt+1,Nx+1);
u_real = zeros(1,Nx+1);

% initial mesh nodes
```

```
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition
u_temp = sin(2*pi*(x+delta_t));  % u(i,0)
visual initial condition

% forward in time step
n = 2;
for i = 2:Nx
    u(n,i) = u_temp(i) - c*( u(n-1,i+1) -
u(n-1,i-1) );        % inner points
end
u(n,1) = u_temp(1) - c*( u(n-1,2) - u(n-
1,Nx) );             % boundary points
u(n,Nx+1) = u_temp(Nx+1) - c*( u(n-1,2) -
u(n-1,Nx) );    % boundary points


for n = 3:Nt+1
    for i = 2:Nx
```

```
        u(n,i) = u(n-2,i) - c*( u(n-1,i+1)
- u(n-1,i-1) );        % inner points
    end
    u(n,1) = u(n-2,1) - c*( u(n-1,2) -
u(n-1,Nx) );             % boundary points
    u(n,Nx+1) = u(n-2,Nx+1) - c*( u(n-1,2)
- u(n-1,Nx) );    % boundary points
end

u_real(:) = sin(2*pi*(x - delta_t*Nt));
MeanErro(1) = norm(u(Nt+1,:)-u_real,1);

%% Compute solution and refine mesh
parfor k = 1:RefineTimes
    xspan = [0 3];
    Nt = 1;
    N_step = 2^(k);
    Nx = Nx_0*N_step;
    h(k+1) = (xspan(2) - xspan(1)) / Nx;
    fprintf('Current mesh nodes
amount: %d\n',Nx)

    % initial variables
```

```
    delta_x = (xspan(2) - xspan(1))/Nx;
    delta_t = c*delta_x;
    tspan = [0, Nt*delta_t];

    x = zeros(1,Nx+1);
    t = zeros(1,Nt+1);
    u = zeros(Nt+1,Nx+1);
    u_real = zeros(1,Nx+1);

    % initial mesh nodes
    x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
    t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

    u(1,:) = sin(2*pi*(x));          %
u(i,0) initial u using initial condition
    u_temp = sin(2*pi*(x+delta_t));  %
u(i,0) visual initial condition

% forward in time step
n = 2;
for i = 2:Nx
```

```
    u(n,i) = u_temp(i) - c*( u(n-1,i+1) -
u(n-1,i-1) );        % inner points
end
u(n,1) = u_temp(1) - c*( u(n-1,2) - u(n-
1,Nx) );             % boundary points
u(n,Nx+1) = u_temp(Nx+1) - c*( u(n-1,2) -
u(n-1,Nx) );    % boundary points

for n = 3:Nt+1
    for i = 2:Nx
        u(n,i) = u(n-2,i) - c*( u(n-1,i+1)
- u(n-1,i-1) );        % inner points
    end
    u(n,1) = u(n-2,1) - c*( u(n-1,2) -
u(n-1,Nx) );             % boundary points
    u(n,Nx+1) = u(n-2,Nx+1) - c*( u(n-1,2)
- u(n-1,Nx) );    % boundary points
end

     u_real(:) = sin(2*pi*(x -
delta_t*Nt));
     MeanErro(k+1) = norm(u(Nt+1,:)-
```

```
u_real,1);
end

%% release memory
clear u u_real;

%% plot
figure('Color',[1 1 1]);
loglog(h,MeanErro,'linewidth',2);
grid on;
xlabel('lg(h)');
ylabel('lg(E)');
title('LeapFrog');
```

3. 第三问程序

OneDimensionWaveEquationProblem3.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%
%%%% Program name:
OneDimensionWaveEquationProblem3.m
```

```
%%%% Program purpose: Compute and Analysis
different numerical scheme
%%%% Aurthor: Yang Yang
%%%% Date: 2015.11.05
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%

%% Fist-order upwind
%\frac{u_{j}^{n+1}-u_{j}^{n}}{\Delta
t}=\frac{u_{j}^{n}-u_{j-1}^{n}}{\Delta x}
% basic parameters
xspan = [0 3];
tspan = [0 3];
Nx = 100;
Nt = 150;
plots = 5;

% initial variables
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = (tspan(2) - tspan(1))/Nt;
c = delta_t / delta_x;
```

```
fprintf('First-order upwind: c = %f\n',c);
x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
u = zeros(Nt+1,Nx+1);
u_real = zeros(1,Nx+1);

% initial mesh nodes
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition

% forward in time step
for n = 2:Nt+1
    for i = 2:Nx
        u(n,i) = u(n-1,i) - c*( u(n-1,i) -
u(n-1,i-1) );        % inner points
    end
    u(n,1) = u(n-1,1) - c*( u(n-1,1) -
u(n-1,Nx) );             % boundary points
```

```
    u(n,Nx+1) = u(n-1,Nx+1) - c*( u(n-
1,Nx+1) - u(n-1,Nx) );    % boundary
points
end


% Compute real solution
u_real(:) = sin(2*pi*(x -
delta_t*Nt));     % compute u's real value
in final time step

% plot results
% curves
figure1=figure('Color',[1 1 1]);
axes1 = axes('Parent',figure1);
Nt_temp = Nt+1;
name_cell = cell(1,2);
plot(x,u(Nt_temp,:),'b','linewidth',2);
hold on
plot(x,u_real(:),'r','linewidth',2)
hold off
name_cell{1} = ['t = ',num2str(Nt_temp),',
Numerical'];
```

```
name_cell{2} = ['t = ',num2str(Nt_temp),',
Exact'];
set(axes1,'XTick',[0 0.25 0.5 0.75 1 1.25
1.5 1.75 2 2.25 2.5 2.75 3]);
legend(name_cell);
xlabel('x');
ylabel('u');
grid on;
title('Fist-order upwind')

%% Implicit Scheme
% \frac{u_{j}^{n+1}-u_{j}^{n}}{\Delta
t}=\frac{u_{j+1}^{n+1}-u_{j-
1}^{n+1}}{2\Delta x}

% basic parameters
xspan = [0 3];
tspan = [0 3.1];
Nx = 100;
Nt = 100;
plots = 5;

% initial variables
```

```
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = (tspan(2) - tspan(1))/Nt;
c = delta_t / delta_x;

fprintf('Implicit Scheme: c = %f\n',c);
x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
u = zeros(Nt+1,Nx+1);

% initial mesh nodes
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition

% assemble matrix
A = zeros(Nx+1,Nx+1);

A(1,Nx) = -1/2*c;
A(1,1) = 1;
A(1,2) = 1/2*c;
```

```
for i = 2:Nx
    A(i,i-1) = -1/2*c;
    A(i,i) = 1;
    A(i,i+1) = 1/2*c;
end
A(Nx+1,Nx) = -1/2*c;
A(Nx+1,Nx+1) = 1;
A(Nx+1,2) = 1/2*c;

% forward in time step
for n = 2:Nt+1
    u(n,:) = (A\(u(n-1,:)'))';    % All
points using Matrix Algebra
end

% plot results
% curves
figure1=figure('Color',[1 1 1]);
axes1 = axes('Parent',figure1);
Nt_temp = Nt+1;
name_cell = cell(1,2);
plot(x,u(Nt_temp,:),'b','linewidth',2);
```

```
hold on
plot(x,u_real(:),'r','linewidth',2)
hold off
name_cell{1} = ['t = ',num2str(Nt_temp),',
Numerical'];
name_cell{2} = ['t = ',num2str(Nt_temp),',
Exact'];
set(axes1,'XTick',[0 0.25 0.5 0.75 1 1.25
1.5 1.75 2 2.25 2.5 2.75 3]);
legend(name_cell);
xlabel('x');
ylabel('u');
grid on;
title('Fist-order implicit')

%% Lax Scheme
% \frac{u_{j}^{n+1}-
\frac{1}{2}\left(  u_{j+1}^{n}+u_{j-
1}^{n}\right)  }{\Delta
t}=\frac{u_{j+1}^{n}-u_{j-1}^{n}}{2\Delta
x}%
% basic parameters
```

```
xspan = [0 3];

tspan = [0 3];

Nx = 100;

Nt = 150;

plots = 5;

% initial variables

delta_x = (xspan(2) - xspan(1))/Nx;

delta_t = (tspan(2) - tspan(1))/Nt;

c = delta_t / delta_x;

fprintf('Lax Scheme: c = %f\n',c);

x = zeros(1,Nx+1);

t = zeros(1,Nt+1);

u = zeros(Nt+1,Nx+1);

% initial mesh nodes

x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes

t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition
```

```
% forward in time step
for n = 2:Nt+1
    for i = 2:Nx
        u(n,i) = 0.5*(1-c)*u(n-1,i+1) -
0.5*(1+c)*u(n-1,i-1);      % inner points
    end
    u(n,1) = 0.5*(1-c)*u(n-1,2) -
0.5*(1+c)*u(n-1,Nx);                %
boundary points
    u(n,Nx+1) = 0.5*(1-c)*u(n-1,2) -
0.5*(1+c)*u(n-1,Nx);       % boundary
points
end

% plot results
% curves
figure1=figure('Color',[1 1 1]);
axes1 = axes('Parent',figure1);
Nt_temp = Nt+1;
name_cell = cell(1,2);
plot(x,u(Nt_temp,:),'b','linewidth',2);
hold on
```

```
plot(x,u_real(:),'r','linewidth',2)
hold off
name_cell{1} = ['t = ',num2str(Nt_temp),',
Numerical'];
name_cell{2} = ['t = ',num2str(Nt_temp),',
Exact'];
set(axes1,'XTick',[0 0.25 0.5 0.75 1 1.25
1.5 1.75 2 2.25 2.5 2.75 3]);
legend(name_cell);
xlabel('x');
ylabel('u');
grid on;
title('Lax Scheme')

%% Lax-Wendroff Scheme
% \frac{u_{j}^{n+1}-u_{j}^{n}}{\Delta
t}+\frac{u_{j+1}^{n}-u_{j-1}^{n}}{2\Delta
x}=\frac{1}{2}\Delta t\frac{u_{j+1}^{n}-
2u_{j}^{n}+u_{j-1}^{n}}{\Delta x^{2}}
% basic parameters
xspan = [0 3];
tspan = [0 3];
```

```
Nx = 100;
Nt = 150;
plots = 5;

% initial variables
delta_x = (xspan(2) - xspan(1))/Nx;
delta_t = (tspan(2) - tspan(1))/Nt;
c = delta_t / delta_x;

fprintf('Lax-Wendroff Scheme: c
= %f\n',c);
x = zeros(1,Nx+1);
t = zeros(1,Nt+1);
u = zeros(Nt+1,Nx+1);

% initial mesh nodes
x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes
t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,0)
initial u using initial condition
```

```
% forward in time step
for n = 2:Nt+1
    for i = 2:Nx
        u(n,i) = (1-c^2)*u(n-1,i) +
0.5*c*(c-1)*u(n-1,i+1) + 0.5*c*(c+1)*u(n-
1,i-1);  % inner points
    end
    u(n,1) = (1-c^2)*u(n-1,1) + 0.5*c*(c-
1)*u(n-1,2) + 0.5*c*(c+1)*u(n-1,Nx);      %
boundary points
    u(n,Nx+1) = (1-c^2)*u(n-1,Nx+1) +
0.5*c*(c-1)*u(n-1,2) + 0.5*c*(c+1)*u(n-
1,Nx);      % boundary points
end

% plot results
% curves
figure1=figure('Color',[1 1 1]);
axes1 = axes('Parent',figure1);
Nt_temp = Nt+1;
name_cell = cell(1,2);
plot(x,u(Nt_temp,:),'b','linewidth',2);
```

```
hold on
plot(x,u_real(:),'r','linewidth',2)
hold off
name_cell{1} = ['t = ',num2str(Nt_temp),',
Numerical'];
name_cell{2} = ['t = ',num2str(Nt_temp),',
Exact'];
set(axes1,'XTick',[0 0.25 0.5 0.75 1 1.25
1.5 1.75 2 2.25 2.5 2.75 3]);
legend(name_cell);
xlabel('x');
ylabel('u');
grid on;
title('Lax-Wendoff Scheme')

%% Leap-frog Scheme
% \frac{u_{j}^{n+1}-u_{j}^{n-1}}{2\Delta
t}=\frac{u_{j+1}^{n}-u_{j-1}^{n}}{2\Delta
x}
% basic parameters
xspan = [0 3];
tspan = [0 3];
```

```
Nx = 100;

Nt = 150;

plots = 5;

% initial variables

delta_x = (xspan(2) - xspan(1))/Nx;

delta_t = (tspan(2) - tspan(1))/Nt;

c = delta_t / delta_x;

fprintf('Leap-frog Scheme: c = %f\n',c);

x = zeros(1,Nx+1);

t = zeros(1,Nt+1);

u = zeros(Nt+1,Nx+1);

% initial mesh nodes

x = xspan(1):delta_x:xspan(2);   % x(i)
spatial nodes

t = tspan(1):delta_t:tspan(2);   % t(i)
time nodes

u(1,:) = sin(2*pi*(x));          % u(i,1)
initial u using initial condition

u_temp = sin(2*pi*(x+delta_t));  % u(i,0)
visual initial condition
```

```
% forward in time step
n = 2;
for i = 2:Nx
    u(n,i) = u_temp(i) - c*( u(n-1,i+1) -
u(n-1,i-1) );        % inner points
end
u(n,1) = u_temp(1) - c*( u(n-1,2) - u(n-
1,Nx) );             % boundary points
u(n,Nx+1) = u_temp(Nx+1) - c*( u(n-1,2) -
u(n-1,Nx) );    % boundary points

for n = 3:Nt+1
    for i = 2:Nx
        u(n,i) = u(n-2,i) - c*( u(n-1,i+1)
- u(n-1,i-1) );        % inner points
    end
    u(n,1) = u(n-2,1) - c*( u(n-1,2) -
u(n-1,Nx) );             % boundary points
    u(n,Nx+1) = u(n-2,Nx+1) - c*( u(n-1,2)
- u(n-1,Nx) );    % boundary points
end
```

```
% plot results
% curves
figure1=figure('Color',[1 1 1]);
axes1 = axes('Parent',figure1);
Nt_temp = Nt+1;
name_cell = cell(1,2);
plot(x,u(Nt_temp,:),'b','linewidth',2);
hold on
plot(x,u_real(:),'r','linewidth',2)
hold off
name_cell{1} = ['t = ',num2str(Nt_temp),',
Numerical'];
name_cell{2} = ['t = ',num2str(Nt_temp),',
Exact'];
set(axes1,'XTick',[0 0.25 0.5 0.75 1 1.25
1.5 1.75 2 2.25 2.5 2.75 3]);
legend(name_cell);
xlabel('x');
ylabel('u');
grid on;
title('Leap-frog Scheme');
%% end
```