华南理工大学
South China University of Technology

课程设计报告书

题目：微分方程数值解第二次大作业报告

学　院　　数学学院

专　业　　数学与应用数学

学生姓名　雷皓兰

学生学号　202130320717

指导教师　黄凤辉

课程编号

课程学分　3.0

起始日期　2023.10.28

教 师 评 语	教师签名： 日期：
成 绩 评 定	
备 注	

抛物型方程几种最简差分格式

编程计算：分别采用向前差分格式、向后差分格式、六点对称格式求

解如下抛物型方程

$$\frac{\partial u}{\partial t}-a\frac{\partial^2 u}{\partial x^2}=f(x,t),\qquad (x,t)\in\Omega$$

其中a、$f(x,t)$、Ω 及初边条件为：

（1）

1. $a=\frac{1}{16}$, $f(x,y)=0$, $\Omega=\{(x,t)\,|\,0<x<1;\ 0<t<1\}$,

且初边条件如下：$\begin{cases} u(x,0)=2\sin(2\pi x),\ 0<x<1; \\ u(0,t)=u(1,t)=0,\quad 0<t<1. \end{cases}$

存在精确解为：$u(x,t)=2e^{-\frac{\pi^2 t}{4}}\sin(2\pi x)$.

首先定义函数结构体：

```
function pde = model_data()
pde = struct('u_initial',@u_initial,'u_left',@u_left,...
    'u_right',@u_right,'f',@f,'time_grid',@time_grid,...
    'space_grid',@space_grid,'a',@a);

    function [T,tau] = time_grid(NT)
        T = linspace(0,1,NT+1);
        tau = 1/NT;
    end
    function [X,h] = space_grid(NS)
        X = linspace(0,1,NS+1)';
        h = 1/NS;
    end
    function u = u_initial(x)
        u = 2*sin(2*pi*x);
    end
    function u = u_left(t)
        u = zeros(size(t));
    end
    function u = u_right(t)
        u = zeros(size(t));
    end
    function f = f(x,t)
        f = zeros(size(x));
```

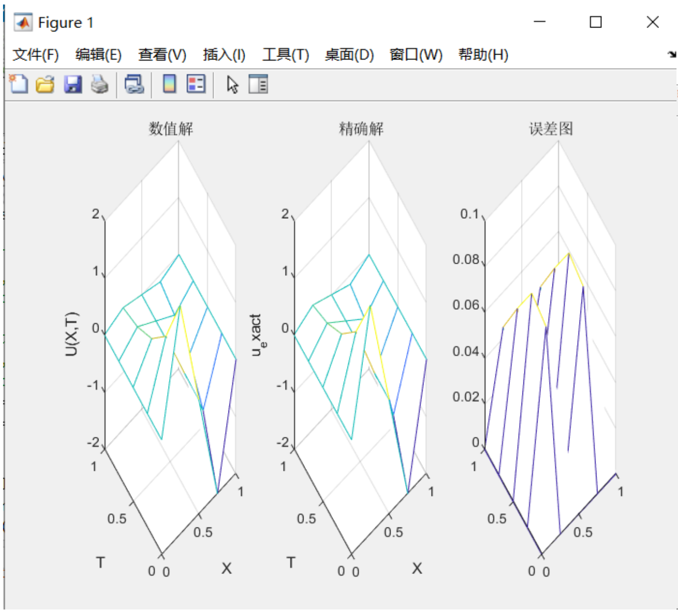
```
[X,h] = pde.space_grid(NS);
[T,tau] = pde.time_grid(NT);
N = length(X);M = length(T);
r = pde.a()*tau/h/h;
U = zeros(N,M);
U(:,1) = pde.u_initial(X);
U(1,:) = pde.u_left(T);
U(end,:) = pde.u_right(T);
```

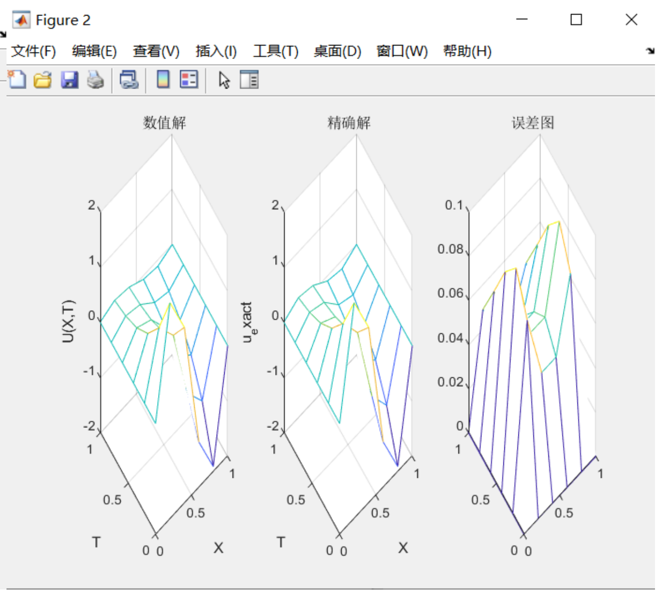
向前差分格式：

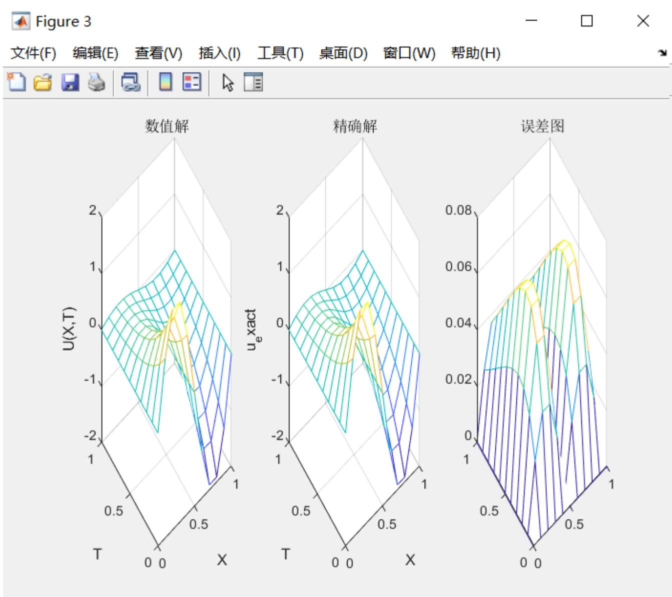
公式代码：

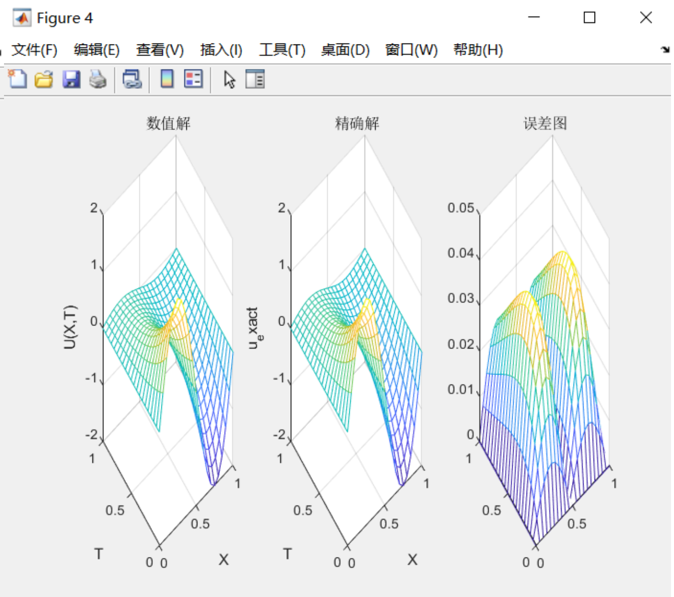
```
pde = model_data(); %模型数据结构体
n=[4,5,10,20,100];
H=zeros(1,5);
e=zeros(1,5);
u_exact=@(x,t) 2*exp(-pi^2*t/4)*sin(x*2*pi);
for i=1:5
[X,T,U1] = heat_equation_fd1d(n(i),n(i),pde,'forward');
showsolution(X,T,U1,i);
H(i)=log(r);
e(i)=getmaxerror(X,T,U1,u_exact);
end
P=polyfit(H,e,1);
figure(6)
plot(H,e,'*',H,polyval(P,H),'b-')
xlabel('步长')
ylabel('误差')
title('误差与步长的关系')
p=P(1);%收敛阶

function forward()
       d = 1 - 2*ones(N-2,1)*r;
       c = ones(N-3,1)*r;
       A = diag(c,-1) + diag(c,1)+diag(d);
       for i = 2:M
           RHS = tau*pde.f(X,T(i));
           RHS(2) = RHS(2) + r*U(1,i-1);
           RHS(end-1) = RHS(end-1) + r*U(end,i-1);
           U(2:end-1,i)=A*U(2:end-1,i-1)+ RHS(2:end-1);
       end
   end
```

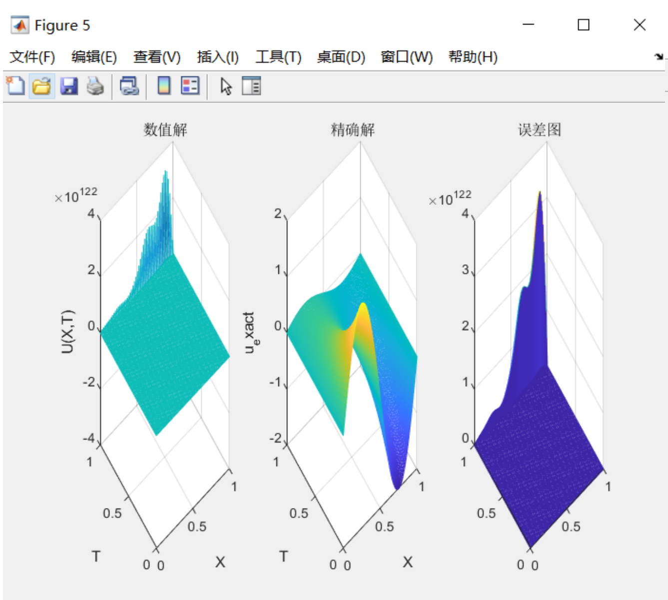
取不同步长求解结果：

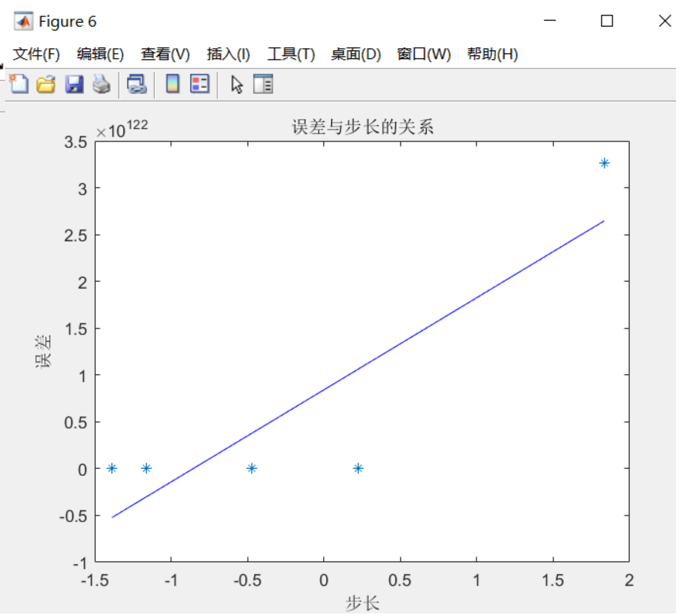












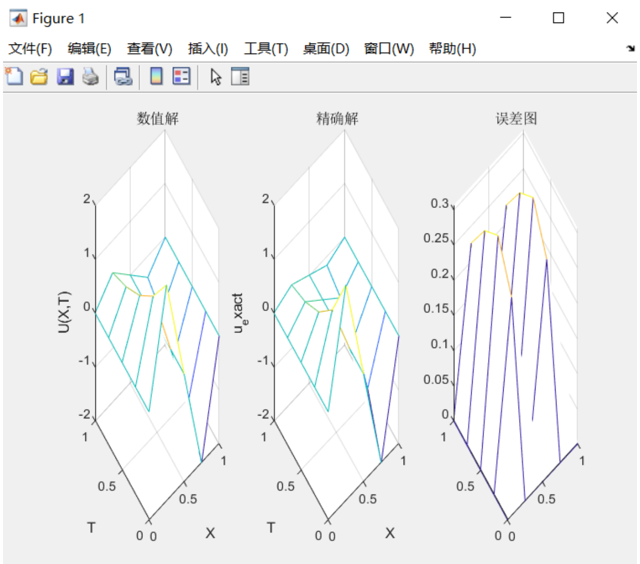
可见，本题中向前差分格式取步长 0.01 时，计算结果已经不稳定。去除不稳定的点，拟合图变为：

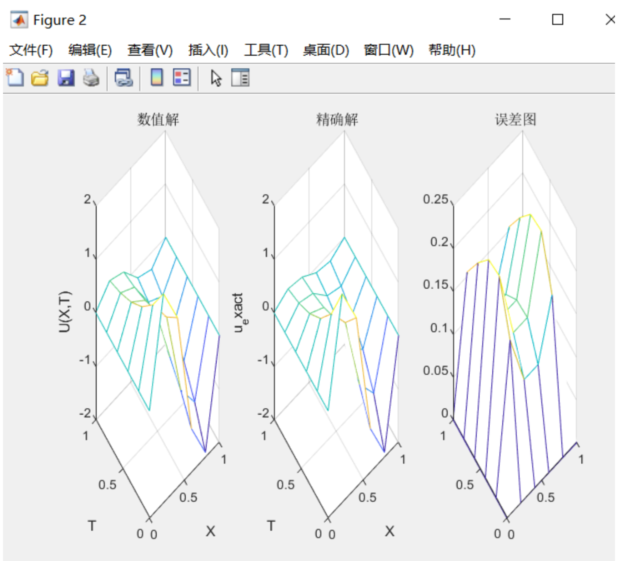
向后差分格式：

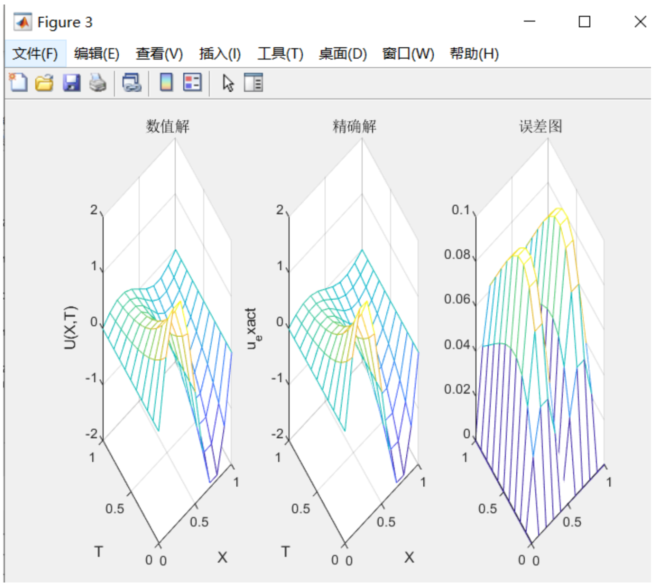
公式代码：

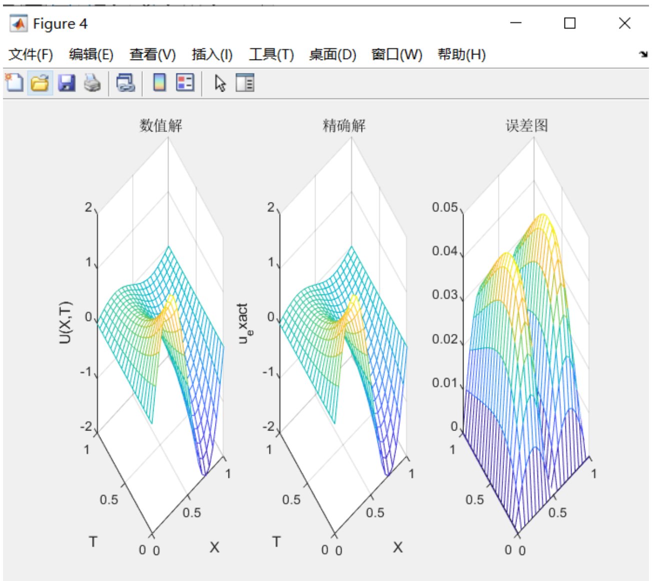
```
pde = model_data(); %模型数据结构体
n=[4,5,10,20,100];
H=zeros(1,5);
e=zeros(1,5);
u_exact=@(x,t) 2*exp(-pi^2*t/4)*sin(x*2*pi);
for i=1:5
[X,T,U2] = heat_equation_fd1d(n(i),n(i),pde,'backward');
showsolution(X,T,U2,i);
H(i)=log(r);
e(i)=getmaxerror(X,T,U2,u_exact);
end
P=polyfit(H,e,1);
figure(6)
plot(H,e,'*',H,polyval(P,H),'b-')
xlabel('步长')
ylabel('误差')
title('误差与步长的关系')
p=P(1);%收敛阶

function backward()
       d = 1 + 2*ones(N-2,1)*r;
       c = -ones(N-3,1)*r;
       A = diag(c,-1) + diag(c,1)+diag(d);
       for i = 2:M
           RHS = tau*pde.f(X,T(i));
           RHS(2) = RHS(2) + r*U(1,i);
           RHS(end-1) = RHS(end-1) + r*U(end,i);
           U(2:end-1,i)=A\(U(2:end-1,i-1)+ RHS(2:end-1));
       end
   end
```

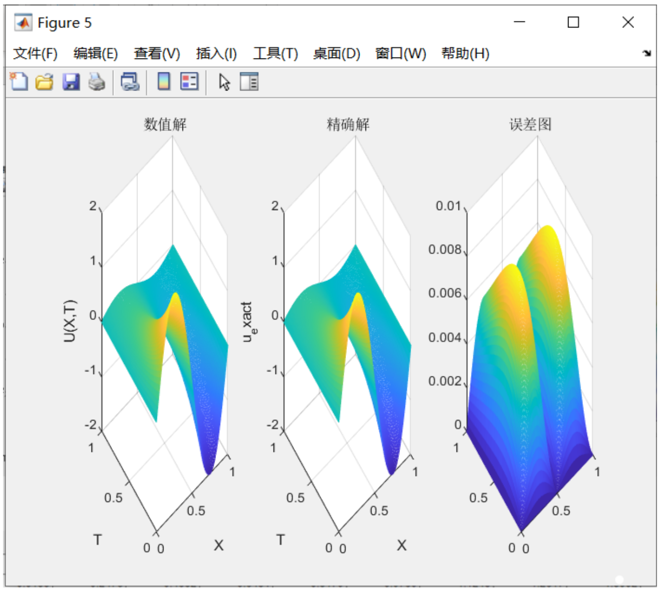
取不同步长计算结果：

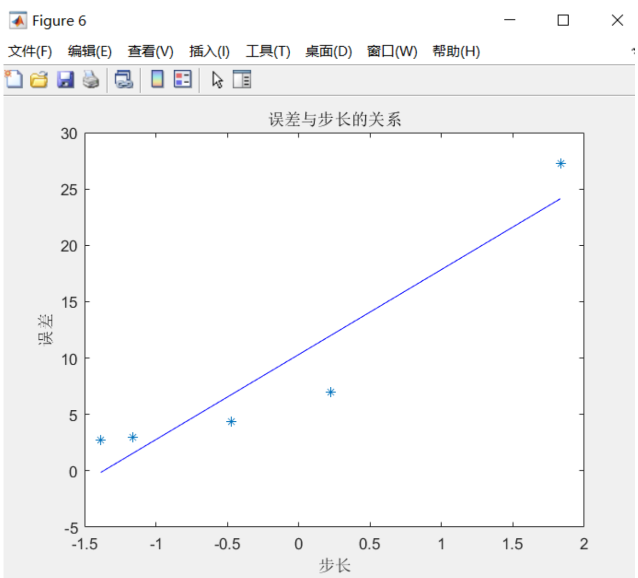












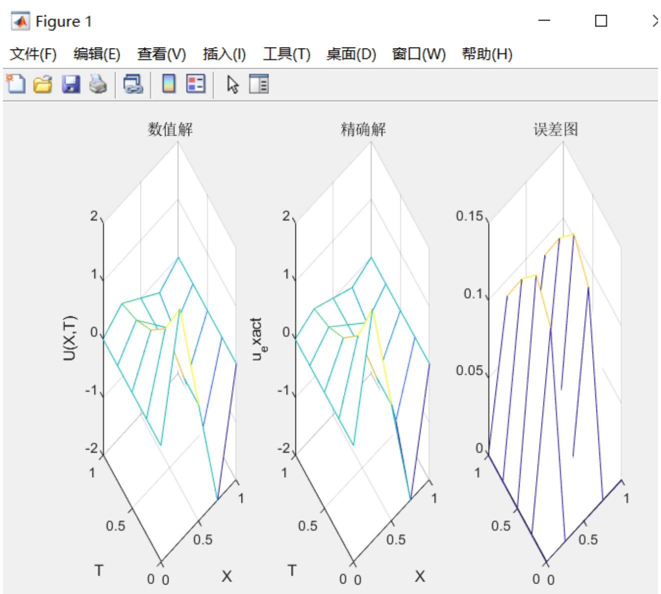
六点对称格式：

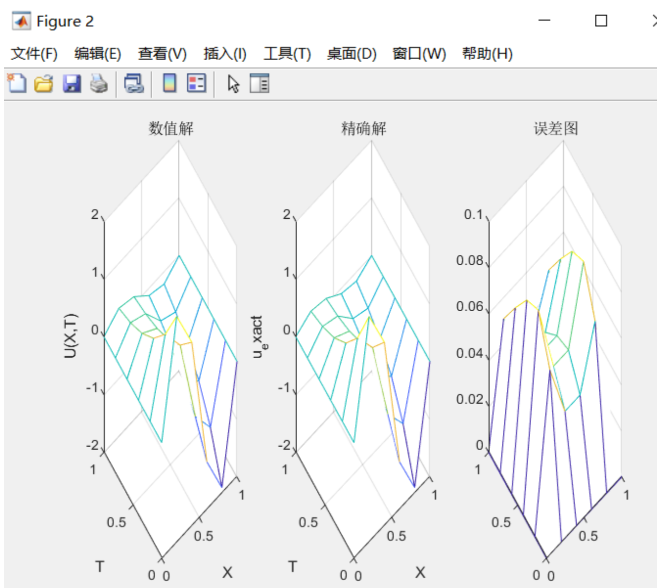
公式代码：

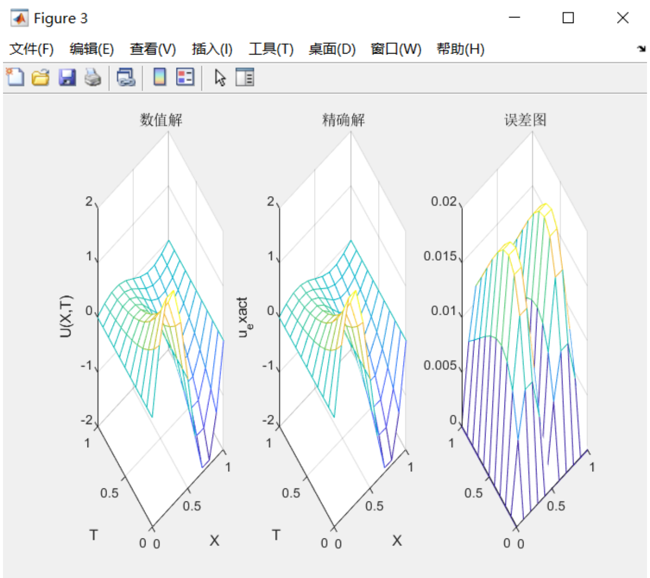
```
pde = model_data(); %模型数据结构体
```

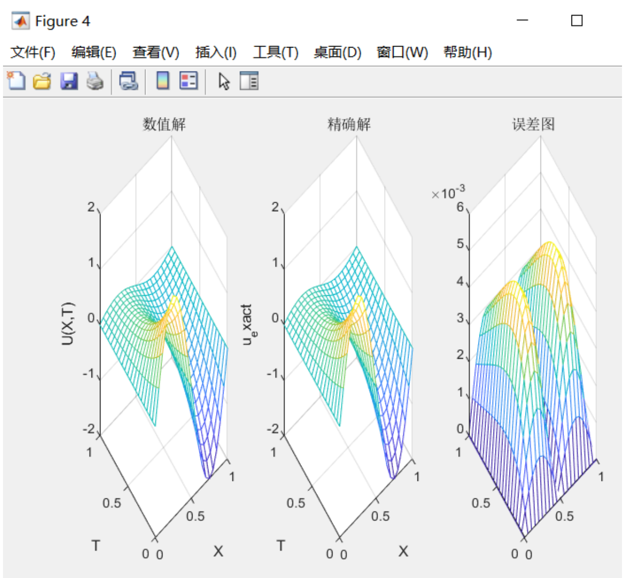
```
n=[4,5,10,20,100];
H=zeros(1,5);
e=zeros(1,5);
u_exact=@(x,t) 2*exp(-pi^2*t/4)*sin(x*2*pi);
for i=1:5
[X,T,U2] = heat_equation_fd1d(n(i),n(i),pde,' crank-nicholson');
showsolution(X,T,U3,i);
H(i)=log(r);
e(i)=getmaxerror(X,T,U3,u_exact);
end
P=polyfit(H,e,1);
figure(6)
plot(H,e,'*',H,polyval(P,H),'b-')
xlabel('步长')
ylabel('误差')
title('误差与步长的关系')
p=P(1);%收敛阶

function crank_nicholson()
        d1 = 1 + ones(N-2,1)*r;
        d2 = 1 - ones(N-2,1)*r;
        c = 0.5*ones(N-3,1)*r;
        A1 = diag(-c,-1) + diag(-c,1)+diag(d1);
        A0 = diag(c,-1) + diag(c,1) + diag(d2);
        for i = 2:M
            RHS = tau*pde.f(X,T(i));
            RHS(2) = RHS(2) + 0.5*r*(U(1,i)+U(1,i-1));
            RHS(end-1) = RHS(end-1) + ...
                0.5*r*(U(end,i)+U(end,i-1));
            U(2:end-1,i)=A1\(A0*U(2:end-1,i-1)+ RHS(2:end-1));
        end
    end
```

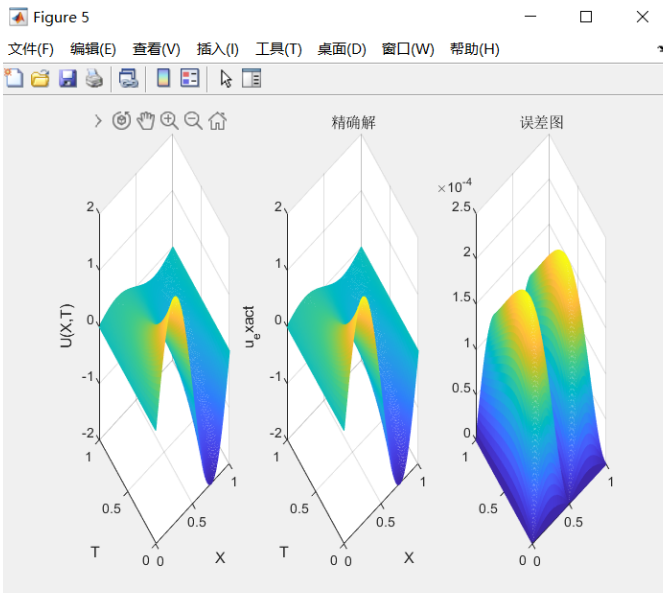
取不同步长计算结果:



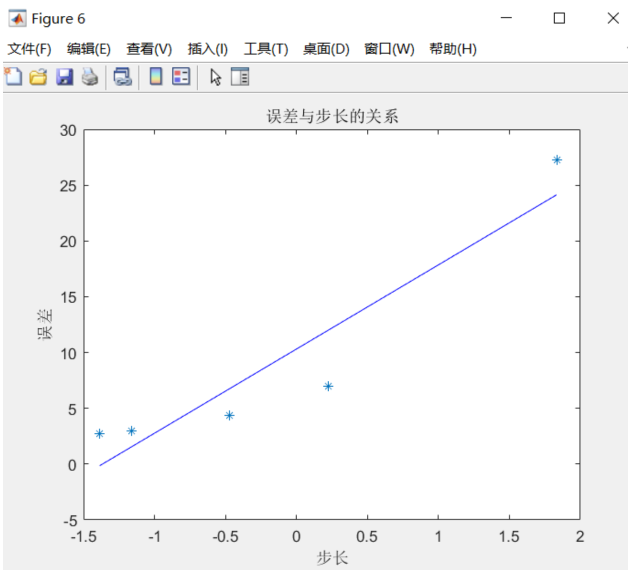




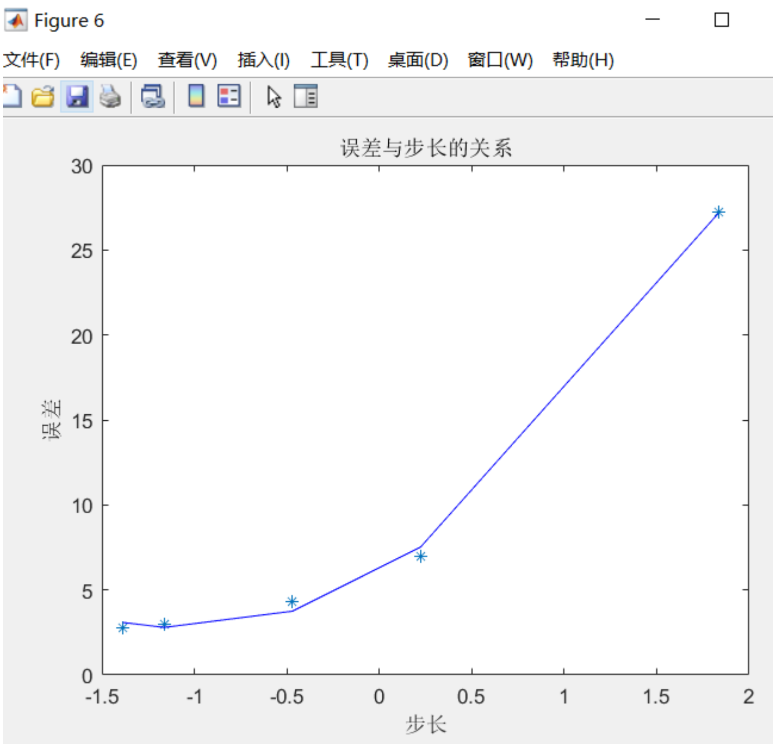




一阶精度拟合：



二阶精度拟合：



可见六点对称格式为二阶精度

（2）

2.　$a=1,\ f(x,y)=2,\ \ \Omega=\{(x,t)\,|\,0<x<1;0<t<2\}$，

且初边条件如下：$\begin{cases} u(x,0)=\sin(\pi x)+x(1-x),\ \ 0<x<1; \\ u(0,t)=u(1,t)=0,\ \ \ t>0. \end{cases}$

存在精确解为：　$u(x,t)=e^{-\pi^2 t}\sin(\pi x)+x(1-x)$

首先定义函数结构体：

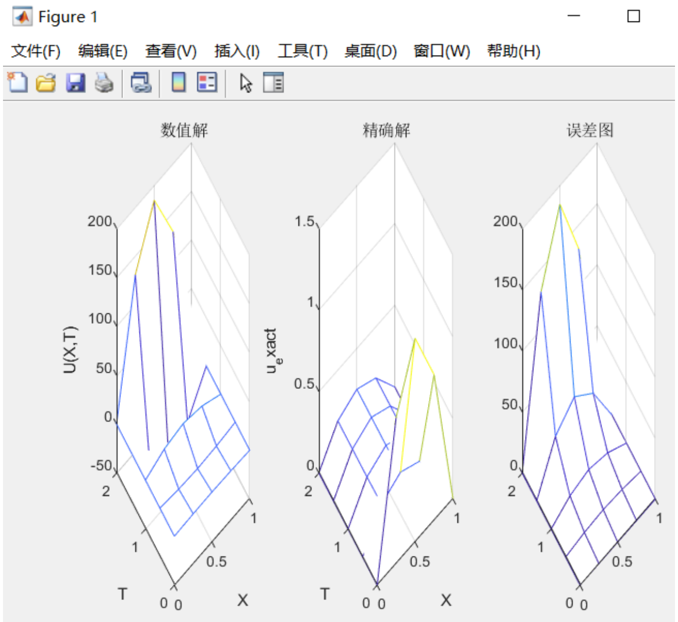
```
function pde = model_data()
pde = struct('u_initial',@u_initial,'u_left',@u_left,...
    'u_right',@u_right,'f',@f,'time_grid',@time_grid,...
    'space_grid',@space_grid,'a',@a);

    function [T,tau] = time_grid(NT)
        T = linspace(0,2,NT+1);
        tau = 2/NT;
    end
    function [X,h] = space_grid(NS)
        X = linspace(0,1,NS+1)';
        h = 1/NS;
    end
    function u = u_initial(x)
        u = sin(pi*x)+x.*(1-x);
```

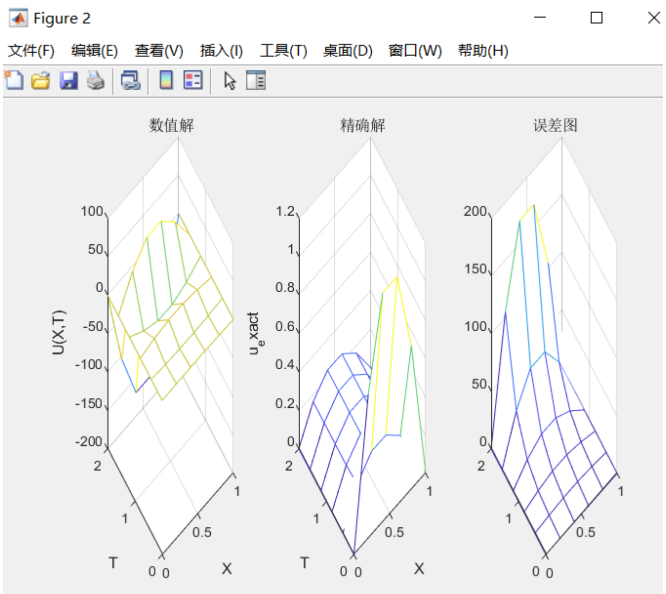
```
    end
    function u = u_left(t)
        u = zeros(size(t));
    end
    function u = u_right(t)
        u = zeros(size(t));
    end
    function f = f(x,t)
        f = 2*ones(size(x));
    end
    function a = a()
        a = 1;
    end
end
```

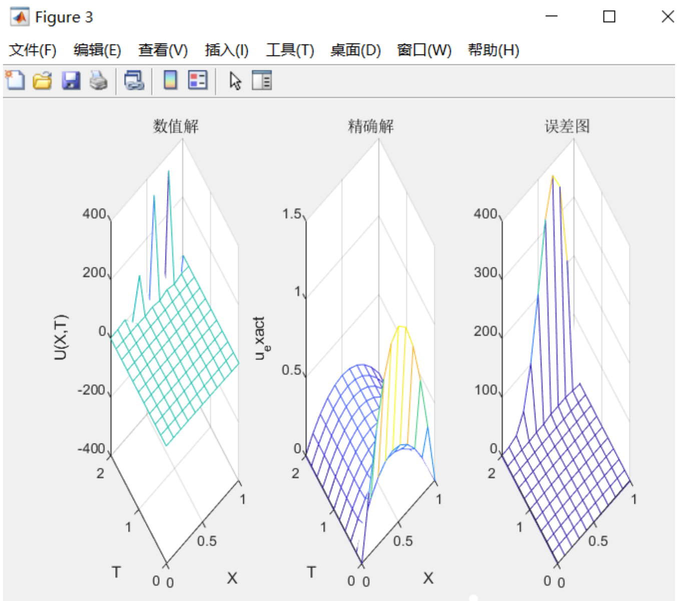
向前差分格式

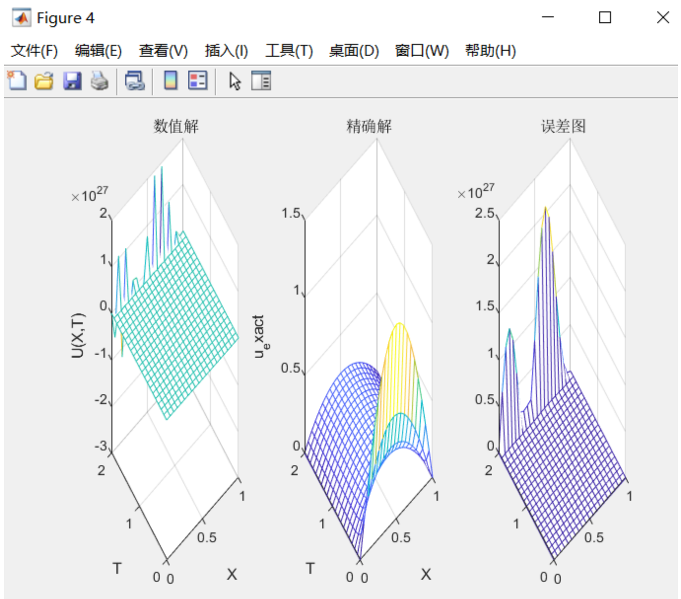
公式代码：同上题一样

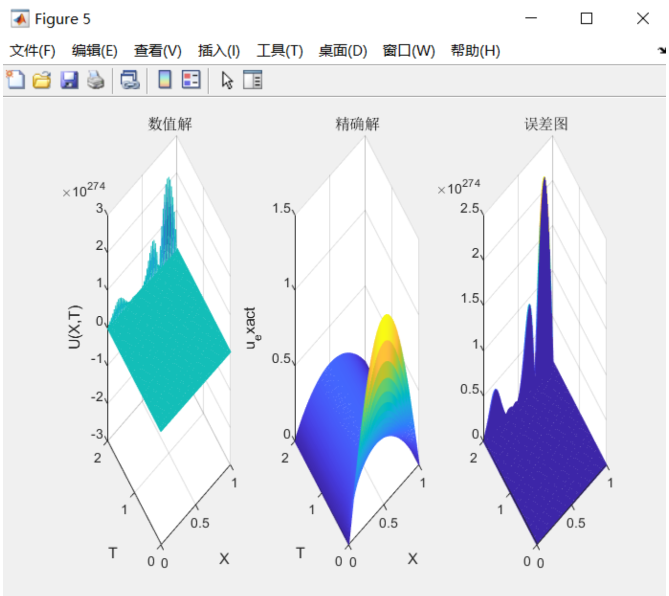
取不同步长计算结果：

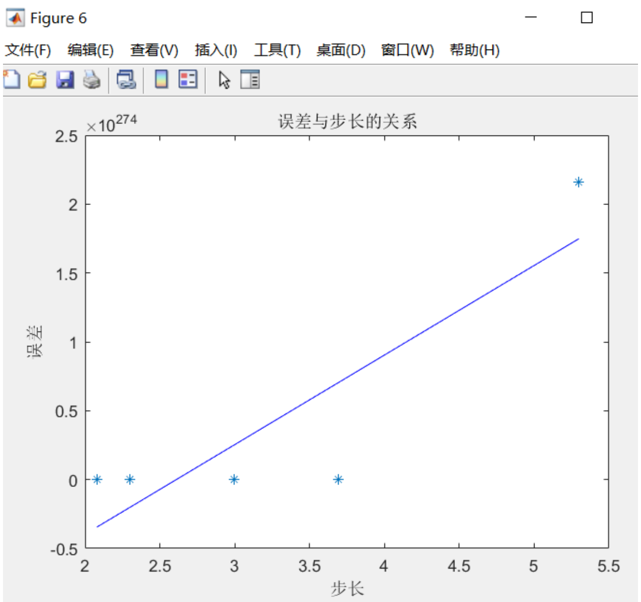








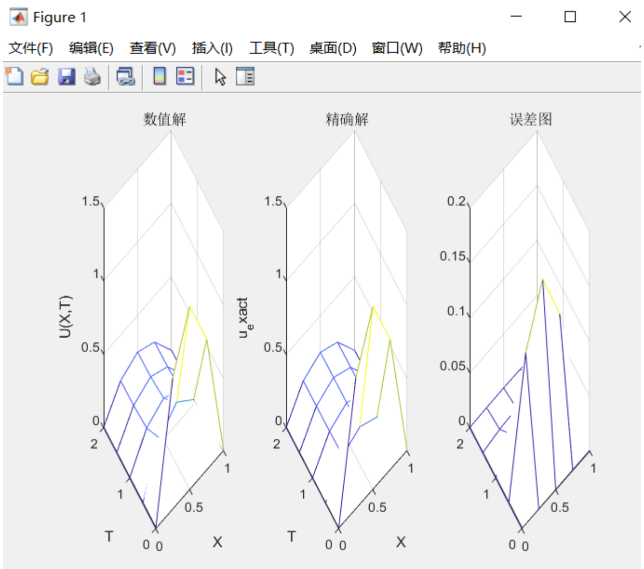


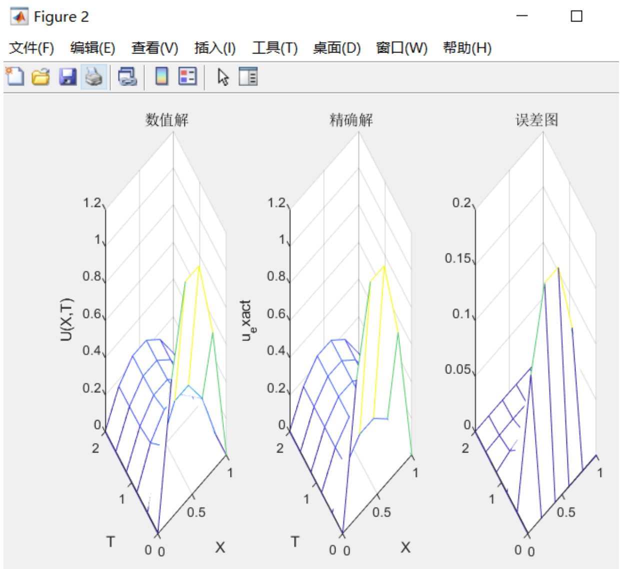


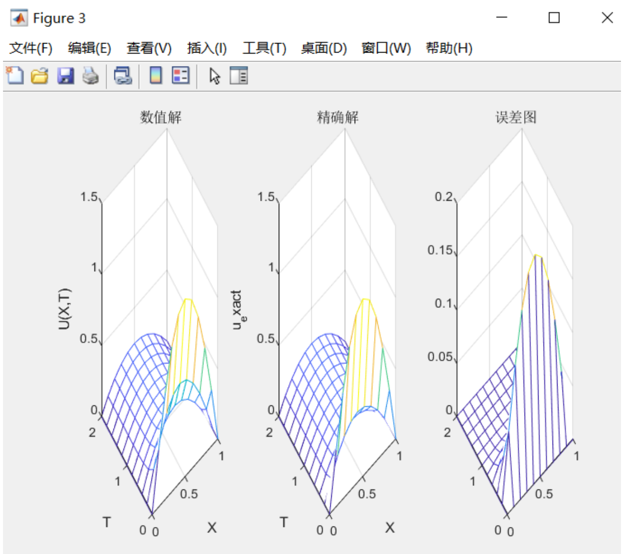
向后差分格式：

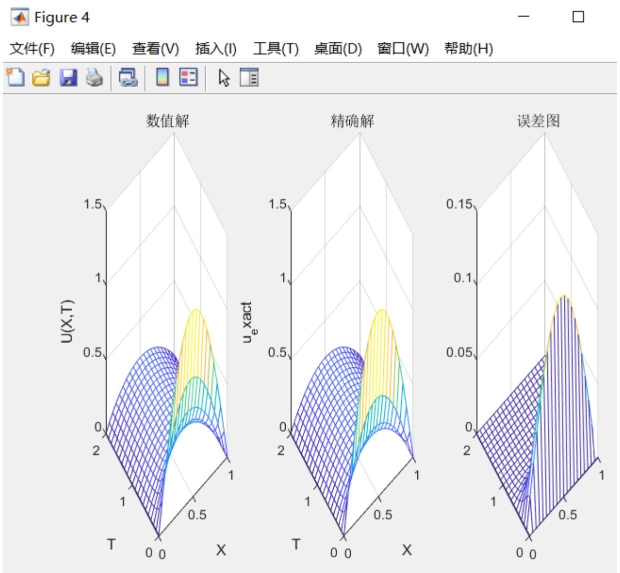
公式代码：同上题一样

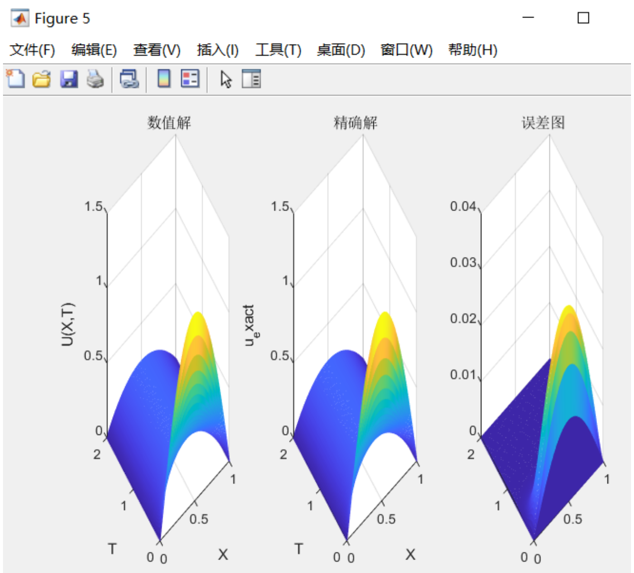
取不同步长计算结果：

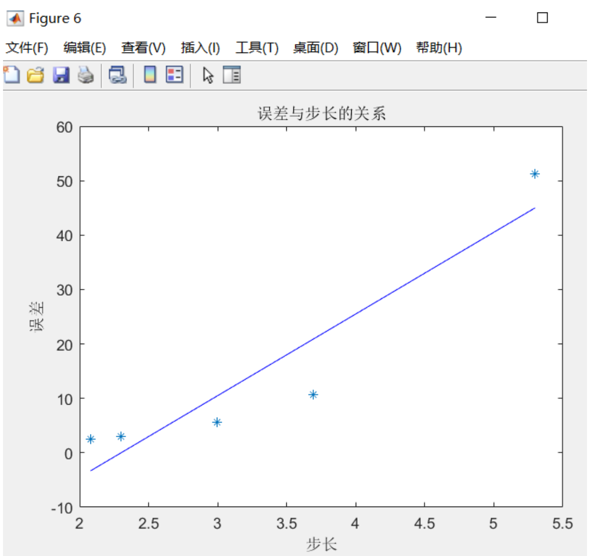












六点对称格式：

公式代码：同上题一样

取不同步长计算结果：

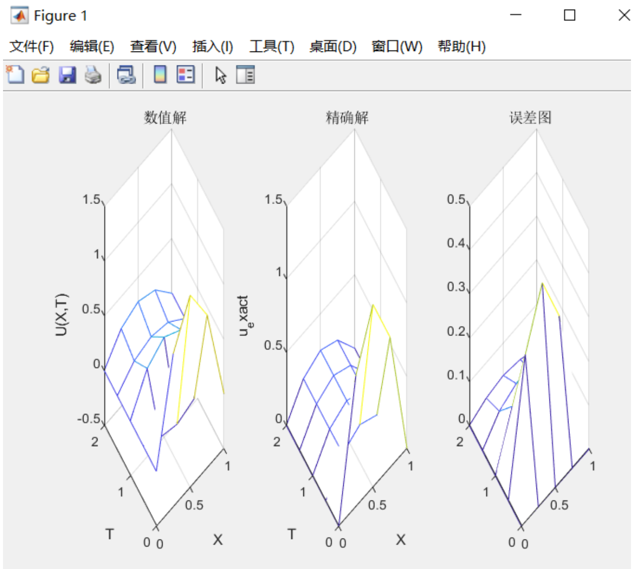


Figure 1
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
数值解
精确解
误差图
U(X,T)
u_exact
T
X

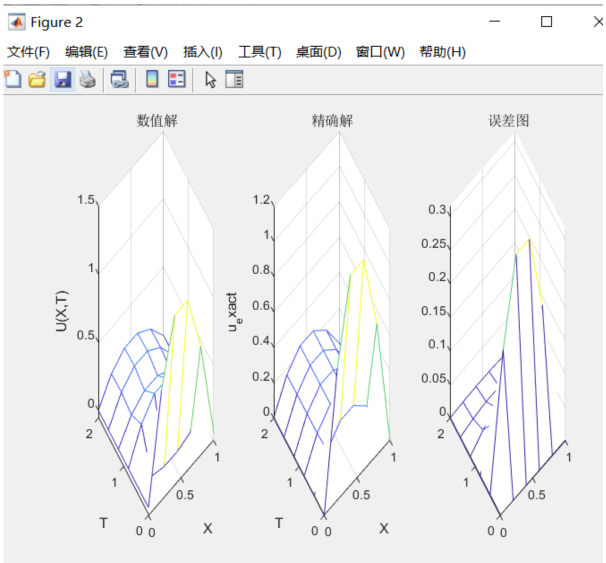


Figure 2
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
数值解
精确解
误差图
U(X,T)
u_exact
T
X

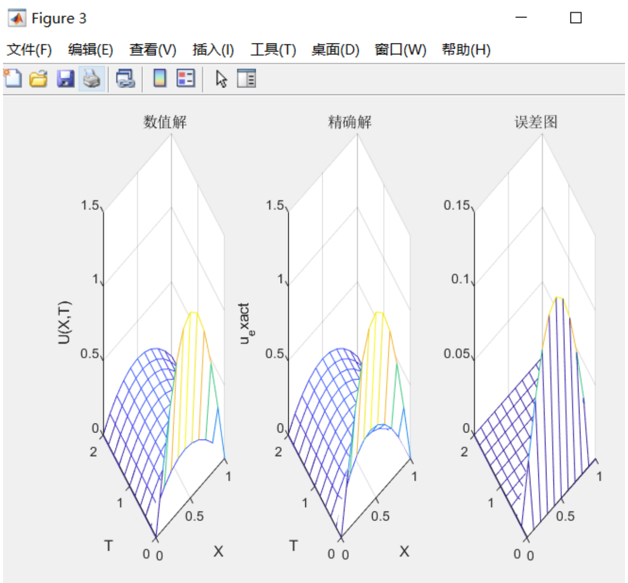


Figure 3
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
数值解
精确解
误差图
U(X,T)
u_exact
T
X

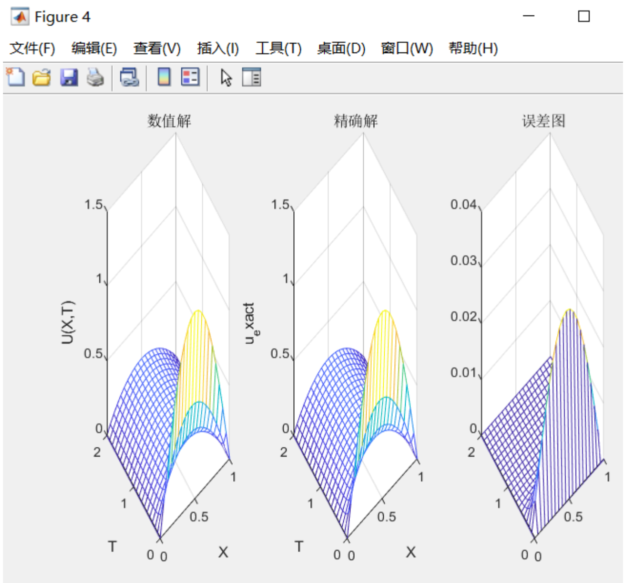


Figure 4
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
数值解
精确解
误差图
U(X,T)
u_exact
T
X

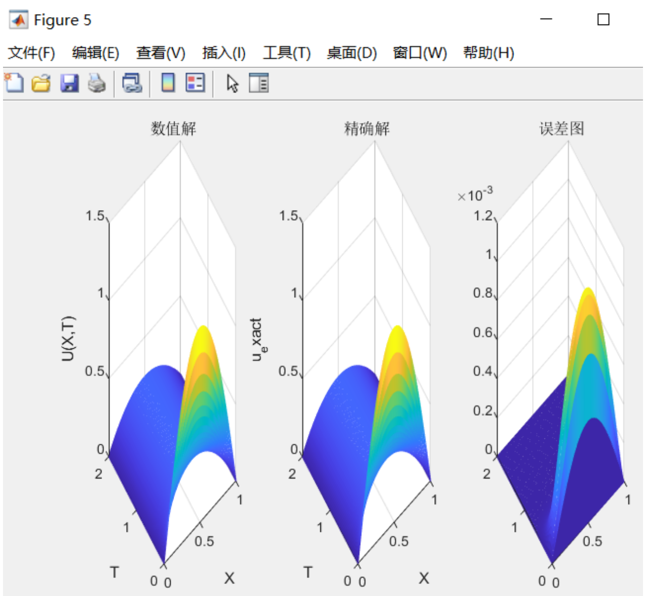
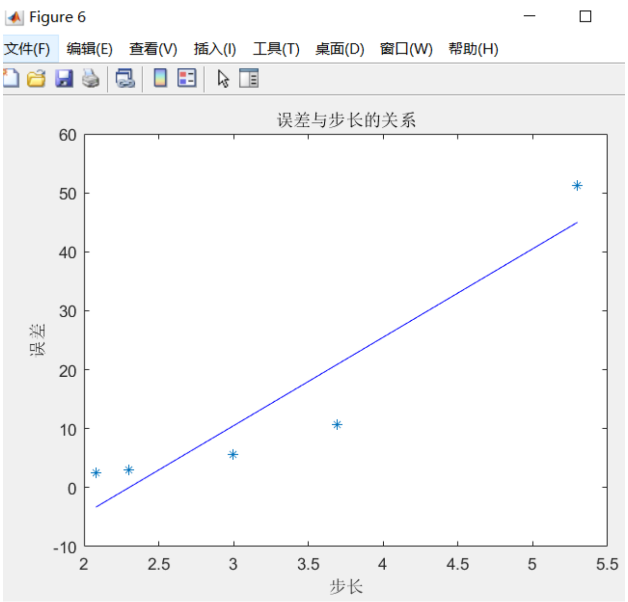
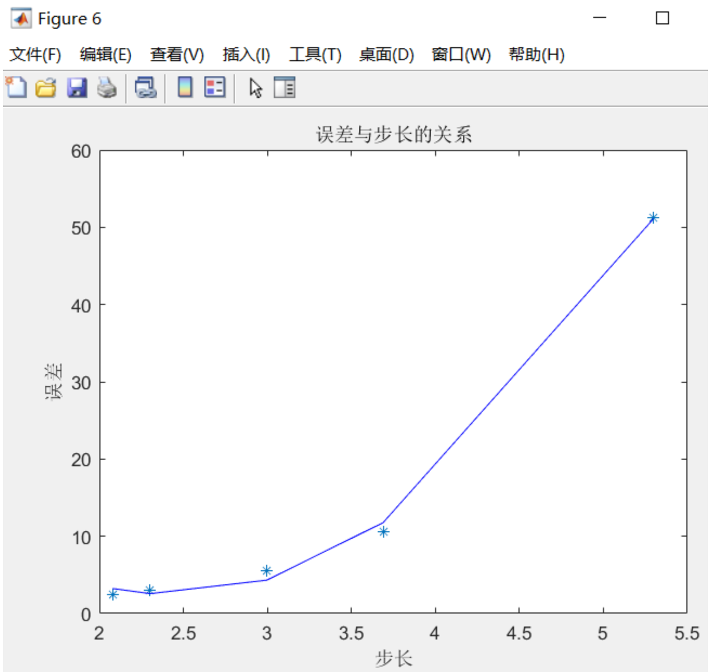


Figure 5
文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 桌面(D) 窗口(W) 帮助(H)
数值解
精确解
误差图
×10^-3
U(X,T)
u_exact
T
X

一阶精度拟合：



二阶精度拟合：



可见六点对称格式为二阶精度

过程论述

参考书上理论知识，同时借鉴了网上的代码，经过不断地修改，完成了这次大作业。

总结

向前差分格式和向后差分格式都是一阶的格式，不过向前差分格式的精度比向后差分格式要低一些。由于向后差分格式是用后一时刻的值计算当前时刻的值，所以比向前差分格式更加稳定。六点对称格式是二阶精度的格式，比向前差分格式和向后差

分格式更加精确。但是由于计算时需要用到更多的点的信息，计算量比较大。

参考文献

[1]《微分方程数值解》