计算流体力学第二次大作业

倪子珺 PB19051007

2022 年 5 月 10 日

目录

1 问题描述

在一个矩形区域 $0 \leq x \leq 1$，$0 \leq x \leq 1$ 内，数值求解下面问题：

已知控制方程为：

$$\frac{\partial u}{\partial t} = \sigma \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \text{（扩散方程）} \tag{1}$$

其解析解为

$$u(x, y, t) = 20 + 80 \left[y - e^{-0.5\sigma\pi^2 t} \sin\left(\frac{\pi x}{2}\right) \sin\left(\frac{\pi y}{2}\right) \right] \tag{2}$$

式中 $\sigma = \sigma_x = \sigma_y = 1$。初始条件为

$$u(x, y)|_{t=0} = 20 + 80 \left[y - \sin\left(\frac{\pi x}{2}\right) \sin\left(\frac{\pi y}{2}\right) \right] \quad (0 < x < 1, 0 < y < 1)$$

边界条件

$$\begin{aligned}
&u(0, y, t) = 20 + 80y = f_1(y, t) \quad (x = 0) \\
&u(1, y, t) = 20 + 80 \left[y - e^{-0.5\sigma\pi^2 t} \sin\left(\frac{\pi}{2} y\right) \right] = f_2(y, t) \quad (x = 1) \\
&u(x, 0, t) = 20 = g_1(x, t) \quad (y = 0) \\
&u(x, 1, t) = 20 + 80 \left[1 - e^{-0.5\sigma\pi^2 t} \sin\left(\frac{\pi}{2} x\right) \right] = g_2(x, t) \quad (y = 1)
\end{aligned}$$

试求解下列问题：

(1) 采用 ADI 或 AF 格式离散上述偏微分方程；

(2) 分别在：

1. 时间步长 $\Delta t = 0.02$，网格间距为 $\Delta x = 0.1,\ 0.05,\ 0.025$ 的均匀网格；
2. 网格间距 $\Delta x = \Delta y = 0.05$ ，时间步长 $\Delta t = 0.005, 0.01, 0.02, 0.04$；

的情况下数值求解问题（1）中所得到的差分方程；

(3) 定义数值误差 $\varepsilon(t) = \|u_{\text{exact}}(t) - u_{\text{num}}(t)\|$，分别给出 $\varepsilon(1)$ 与时间步长、网格间距之间的关系（使用对数坐标系, 即 $\log \varepsilon \sim \log h$），观察是否为线性，斜率如何？并试解释其原因，分析收敛性；

(4) 定义计算误差 $E_h = \left\| u_{i,j}^{n+1} - u_{i,j}^{n} \right\|$（即 $E_h = \max_{i,j \in \Omega} \left| u_{i,j}^{n+1} - u_{i,j}^{n} \right|$），作出 $\log E_h$ 随时间变化曲线。

2 问题求解和方程的 ADI 离散

2.1 内点的离散

由 ADI 格式可以将方程(1)写成如下两步格式：

$$\begin{cases} u_{i,j}^{*}-u_{i,j}^{n}=\dfrac{1}{2}\Delta t\left[\sigma_x\dfrac{u_{i+1,j}^{*}+u_{i-1,j}^{*}-2u_{i,j}^{*}}{\Delta(\Delta x)^2}+\sigma_y\cdot\dfrac{u_{i,j+1}^{2n}+u_{i,j-1}^{n}-2u_{i,j}^{n}}{(\Delta y)^2}\right] & (3)\\ u_{i,j}^{n+1}-u_{i,j}^{*}=\dfrac{1}{2}\Delta t\left[\sigma_x\dfrac{u_{i+1,j}^{*}+u_{i-1,j}^{*}-2u_{i,j}^{*}}{(\Delta x)^2}+\sigma_j\cdot\dfrac{u_{i,j+1}^{n+1}+u_{i,j,-1}^{n+1}-2u_{i,j}^{n}}{(\Delta y)^2}\right] & (4)\end{cases}$$

由于题中所用网格均为均匀网格（即 $\Delta x=\Delta y=\Delta l$），则上两式可以整理为：

$$\begin{cases} \alpha u_{i+1,j}^{*}-(2\alpha+1)u_{i,j}^{*}+\alpha u_{i-1,j}^{*}=-\alpha u_{i,j+1}^{n}-\alpha u_{i,j-1}^{n}+(2\alpha-1)u_{i,j}^{n} & (5)\\ \alpha u_{i,j+1}^{n+1}-(2\alpha+1)u_{i,j}^{n+1}+\alpha u_{i,j-1}^{n+1}=-\alpha u_{i+1,j}^{*}-\alpha u_{i-1,j}^{*}+(2\alpha-1)u_{i,j}^{*} & (6)\end{cases}$$

其中 $\alpha=\sigma\Delta t/2(\Delta l)^2$。

2.2 边界点的处理

由于题中的边界条件均为第一类边界条件，故在边界处可以直接求得导数项，无需利用差分方法。

当 $x=0$，即 $i=1$ 时，此时无需求解原来的两步法，因

$$\begin{cases} u_{1,j}^{*}=f_1(j\Delta y,(n+1/2)\Delta t).\\ u_{1,j}^{n+1}=f_1(j\Delta y,(n+1)\Delta t)\end{cases}$$

均可直接由边界条件得到。其他边界条件同理，列出如下：

$$\begin{cases} u_{N,j}^{*}=f_2\left(j\Delta y_{(}(n+1/2)\Delta t\right),u_{N,j}^{n+1}=f_2(j\Delta y,(n+1)\Delta t)\\ u_{i,1}^{*}=g_1(i\Delta x,(n+1/2)\Delta t),u_{i,1}^{n+1}=g_1(i\Delta x,(n+1)\Delta t)\\ u_{i,N}^{*}=g_2(i\Delta x,(n+1/2)\Delta t),u_{i,N,}^{n+1}=g_2(i\Delta x,(n+1)\Delta t)\end{cases} \quad (7)$$

2.3 求解流程简述

由 2.1和 2.2中的分析，最终得到总体的算法如下：

第一步求解 对于每一个 $j=2,3,\cdots,N-1$，先求解如下线性方程组：

$$\begin{bmatrix} 1 & 0 & & & & & \\ \alpha & -(2\alpha+1) & \alpha & & & & \\ & \alpha & -(2\alpha+1) & \alpha & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \alpha & -(2\alpha+1) & \alpha & \\ & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} u^*_{1,j} \\ u^*_{2,j} \\ u^*_{3,j} \\ \vdots \\ u^*_{N-1,j} \\ u^*_{N,j} \end{bmatrix} = \begin{bmatrix} f_1(j\Delta y,(n+1/2)\Delta t) \\ -\alpha u^n_{2,j+1}-\alpha u^n_{2,j-1}+(2\alpha-1)u^n_{2,j} \\ -\alpha u^n_{3,j+1}-\alpha u^n_{3,j-1}+(2\alpha-1)u^n_{3,j} \\ \vdots \\ -\alpha u^n_{N-1,j+1}-\alpha u^n_{N-1,j-1}+(2\alpha-1)u^n_{N-1,j} \\ f_2(j\Delta y,(n+1/2)\Delta t) \end{bmatrix} \tag{8}$$

而当 $j=1$ 或 N 时，由(7)可知此时 $u^*_{i,1},u^*_{i,N},u^n_{i,N},u^n_{i,N}$ 均已知。由此，即得到第一步求解的 $u^*_{i,j}$ 矩阵。

第二步求解 同理，对于每一个 $i=2,3,\cdots,N-1$ 时，可以得到如下所示线性方程组：

$$\begin{bmatrix} 1 & 0 & & & & & \\ \alpha & -(2\alpha+1) & \alpha & & & & \\ & \alpha & -(2\alpha+1) & \alpha & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \alpha & -(2\alpha+1) & \alpha & \\ & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} u^{n+1}_{i,1} \\ u^{n+1}_{i,2} \\ u^{n+1}_{i,3} \\ \vdots \\ u^{n+1}_{i,N-1} \\ u^{n+1}_{i,N} \end{bmatrix} = \begin{bmatrix} g_1(i\Delta x,(n+1)\Delta t) \\ -\alpha u^*_{i+1,2}-\alpha u^*_{i-1,2}+(2\alpha-1)u^*_{i,2} \\ -\alpha u^*_{i+1,3}-\alpha u^*_{i-1,3}+(2\alpha-1)u^*_{i,3} \\ \vdots \\ -\alpha u^*_{i+1,N-1}-\alpha u^*_{i-1,N-1}+(2\alpha-1)u^*_{i,N-1} \\ g_2(i\Delta x,(n+1)\Delta t) \end{bmatrix} \tag{9}$$

同样地，当 $i=1$ 或 N 时，由(7)可知此时 $u^{n+1}_{1,j},u^{n+1}_{N,j}$ 均已知。由此，即得到第二步求解的 $u^{n+1}_{i,j}$ 矩阵，即下一个时间步的结果。

3 程序编写

3.1 三对角矩阵的求解

以下展示求解三对角矩阵的函数

```
function [sol] = catching(A,b)
%计算beta
[N,~] = size(b');
beta(1) = A(1,2)/A(1,1);
for n = 2:N-1
    beta(n) = A(n,n+1)/(A(n,n) - A(n,n-1)*beta(n-1));
end
%求解Ly = b
y(1) = b(1)/A(1,1);
for n = 2:N
    y(n) = (b(n)-A(n,n-1)*y(n-1))/(A(n,n)-A(n,n-1)*beta(n-1));
end

%求解Ux = y
sol(N) = y(N);
for n = 1:N-1
    sol(N-n) = y(N-n) - beta(N-n)*sol(N-n+1);
end
end
```

3.2 数据存储

利用 Matlab 提供的结构体数组，可以方便地存储时间步结果以及对应的参数。结构体 diff 中包含 4 个字段：dl, dt, RES 和 maxerror。其中 dl 为均匀网格的尺寸，dt 是时间步长，RES 则存储每一个时间步的结果，maxerror 则存储对应于每个时间步的最大误差绝对值。故程序开始计算前先进行了如下初值定义：

```
%常数定义
diff(1).dt = 0.02;
diff(1).dl = 0.1;
diff(1).RES = {};
diff(1).maxerror = 0;
```

```
diff(2).dt = 0.02;
diff(2).dl = 0.05;
diff(3).dt = 0.02;
diff(3).dl = 0.025;
diff(4).dt = 0.005;
diff(4).dl = 0.05;
diff(5).dt = 0.01;
diff(5).dl = 0.05;
diff(6).dt = 0.02;
diff(6).dl = 0.05;
diff(7).dt = 0.04;
diff(7).dl = 0.05;
global sigma
sigma = 1;
global T
T = 1;%计算时间域
global L
L = 1;%计算空间域
%常数定义完毕
```

而对于数据存贮，程序则在每一个时间步计算完成后进行如下处理：

```
diff(z).RES{tn} = u;
REALRES{tn} = percise(dl0,tn*dt0);
ERROR{tn} = abs(u - REALRES{tn});
```

其中 dl0 和 tn 等参数详见 3.4节中代码的注释。

3.3 函数定义部分

除了三对角矩阵的求解函数外，为了使程序简单且更具可移植性，程序将边界条件以及精确解等编写为函数在主程序中调用，如下所示：

```
function [res] = percise(dl,t)%微分方程精确解
global sigma;
global T;
mid = -sigma*pi^2*t/2;
N = T/dl + 1;
```

```
for i = 1 : N
    for j = 1:N
midx = pi*(i-1)*dl/2;
midy = pi*(j-1)*dl/2;
res(i,j) = 20+80*((j-1)*dl - exp(mid)*sin(midx)*sin(midy));
    end
end
end

function [res] = f1(y0,~)%x = 0 处的边界条件
res = 20 + 80*y0;
end

function [res] = f2(y0,t)% x = 1处的边界条件
global sigma;
mid = -sigma*(pi^2)*t/2;
res = 20 + 80*(y0 - exp(mid)*sin(pi*y0/2));
end

function [res] = g1(~,~)% y = 0 处的边界条件
res = 20;
end

function [res] = g2(x0,t)
global sigma;
mid = -sigma*pi^2*t/2;
res = 20 + 80*(1 - exp(mid)*sin(pi*x0/2));
end
```

3.4 程序主体部分

代码如下：

```
for z = 1:7%7为总的网格剖分情况数
    clear A;
    clear b;
```

```
clear u;
clear u_star;
dl0 = diff(z).dl;
dt0 = diff(z).dt;
alpha = sigma*dt0/(2*(dl0)^2);
N = L/dl0 + 1;
Nt = T/dt0;
%生成刚度矩阵
A(1,1) = 1;
A(N,N) = 1;
for i = 2:N-1
    A(i,i-1) = alpha;
    A(i,i) =-(2*alpha+1);
    A(i,i+1) = alpha;
end
u = percise(dl0,0);
u_star = zeros(N,N);
for tn = 1:Nt
    %开始求解u*
        %当j=1
        for n = 1:N
            u_star(n,1) = g1((n-1)*dl0,tn*dt0);
        end
        %当j = 2~N-1
        for j = 2:N-1
            b(1) = f1((j-1)*dl0,(tn-1/2)*dt0);
            for l = 2:N-1
                b(l) = -alpha*(u(l,j+1)+u(l,j-1) ...
                -2*u(l,j))-u(l,j);
            end
            b(N) = f2((j-1)*dl0,(tn-1/2)*dt0);
            u_star(:,j) = catching(A,b);
        end
        %当j = N
        for n = 1:N
```

```
            u_star(n,N) = g2((n-1)*dl0,tn*dt0);
        end
        %u*求解完成
    %开始求解u^{n+1}
        %当i = 1
        for n = 1:N
            u(1,n) = f1((n-1)*dl0,tn*dt0);
        end
        %当i=2~N-1
        for i = 2:N-1
            b(1) = g1((i-1)*dl0,(tn)*dt0);
            for l = 2:N-1
                b(l) = -alpha*(u_star(i+1,l)+ ...
                    u_star(i-1,l)-2*u_star(i,l)) ...
                    -u_star(i,l);
            end
            b(N) = g2((i-1)*dl0,(tn)*dt0);
            u(i,:) = catching(A,b);
        end
        %当i = N
        for n = 1:N
            u(N,n) = f2((n-1)*dl0,tn*dt0);
        end
    %数据存储部分和作图部分略去
    end
end
```

4 结果分析

4.1 数值误差与离散步长的关系

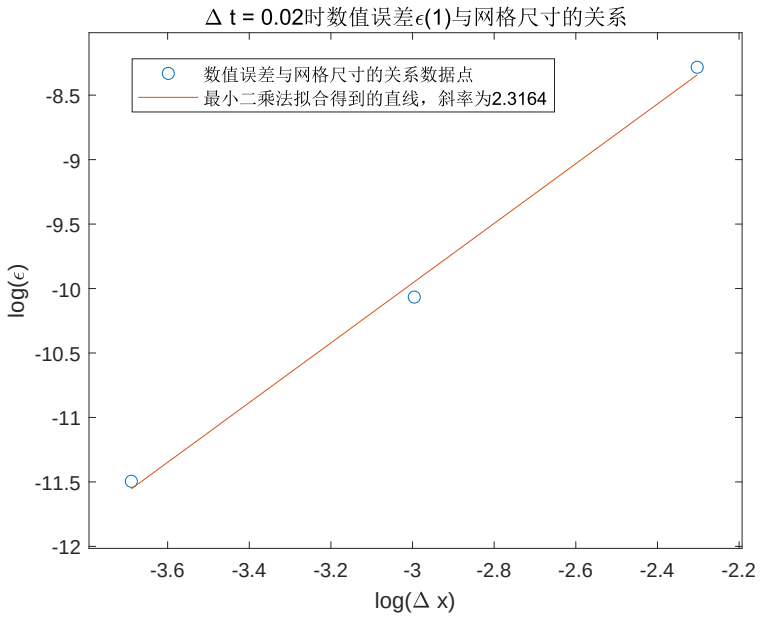

图 1: 数值误差与网格间距的关系（$\Delta t = 0.02$）

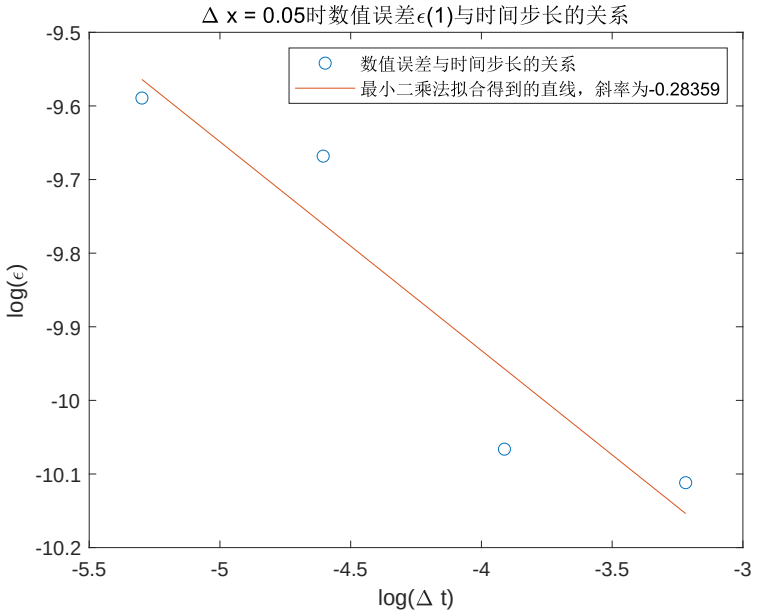

图 2: 数值误差与时间步长的关系（$\Delta x = 0.05$）

由图 1可知，当时间步长 $\Delta t = 0.02$ 时，数值误差与网格间距二者的对数值均大致呈线性关系，图 1中由最小二乘法求得的斜率约为 $2.31 > 0$，即 $\epsilon(1) \sim \Delta x^{2.31}$，说明此格式收敛，且精度约为二阶。

由图 2可知，当网格间距 $\Delta l = 0.05$ 时，数值误差与时间步长二者的对数值均大致呈线性关系，图 2中由最小二乘法求得的斜率约为 $-0.28 < 0$，即 $\epsilon(1) \sim \Delta t^{-0.28}$，这与 ADI 格式的收敛性相悖。尝试更换网格间距为 $\Delta l = 0.01$ 时得到如下图所示结果：

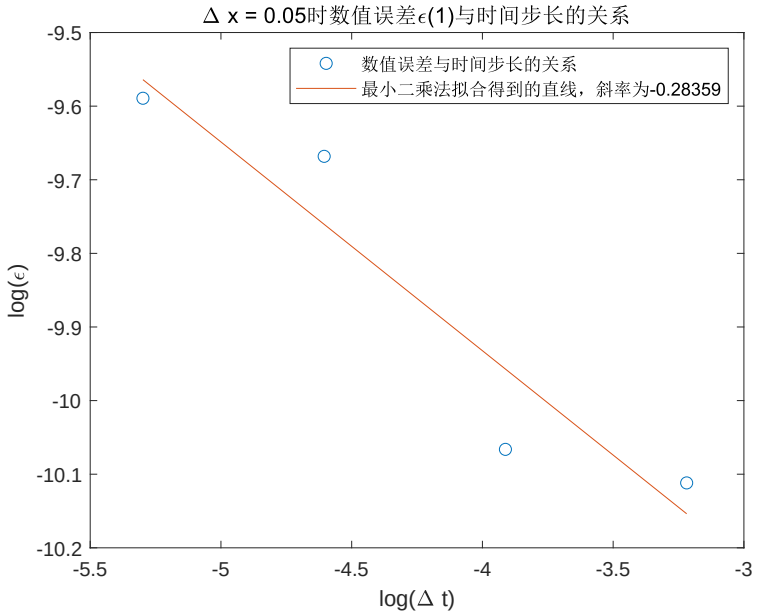

图 3: 数值误差与时间步长的关系（$\Delta x = 0.01$）

此时由最小二乘法求得的斜率约为 2.25，与 $\Delta l = 0.05$ 时的结果不一致。为了探究出现这种情况的原因，再取 $\Delta x =, \frac{1}{40}, \frac{1}{60}, \frac{1}{80}, \frac{1}{100}$ 画出如下所示数值误差与时间步长的关系图：

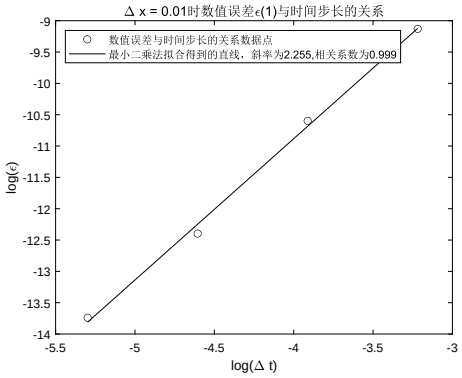

图 4: $\Delta x = \Delta y = 1/100$

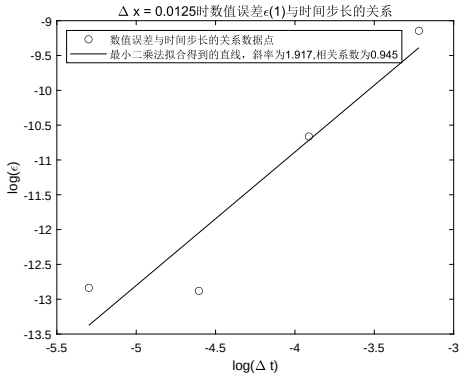

图 5: $\Delta x = \Delta y = 1/80$

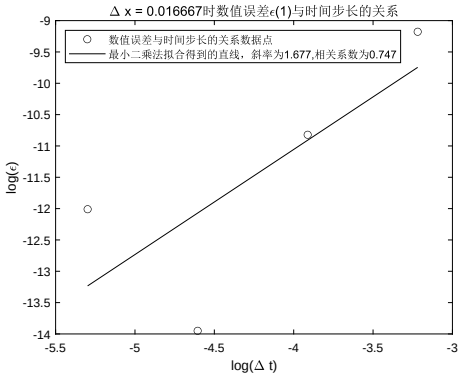

图 6: $\Delta x = \Delta y = 1/60$

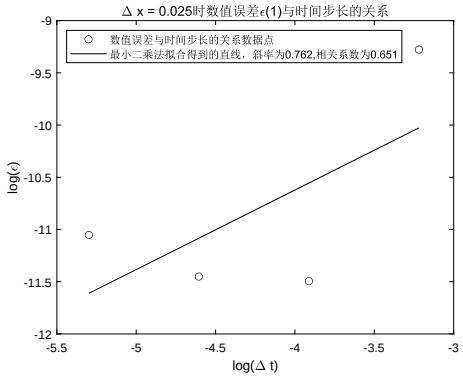

图 7: $\Delta x = \Delta y = 1/40$

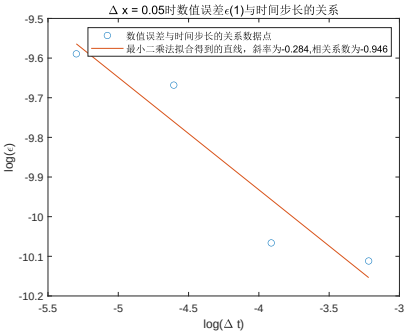

图 8: $\Delta x = \Delta y = 1/20$

由上述五个图像可以看出，当网格间距增大时，数值误差与时间的关系的线性性越来越差（相关系数越来越小），同时斜率也越来越低。故造成如图 2所示斜率为负数（即时间步长越小误差反而越大）的原因可能是此时网格间距较大，从而使数值误差主要由网格间距主导，与时间步长关系较弱。

4.2 计算误差与时间的关系

采用半对数坐标系，纵坐标计算误差 E_h 为对数刻度，横坐标计算时间 t 为正常刻度，针对题中六种不同参数组合得到下列图像：

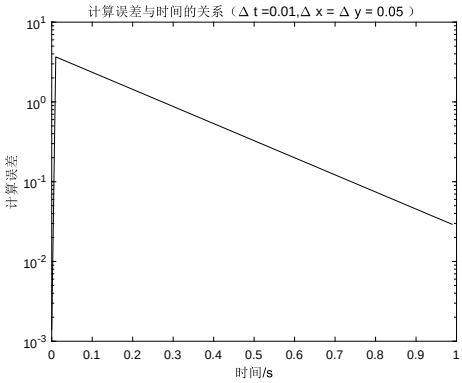

图 9: $\Delta t = 0.01, \Delta x = \Delta y = 0.05$

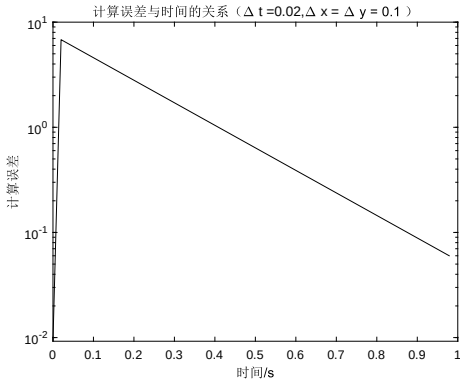

图 10: $\Delta t = 0.02, \Delta x = \Delta y = 0.1$

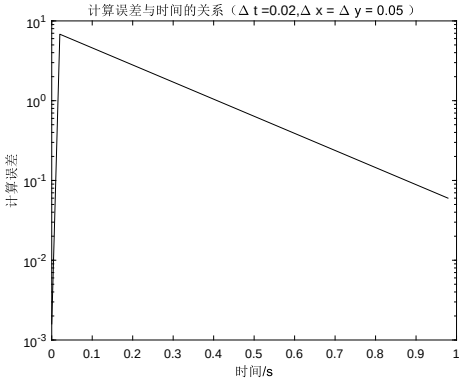

图 11: $\Delta t = 0.02, \Delta x = \Delta y = 0.05$

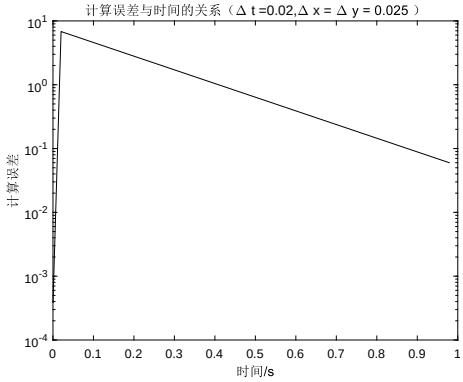

图 12: $\Delta t = 0.02, \Delta x = \Delta y = 0.025$

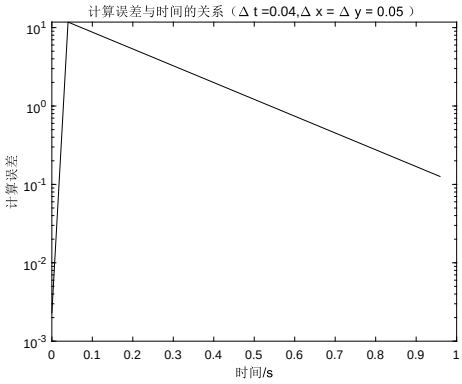

图 13: $\Delta t = 0.04, \Delta x = \Delta y = 0.05$

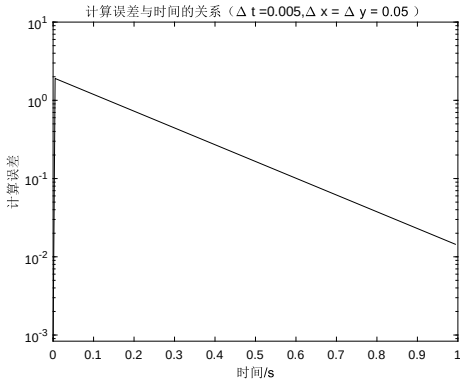

图 14: $\Delta t = 0.005, \Delta x = \Delta y = 0.05$

由上述图像可以看出，$\log(E_h)$ 与时间呈线性关系，且随着计算时间的增加，误差也随之减小。