

作业 3: 数值求解热传导方程

谢文进

2021 年 5 月 23 日

1 数值求解热传导方程

1.1 最简显格式

对于如下热传导方程:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} & 0 < t < 0.03, 0 < x < 1 \\ u(x, 0) = \sin(4\pi x) & 0 \leq x \leq 1 \\ u(0, t) = 0 & 0 \leq t \leq 0.03 \\ u(1, t) = 0 & 0 \leq t \leq 0.03 \end{cases} \quad (1)$$

该方程的精确解为 $u(t, x) = e^{-(4\pi)^2 t} \sin(4\pi x), 0 \leq t \leq 0.03, 0 \leq x \leq 1$.

首先对时间及空间进行划分, 将时间划分为 $m - 1$ 份, 将空间划分为 $n - 1$ 份, 则 $\tau = \frac{0.03}{m-1}, h = \frac{1}{n-1}$ 。根据最简显格式可以得到:

$$u_j^{i+1} = \frac{\tau}{h^2} (u_{j+1}^i - 2u_j^i + u_{j-1}^i) + u_j^i, \quad (2)$$

其中 $i = 0, 1, \dots, m - 1, j = 0, 1, \dots, n - 1$ 。

1.2 编程实现

根据理论推导, 用 python 编写程序如下:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sun May 23 11:16:13 2021
4
```

```
5 @author: XieWenjin
6 """
7
8 import math
9 import numpy as np
10 from matplotlib import pyplot as plt
11 from mpl_toolkits.mplot3d import Axes3D
12
13 t = 0.03 # 时间范围
14 x = 1.0 # 空间范围
15 m = input("请输入m: ")
16 m = int(m)
17 n = input("请输入n: ")
18 n = int(n)
19 # m = 320 # 时间方向分为320个格子
20 # n = 64 # 空间方向的格子数
21 dt = t / (m - 1) # 时间步长
22 dx = x / (n - 1) # 空间步长
23
24 def generate_u(m,n):
25     u = np.zeros([m,n])
26     # 边界条件
27     for j in range(n):
28         u[0,j] = math.sin(4 * math.pi * j * dx)
29     for i in range(m):
30         u[i,0] = 0
31         u[i,-1] = 0
32
33     # 差分法
34     for i in range(m - 1):
35         for j in range(1,n - 1):
36             u[i+1, j] = dt * (u[i, j + 1] + u[i, j - 1] - 2 * u[i, j])
37             / dx ** 2 + u[i, j]
38
39     return u
40
41 def drawing(X,Y,Z):
42     fig = plt.figure()
43     ax = Axes3D(fig)
44     ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap='rainbow')
```

```

43     plt.show()
44
45     def error(u,u_exact):
46         err = abs(u - u_exact)
47         return max(map(max, err))
48
49     X = np.arange(0, t + dt, dt) # remark:t+dt,not t
50     Y = np.arange(0, x + dx, dx)
51     X, Y = np.meshgrid(X, Y)
52     u_exact = np.exp(-(4*np.pi)**2*X)*np.sin(4*np.pi*Y)
53
54     u = generate_u(m,n)
55     u = np.transpose(u) # 注意这里是转置,而不是np.reshape(u,(n,m))
56
57     # print(m,n)
58     print(error(u,u_exact))
59
60     drawing(X,Y,u) # 数值解
61     # drawing(X,Y,u_exact) # 精确解
62     # drawing(X,Y,abs(u-u_exact)) # 数值解与精确解之差的绝对值

```

1.3 结果分析

当取 $m = 320, n = 64$ 时, 得到数值解 $[u]$ 如图 1(a), 精确解 u 如图 1(b)。

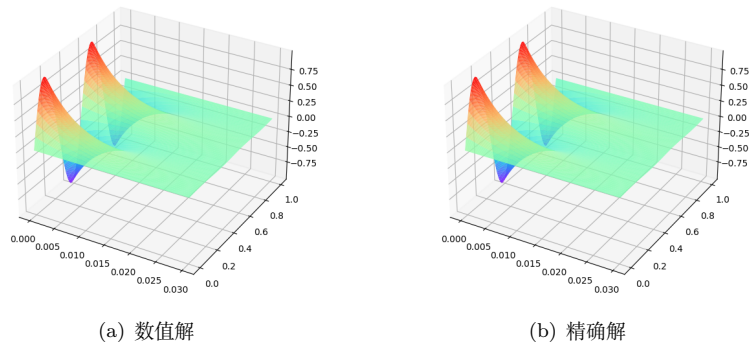


图 1: $m = 320, n = 64$ 结果图

精确解与数值解之差的绝对值 $\|u - [u]\|$ 如图 2, 计算得到误差为 $\|u - [u]\|_c = 0.0015189395174771136$ 。

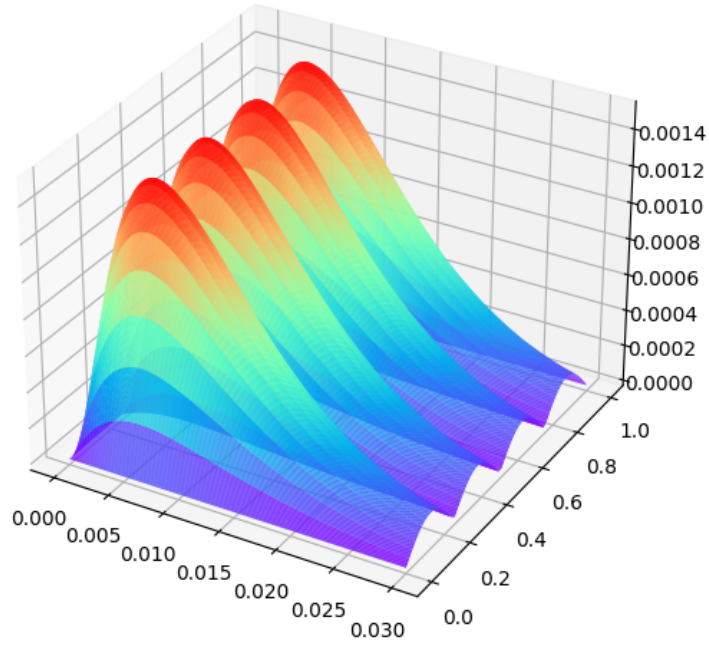


图 2: $\|u - [u]\|$ 结果图

当取不同的 τ, h 时, 计算得到的误差如表所示。

表 1: 不同 τ, h 的误差表

τ	h	误差 e	e_i/e_{i+1}
$\frac{1}{10}$	$\frac{1}{10}$	0.0428079643162558	
$\frac{1}{40}$	$\frac{1}{20}$	0.00951825176096948	4.4975
$\frac{1}{160}$	$\frac{1}{40}$	0.00244056613219328	3.9000
$\frac{1}{640}$	$\frac{1}{80}$	0.000606385251482932	4.0248
$\frac{1}{2560}$	$\frac{1}{160}$	0.000151362159712509	4.0062

我们知道最简显格式的误差为 $\|[u]^k - u^k\| = O(\tau + h^2)$, 设 $e_i = C \times$

$(\tau_i + h_i^2)$, 取 $\tau_{i+1} = \frac{1}{4}\tau_i, h_{i+1} = \frac{1}{2}h_i$, 可以得到 $\frac{e_i}{e_{i+1}} = 4$, 与表 1 结果相符, 从而也验证了最简显格式的误差阶数为 $O(\tau + h^2)$.