抛物型方程的前向 Euler 显式差分解法

陶睿 13020031105
（中国海洋大学 信息科学与工程学院 山东省 青岛市 266100）

摘要：使用python语言实现了对抛物型方程的前向Euler显式差分格式求数值解，分析了显格式的稳定性以及最大误差的变化规律。通过亲自实现，对差分方法及其稳定性有了更加深刻的理解。

关键词：抛物型方程　前向Euler　显式格式　稳定性

0 引言

抛物型方程最简单的形式是一维热传导方程。
本文考虑的一维非齐次热传导方程的定解问题：

$$\frac{\partial u}{\partial t}-a\frac{\partial^2 u}{\partial x^2}=f(x,t), \qquad 0<x<l, 0<t\le T,$$

$$u(x,0)=\phi(x), \qquad 0\le x\le l,$$

$$u(0,t)=\alpha(t), \qquad u(1,t)=\beta(t), \qquad 0<t\le T.$$

其中 a 为正常数， $f(x,t),\varphi(x),\alpha(t),\beta(t)$ 为已知函数， $\varphi(0)=\alpha(0),\varphi(1)=\beta(0).$

目前常用的求解热传导方程的差分格式有前向 Euler 差分格式、向后 Euler 差分格式、Crank-Nicolson 格式、Richardson 格式[1,2]. 本文将给出前向 Euler 格式和紧差分格式，并给出其截断误差和数值例子.

1 物理背景

热传导是由于物体内部温度分布不均匀, 热量要从物体内温度较高的点流向温度较低的点处. 以函数 $u(x,y,z,t)$ 表示物体在 t 时刻, $M=M(x,y)$ 处的温度, 并假设 $u(x,y,z)$ 关于 x,y,z 具有二阶连续偏导数, 关于 t 具有一阶连续偏导数.

$k=k(x,y,z)$ 是物体在 $M(x,y,z)$ 处的热传导系数, 取正值. 设物体的比热容为 $c=c(x,y,z)$, 密度为 $\rho(x,y,z)$. 根据 *Fourier* 热传导定律, 热量守恒定律以及 *Gauss* 公式得

$$c\rho\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}\left(kx\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial u}{\partial y}\right)+\frac{\partial}{\partial z}\left(k\frac{\partial u}{\partial z}\right),$$

如果物体是均匀的,此时k,c以及ρ均为常数.令$a^2=\dfrac{k}{c\rho}$,上式方程化为

$$u_t=a^2\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)=a^2\Delta u,$$

若考虑物体内有热源,其热源密度函数为$F=F\left(x,y,z\right)$,则有热源的热传导方程为

$$u_t=a^2\Delta u+f\left(x,y,z,t\right),$$

其中$f=\dfrac{F}{c\rho}$.

2　网格剖分

取空间步长$h=l/N$和时间步长$\tau=T/M$，其中N,M都是正整数. 用两族平行直线$x_j=jh(j=0,1\cdots,N)$和$t_k=k\tau(k=0,1,\cdots,M)$将矩形域$\overline{G}=\{0\le x\le l,0\le t\le T\}$分割成矩形网格，网格节点为$(x_j,t_k)$. 记$u_j^k=u(x_j,t_k)$.

3 显格式的建立和求解

定义$\Omega_{h\tau}$上的网格函数

$$U=\{U_i^k\,|0\le i\le m,0\le k\le n\},$$

其中

$$U_i^k=u(x_i,t_k),\qquad 0\le i\le m,\qquad 0\le k\le n-1.$$

在结点处考虑微分方程(3.1-1),有

$$\frac{\partial u}{\partial t}(x_i,t_k)-a\frac{\partial^2 u}{\partial x^2}(x_i,t_k)=f(x_i,t_k),\qquad 1\le i\le m-1,\ \ 0\le k\le n-1. \tag{3.2}$$

得到

$$\frac{\partial^2 u}{\partial x^2}(x_i,t_k)=\frac{1}{h^2}[u(x_{i-1},t_k)-2u(x_i,t_k)+u(x_{i+1},t_k)]-\frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(\xi_{ik},t_k)$$

记 $r=a\tau/h^2$，称r为步长比。差分格式可写为

$$u_i^{k+1}=(1-2r)u_i^k+r(u_{i-1}^k+u_{i+1}^k)+\tau f(x_i,t_k), \qquad 0\le i\le m-1,\ 0\le k\le n-1.$$

上式表明第 k+1 层上的值由第 k 层上的值显示表示出来。若已知第 k 层的值 $\{u_i^k \mid 0\le i\le m\}$，则由上式就可直接得到第 k+1 层上的值 $\{u_i^{k+1} \mid 0\le i\le m\}$。有时也称为古典显格式。可把古典显格式写成矩阵形式

$$\begin{pmatrix} u_1^{k+1} \\ u_2^{k+1} \\ \\ u_{m-2}^{k+1} \\ u_{m-1}^{k+1} \end{pmatrix}=\begin{pmatrix} 1-2r & r & & & \\ r & 1-2r & r & & \\ & & & & \\ & & r & 1-2r & r \\ & & & r & 1-2r \end{pmatrix}\begin{pmatrix} u_1^k \\ u_2^k \\ \\ u_{m-2}^k \\ u_{m-1}^k \end{pmatrix}+\begin{pmatrix} \tau f(x_1,t_k)+ru_0^k \\ \tau f(x_2,t_k) \\ \\ \tau f(x_{m-2},t_k) \\ \tau f(x_{m-1},t_k)+ru_m^k \end{pmatrix}$$

4　显格式的稳定性

由计算方法[3]可知，在 r<=1/2 时，显格式是稳定的，误差收敛。在 r>1/2 时，显格式不稳定。实际计算时选取步长比必须满足 $r\le 1/2$，即 $a\tau/h^2\le 1/2$

5　一个数值例子

计算定解问题

$$\frac{\partial u}{\partial t}-\frac{\partial^2 u}{\partial x^2}=0, 0<x<1, 0<t\le 1$$

$$u(x,0)=e^x, \qquad 0\le x\le 1$$

$$u(0,t)=e^t, u(1,t)=e^{1+t}, 0<t\le 1$$

上述定解问题的精确解为 $u(x,t)=e^{x+t}$.

5.1 部分节点处数值解、精确解和误差的绝对值

部分节点处数值解、精确解和误差的绝对值 $(h=1/10,\tau=1/200)$

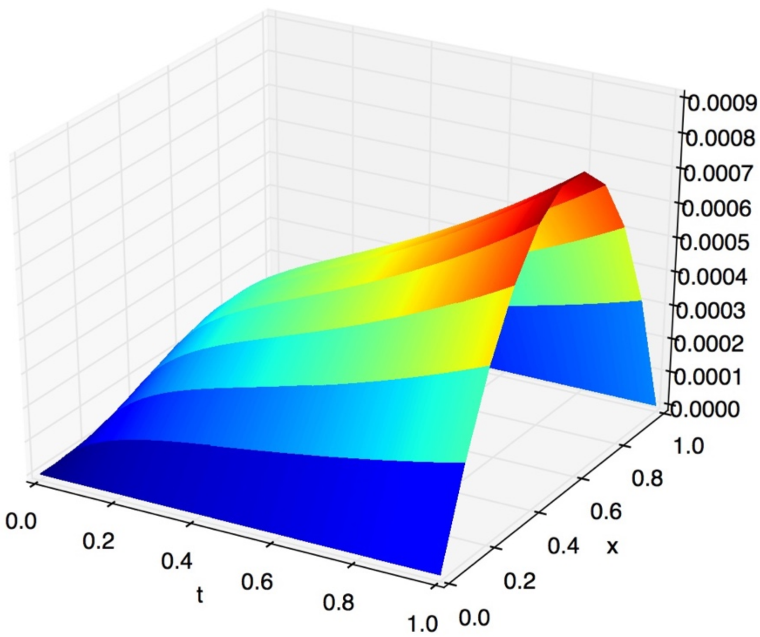
k	$k(x,t)$	数值解	精确解	|精确解-数值解|
20	(0.5,0.1)	1.8219e+000	1.8221e+000	2.3008e-004
40	(0.5,0.2)	2.0134e+000	2.0138e+000	3.4361e-004
60	(0.5,0.3)	2.2251e+000	2.2255e+000	4.1249e-004
80	(0.5,0.4)	2.4591e+000	2.4596e+000	4.6788e-004
100	(0.5,0.5)	2.7178e+000	2.7183e+000	5.2148e-004
120	(0.5,0.6)	3.0036e+000	3.0042e+000	5.7794e-004

140	(0.5,0.7)	3.3195e+000	3.3201e+000	6.3932e-004
160	(0.5,0.8)	3.6686e+000	3.6693e+000	7.0677e-004
180	(0.5,0.9)	4.0544e+000	4.0552e+000	7.8118e-004
200	(0.5,0.10)	4.4808e+000	4.4817e+000	8.6337e-004

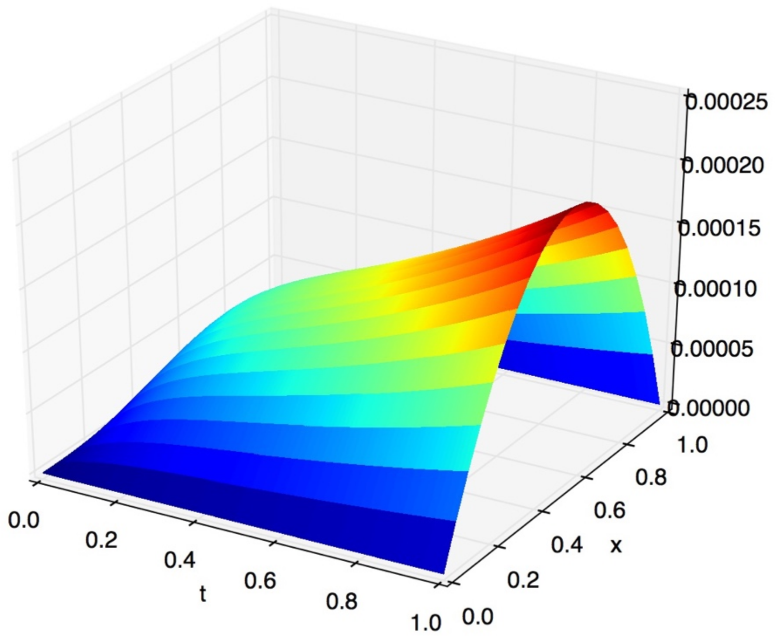
取不同不长时数值解的最大误差$(r=1/2)$

h	τ	$E_\infty(h,\tau)$	$E_\infty(2h,4\tau)/E_\infty(h,\tau)$
1/10	1/200	8.6337e-004	*
1/20	1/800	2.1748e-004	3.9699e+000
1/30	1/3200	5.4366e-005	4.0003e+000
1/40	1/12800	1.3591e-005	4.0001e+000

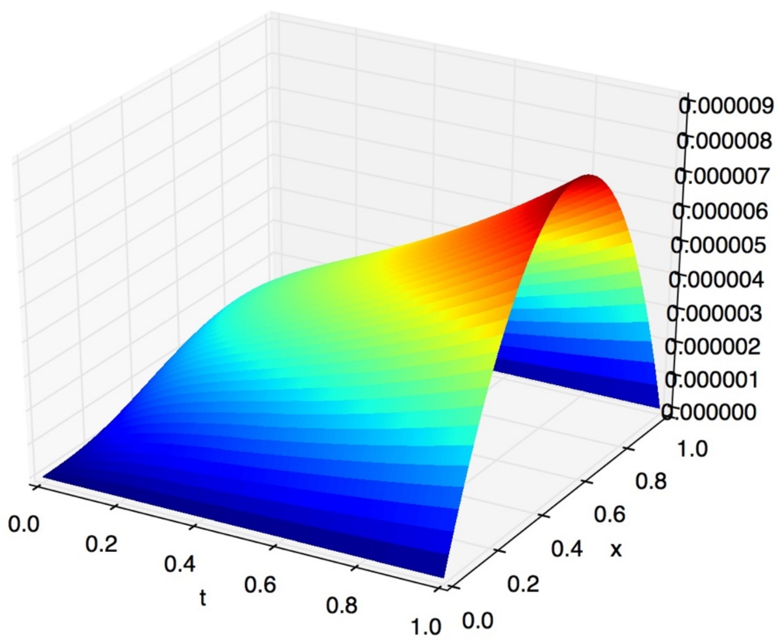
5.2 误差曲面图



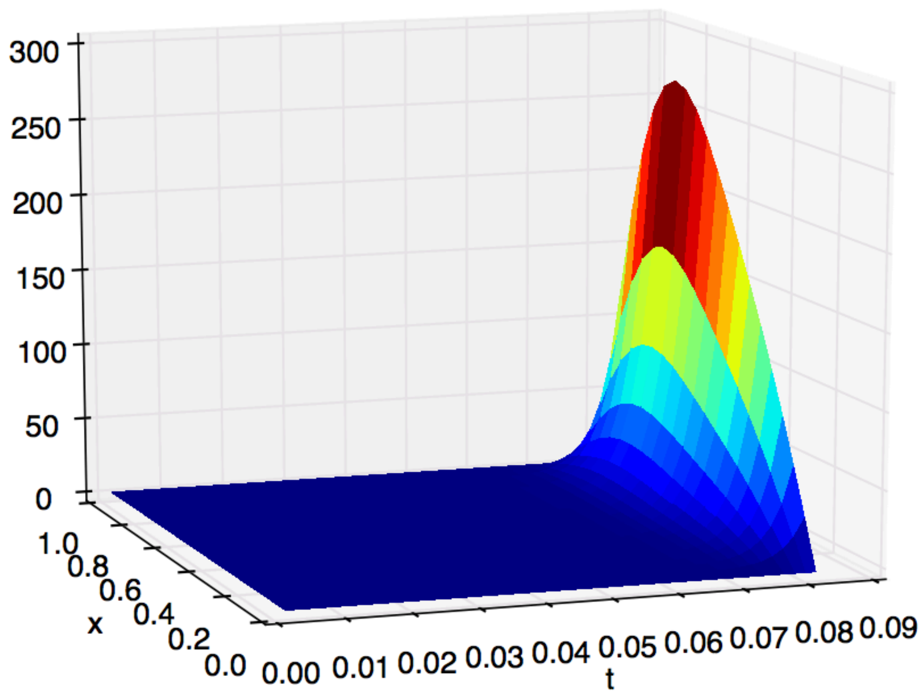
$h=\frac{1}{10},\tau=\frac{1}{200}$的误差曲面图（$r=1/2$）



$h=\frac{1}{20},\tau=\frac{1}{800}$的误差曲面图（$r=1/2$）



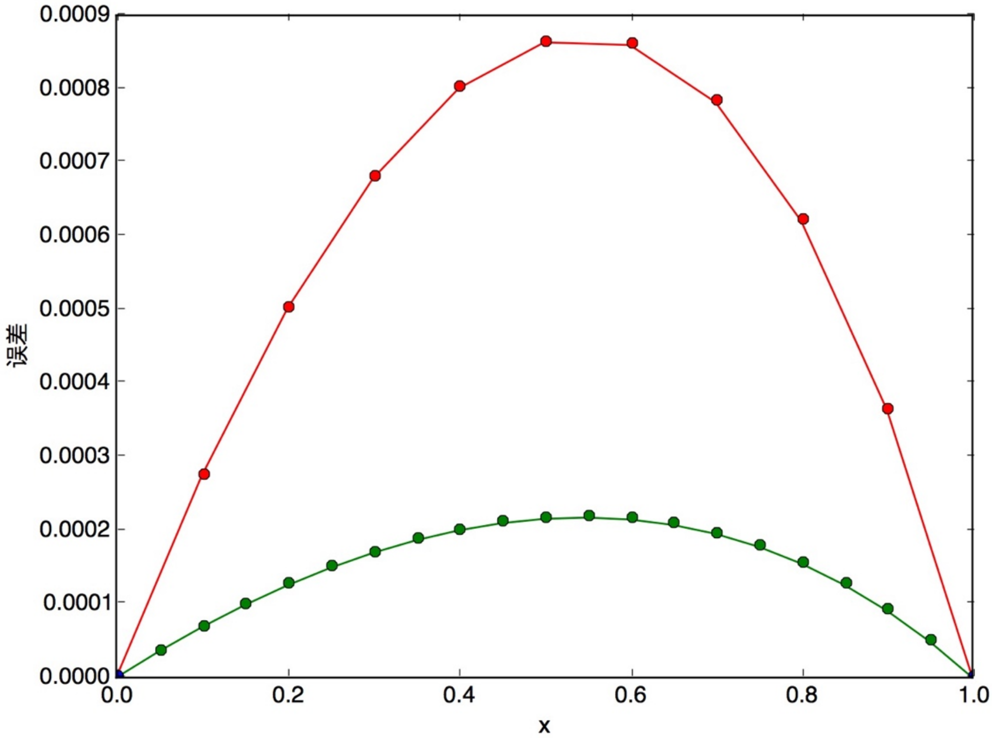
$h=\frac{1}{50},\tau=\frac{1}{10000}$的误差曲面图（$r=1/2$）



$h=\frac{1}{20},\tau=\frac{1}{600}$的误差曲面图（$r=2/3$）

可以看出，r=1/2 时，差分格式稳定性好。r>1/2 时，差分格式不稳定，误差迅速增长。

5.3 误差曲线图



$t=1$时的误差曲面图（$r=1/2$）绿色$h=\frac{1}{10},\tau=\frac{1}{200}$ 红色$h=\frac{1}{20},\tau=\frac{1}{800}$,

6 总结

本文采用差分格式来求解抛物型方程.差分格式采用二层共三个点,条件稳定显格式,当$r>\dfrac{1}{2}$时误差随着r无限增长。其稳定条件为$r\le\dfrac{1}{2}$,因此我们给出了当$r=\dfrac{1}{2}$时的最大误差.可以得出，在 r=1/2 不变时，当距离步长变为 2 倍，时间步长相应变为 4 倍时，最大误差也扩大了 4 倍。

隐格式的程序求解与显格式类似，不再给出。隐格式相对于显格式的优点在于无条件稳定，即无论 r 的取值，隐格式差分均为稳定的。

参 考 文 献

[1] 孙志忠. 偏微分方程数值解法[M]. 北京：科学出版社，2005.
[2] 李荣华. 偏微分方程数值解法[M]. 北京：高等教育出版社，2005.
[3] 徐萃薇,孙绳武. 计算方法引论[M]. 北京：高等教育出版社，2007.

附录 python 程序代码（计算及绘图）

```python
# -*- coding:utf-8 -*-
__author__ = 'Tao Rui'
__date__   = '2016-1-10'
import numpy as np
import scipy.linalg as lina
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter

n = input('n = ') # 空间剖分数
m = input('m = ') # 时间剖分数
x = np.linspace(0, 1, n+1)
t = np.linspace(0, 1, m+1)
r = n * n / float(m) # 网比
A = np.diagflat(np.ones((1, n - 1), dtype = 'float64') * (1 - 2 * r))\
    + np.diagflat(np.ones((1, n - 2), dtype = 'float64') * r, -1)\
    + np.diagflat(np.ones((1, n - 2), dtype = 'float64') * r, 1)
#---------------------精确解求解--------------------
u = np.zeros((n + 1, m + 1), dtype = 'float64')
for i in range(0, n + 1):
    for j in range(0, m + 1):
        u[i,j] = np.exp(x[i] + t[j])
#--------------------初值条件求解--------------------
u1 = np.zeros((n + 1, m + 1), dtype = 'float64')

for i in range(0, n + 1):
    u1[i, 0] = np.exp(x[i])
for j in range(0, m + 1):
    u1[0, j] = np.exp(t[j])
for j in range(0, m + 1):
    u1[n, j] = np.exp(1 + t[j])
#---------------------数值解求解--------------------
for j in range(0, m):
    f = np.zeros((n));
    f[1] = r*u1[0,j];
    f[n - 1] = r*u1[n,j]
    u1[1:n, j+1] = np.dot(A, u1[1:n, j]) + f[1:n]
#--------------------画图------------------------
#fig = plt.figure()
#--------------------误差曲面图--------------------
#error = abs(u-u1)
```

```python
#error = error.transpose()
#ax = fig.gca(projection='3d')
#X,T = np.meshgrid(x, t)
#surf = ax.plot_surface(T[1:50,:], X[1:50,:], error[1:50,:],rstride=1, cstride=1, cmap=cm.jet,linewidth=0, antialiased=False)
#surf = ax.plot_surface(T, X, u1.transpose(),rstride=1, cstride=1, cmap=cm.jet,linewidth=0, antialiased=False)
#--------------------误差曲线图---------------------
#ax = fig.gca()
E = abs(u[:,m]-u1[:,m])
ax.plot(x,E,'bo-');
plt.xlabel('x')
plt.xticks([0,0.2,0.4,0.6,0.8,1.0])
plt.ylabel(u'误差')
#plt.title(u't=1 时误差曲线, n= %d m = %d'%(n,m))
#-------------------------------------------------
#plt.show()
#plt.savefig('1.jpg')
```