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：







其中为正常数，为已知函数，

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.以函数表示物体在时刻,处的温度,并假设关于具有二阶连续偏导数,关于具有一阶连续偏导数.是物体在处的热传导系数,取正值.设物体的比热容为,密度为.根据热传导定律,热量守恒定律以及公式得



如果物体是均匀的,此时以及均为常数.令,上式方程化为



若考虑物体内有热源,其热源密度函数为,则有热源的热传导方程为



其中.

**2 网格剖分**

取空间步长和时间步长，其中都是正整数．用两族平行直线和将矩形域分割成矩形网格，网格节点为．记．

**3 显格式的建立和求解**

定义上的网格函数



其中

在结点处考虑微分方程(3.1-1),有

 (3.2)

得到



记，称为步长比。差分格式可写为

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，则由上式就可直接得到第k+1层上的值。有时也称为古典显格式。可把古典显格式写成矩阵形式



**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，即

**5 一个数值例子**

计算定解问题







上述定解问题的精确解为.

**5.1 部分节点处数值解、精确解和误差的绝对值**

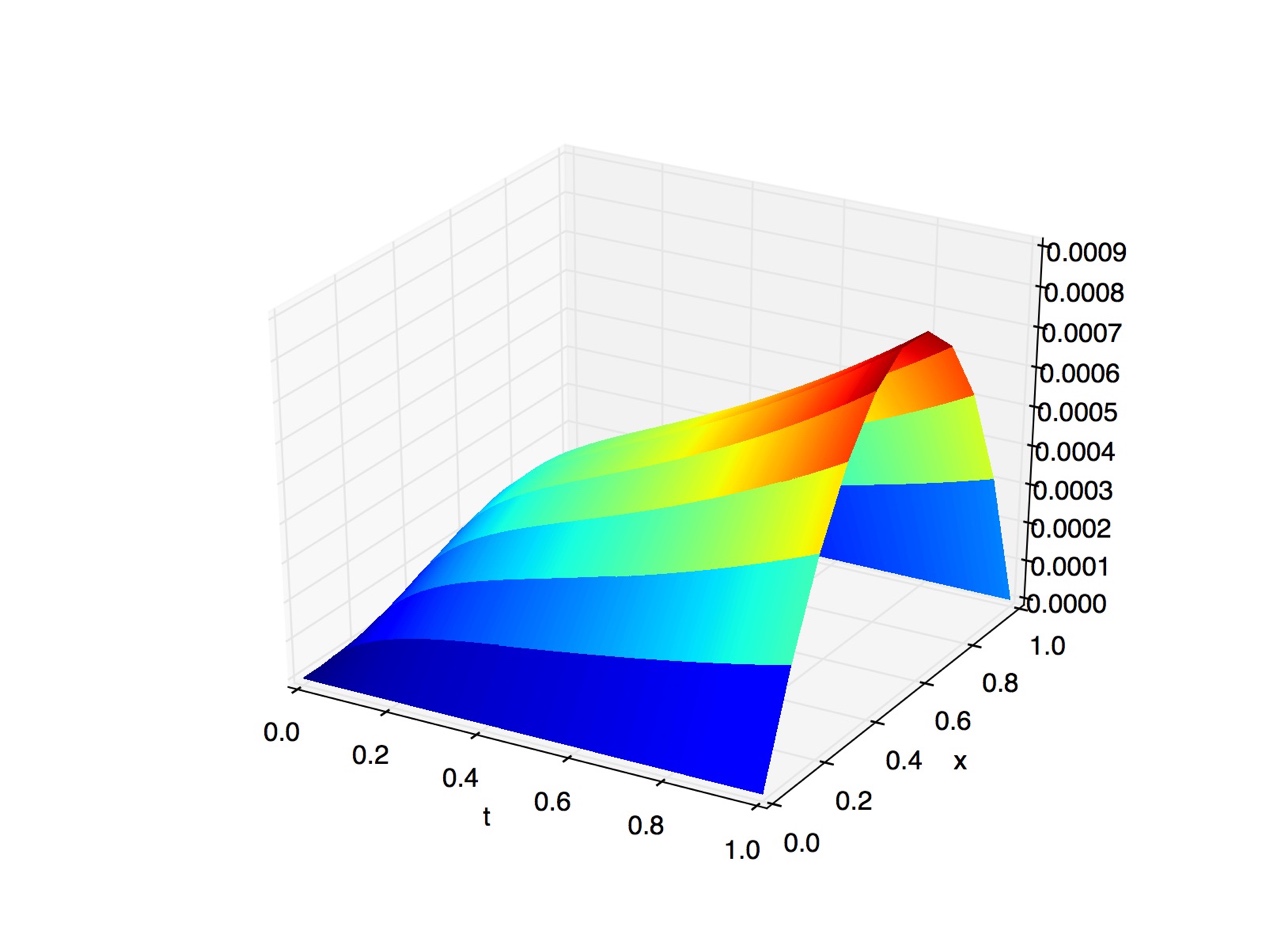
部分节点处数值解、精确解和误差的绝对值

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 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 |

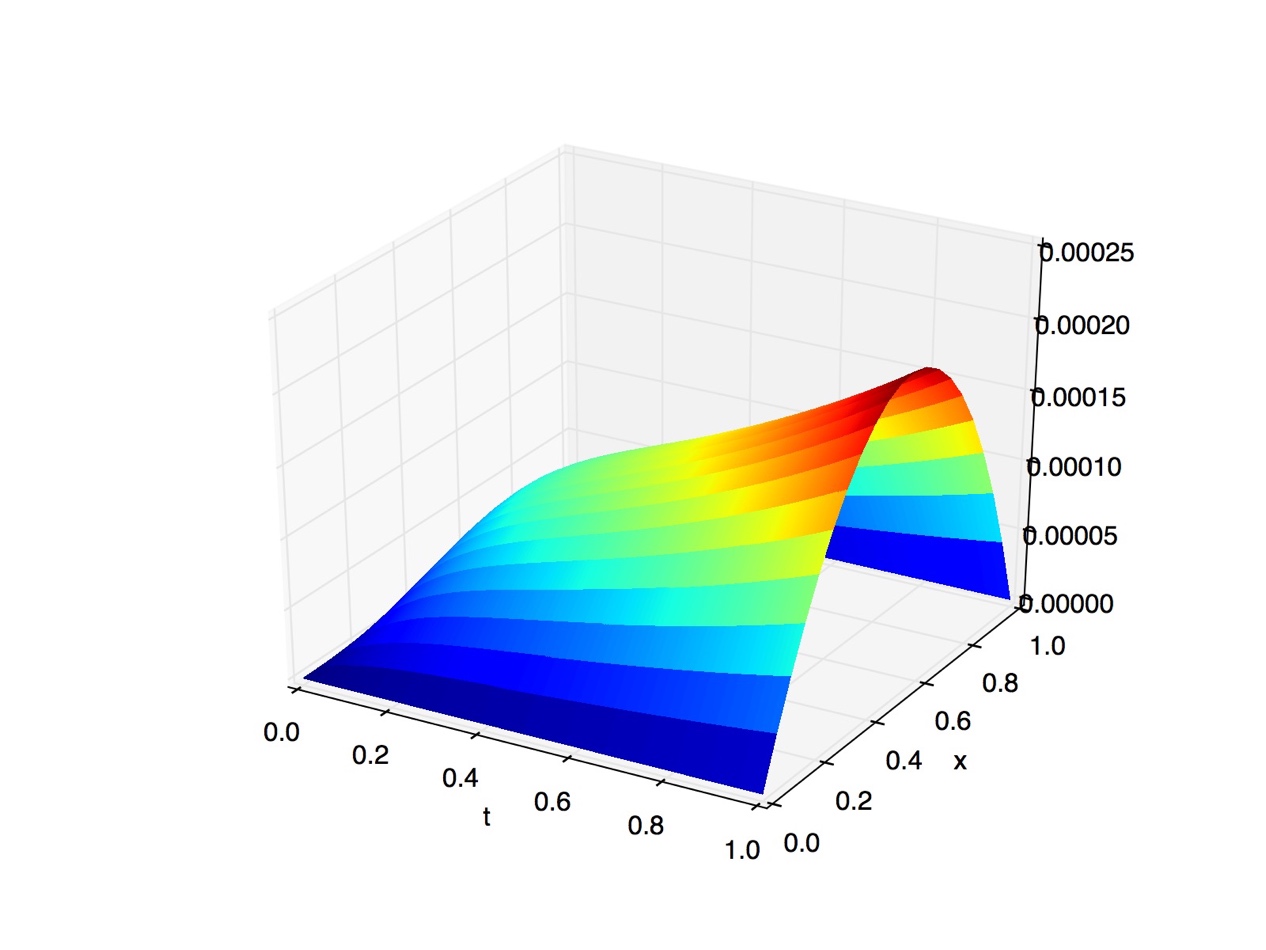
取不同不长时数值解的最大误差

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 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 |

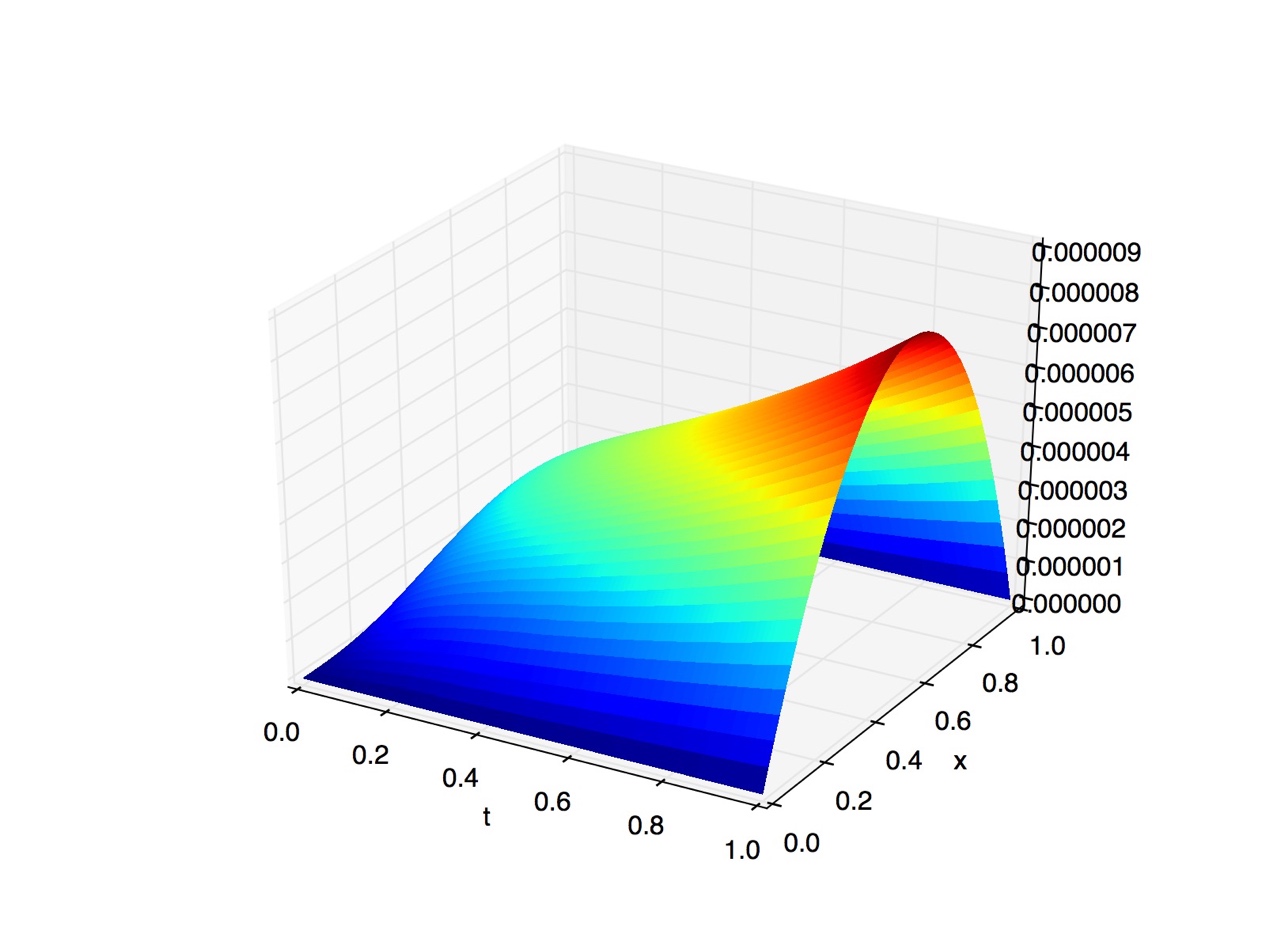
**5.2 误差曲面图**



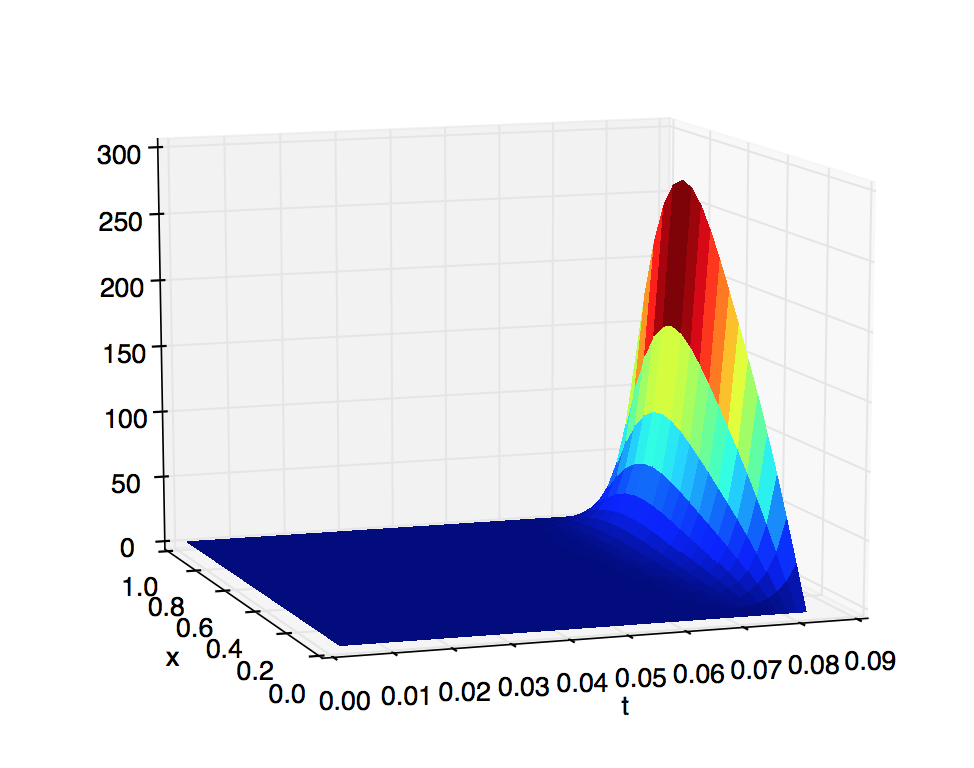
的误差曲面图（）



的误差曲面图（）



的误差曲面图（）



的误差曲面图（）

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**

时的误差曲面图（）绿色 红色

**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,当时误差随着无限增长。其稳定条件为,因此我们给出了当时的最大误差.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）

|  |  |
| --- | --- |
| 1 | # -\*- coding:utf-8 -\*- |
| 2 | \_\_author\_\_ = 'Tao Rui' |
| 3 | \_\_date\_\_   = '2016-1-10' |
| 4 | import numpy as np |
| 5 | import scipy.linalg as lina |
| 6 | import matplotlib.pyplot as plt |
| 7 | from mpl\_toolkits.mplot3d import Axes3D |
| 8 | from matplotlib import cm |
| 9 | from matplotlib.ticker import LinearLocator, FormatStrFormatter |
| 10 |  |
| 11 | n = input('n = ') # 空间剖分数 |
| 12 | m = input('m = ') # 时间剖分数 |
| 13 | x = np.linspace(0, 1, n+1) |
| 14 | t = np.linspace(0, 1, m+1) |
| 15 | r = n \* n / *float*(m) # 网比 |
| 16 | A = np.diagflat(np.ones((1, n - 1), *dtype* = 'float64') \* (1 - 2 \* r))\ |
| 17 | + np.diagflat(np.ones((1, n - 2), *dtype* = 'float64') \* r, -1)\ |
| 18 | + np.diagflat(np.ones((1, n - 2), *dtype* = 'float64') \* r, 1) |
| 19 | #---------------------精确解求解-------------------- |
| 20 | u = np.zeros((n + 1, m + 1), *dtype* = 'float64') |
| 21 | for i in range(0, n + 1): |
| 22 | for j in range(0, m + 1): |
| 23 | u[i,j] = np.exp(x[i] + t[j]) |
| 24 | #--------------------初值条件求解------------------- |
| 25 | u1 = np.zeros((n + 1, m + 1), *dtype* = 'float64') |
| 26 |  |
| 27 | for i in range(0, n + 1): |
| 28 | u1[i, 0] = np.exp(x[i]) |
| 29 | for j in range(0, m + 1): |
| 30 | u1[0, j] = np.exp(t[j]) |
| 31 | for j in range(0, m + 1): |
| 32 | u1[n, j] = np.exp(1 + t[j]) |
| 33 | #---------------------数值解求解-------------------- |
| 34 | for j in range(0, m): |
| 35 | f = np.zeros((n)); |
| 36 | f[1] = r\*u1[0,j]; |
| 37 | f[n - 1] = r\*u1[n,j] |
| 38 | u1[1:n, j+1] = np.dot(A, u1[1:n, j]) + f[1:n] |
| 39 | #--------------------画图------------------------ |
| 40 | #fig = plt.figure() |
| 41 | #--------------------误差曲面图-------------------- |
| 42 | #error = abs(u-u1) |
| 43 | #error = error.transpose() |
| 44 | #ax = fig.gca(projection='3d') |
| 45 | #X,T = np.meshgrid(x, t) |
| 46 | #surf = ax.plot\_surface(T[1:50,:], X[1:50,:], error[1:50,:],rstride=1, cstride=1, cmap=cm.jet,linewidth=0, antialiased=False) |
| 47 | #surf = ax.plot\_surface(T, X, u1.transpose(),rstride=1, cstride=1, cmap=cm.jet,linewidth=0, antialiased=False) |
| 48 | #--------------------误差曲线图--------------------- |
| 49 | #ax = fig.gca() |
| 50 | E = abs(u[:,m]-u1[:,m]) |
| 51 | ax.plot(x,E,'bo-'); |
| 52 | plt.xlabel('x') |
| 53 | plt.xticks([0,0.2,0.4,0.6,0.8,1.0]) |
| 54 | plt.ylabel(*u*'误差') |
| 55 | #plt.title(u't=1时误差曲线，n= %d m = %d'%(n,m)) |
| 56 | #------------------------------------------------- |
| 57 | #plt.show() |
| 58 | #plt.savefig('1.jpg') |