

电动力学数值实验报告

欧纪阳 2019141220016

2021 年 5 月 14 日

摘要

本文报告了一种在二维空间中求解泊松方程的一类边值问题的方法——多重网格-五点差分-共轭梯度法，该算法为有限差分法的改进方法，利用五点差分法作为数值微分近似，构建对角线稀疏矩阵，以共轭梯度方法为线性系统求解器，采用 V 循环多重网格方法加速迭代。本文还展示了用 Python 求解在有限矩形区域内给定电荷分布，并已知电势的边值的条件下的，该区域内的电势和电场分布。相较于高斯-赛德尔、过松弛迭代等求解算法，其收敛速度更快，精度更高。在复杂的边值条件下，与普通网格下的共轭梯度迭代相比，可快速降低误差。

关键词：多重网格法 共轭梯度法 稀疏矩阵 泊松方程 边值问题 Python

目录

I 电荷计算电场	2
II 电场计算电荷	2
1 方法	2
2 结果	2
III 单粒子在均匀电磁场中的运动	3
3 方法	3
4 结果	4
4.1 静磁场	4

Part I

电荷计算电场

该部分内容较长，另附文章《多重网格-五点差分-共轭梯度法——求解二维泊松方程的一类边值问题：从电荷分布求解电场的数值方法》

Part II

电场计算电荷

1 方法

对于一个三阶可微函数 $f(x)$ ，根据泰勒定理有，

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f^{(3)}(c_1) \quad (1)$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f^{(3)}(c_2) \quad (2)$$

其中 $x-h < c_2 < x < c_1 < x+h$ ，最后一项为误差项，两式相减可以得到一阶导数的三点中心差分公式，

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6}f^{(3)}(c) = \frac{f(x+h) - f(x-h)}{2h} + O(x^3) \quad (3)$$

用该公式在离散情况下对 $\vec{E} = -\nabla\varphi$ 做近似可得，

$$\vec{E}_{i,j} = \left(\frac{\varphi_{i+1,j} - \varphi_{i-1,j}}{2h_x}, \frac{\varphi_{i,j+1} - \varphi_{i,j-1}}{2h_y} \right) \quad (4)$$

其中 $\{\varphi_{i,j} | 0 \leq i \leq m-1, 0 \leq j \leq n-1\}$, h_x, h_y 分别为 x 方向和 y 方向上的网格精度，利用以下代码可以在 Python 中实现此功能，其中 Ex 和 Ey 为网格矩阵 (xx,yy) 的函数，所有是空间位置函数的变量（例如 Ex）的行标号用于表示 y 轴坐标，列标号用于表示 x 轴坐标。

```

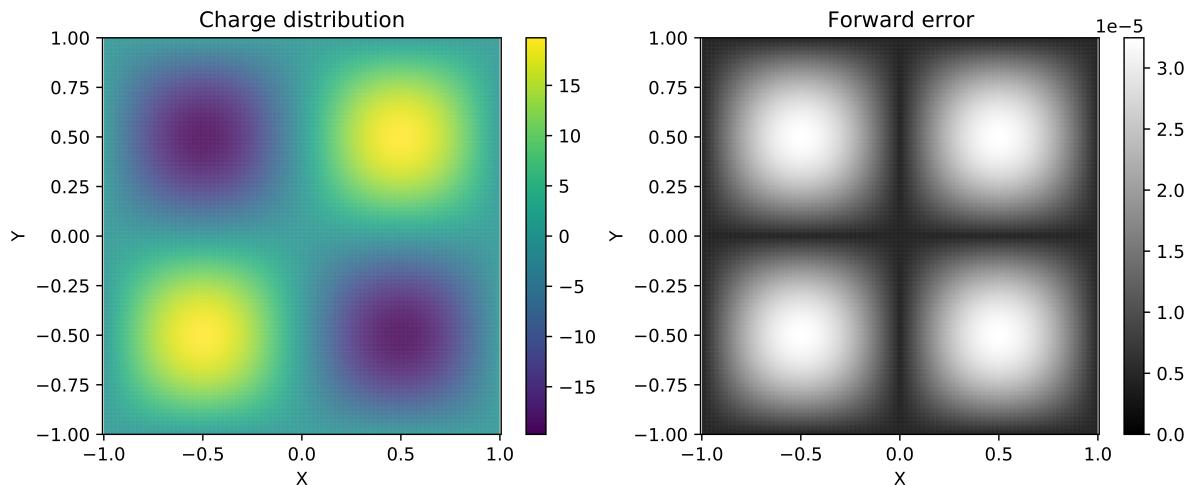
1 import numpy as np
2
3 d = np.array([1e-3, 1e-3]) # [y,x]
4 x = np.arange(xMin, xMax+d[1], d[1])
5 y = np.arange(yMin, yMax+d[0], d[0])
6
7 xx, yy = np.meshgrid(x, y)
8 Ex, Ey = Function(xx, yy)
9 Exy = np.stack((Ey, Ex), axis=2)
10 rho1 = (Exy[1:-1, 2:, 1]-Exy[1:-1, :-2, 1])/2/d[1]*epsilon0
11 rho2 = (Exy[2:, 1:-1, 0]-Exy[:-2, 1:-1, 0])/2/d[0]*epsilon0
12 rho = rho1+rho2

```

2 结果

给予电场一正余弦分布，

$$E = -\pi \cos(\pi x) \sin(\pi y) \vec{e}_x + \pi \sin(\pi x) \cos(\pi y) \vec{e}_y \quad (5)$$



左图为数值计算结果，第一象限与第三象限分别有一个波峰，波谷存在于第二和第四象限内；右图为与解析解对照的前向误差图，用灰度表示误差的大小，颜色约浅表示误差越大，衍射越深表示误差越小。

图 1：正余弦电场分布的数值计算结果及误差图

根据高斯定理 $\nabla \cdot E = \rho/\epsilon$ ，设 $\epsilon = 1$ ，则可以得到电荷分布的解析解，

$$\rho = 2\pi^2 \sin(\pi x) \sin(\pi y) \quad (6)$$

利用 Python 编写上述的求解方法，对此问题进行求解，设置求解区域为 $S = \{(x, y) | -1 \leq x \leq 1, -1 \leq y \leq 1\}$ 的正方形区域，在两个维度上的网格精度均为 10^{-3} ，即求解区域的网格大小为 2000×2000 。该模型的数值计算结果以及误差如图1所示，由误差图可以看出计算精度已经达到了四位有效数字，在结果中数值越大的区域其误差也越大，根据三点中心差分公式，其误差来源可以认为是被忽略掉了的二阶小量 $O(x^2) \sim (h_x^2 + h_y^2)\pi^3 \sim 10^{-5}$ ，粗略估计由方法所带来的误差与实际计算的误差相符。

Part III

单粒子在均匀电磁场中的运动

3 方法

考虑带电量为 Q ，质量为 m 的带电粒子在电磁场中的运动，其在电磁场中的运动方程为，

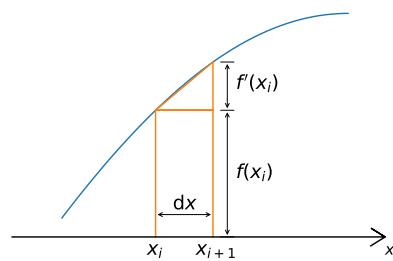
$$m\vec{a} = q\vec{v} \times \vec{B} + q\vec{E} \quad (7)$$

粒子运动的位移可以表示为其运动速度的积分，利用梯形法则可以写作，

$$\vec{r} = \int_{t_0}^{t_1} \vec{v} dt = \sum_i \frac{dt}{2} [v(t_i) + v(t_{i+1})] \quad (8)$$

$$= \sum_i v(t_i) dt + \frac{1}{2} v'(t_i) dt^2 \quad (9)$$

$$= \sum_i v_i dt + \frac{1}{2} a_i dt^2 \quad (10)$$



利用以下代码可以在 python 中实现此功能，所有的矢量的三个分量按照 (x,y,z) 的顺序排列，其中 np.cross 函数表示向量的叉乘。

```

1 import numpy as np
2
3 E = np.array([0., 0., 0.]) # 电场强度
4 B = np.array([0., 0., 1.]) # 磁感应强度
5 Q = 1 # 电荷量
6 m = 1 # 质量
7 r0 = np.array([-1., 0., 0.]) # 初始位移
8 v0 = np.array([0., 1., 0.]) # 初始速度
9 dt = 1e-3 # 时间精度
10 T = 20*np.pi # 求解时长
11 tt = np.arange(0, T+dt, dt)
12
13 r = np.array([r0])
14 v = np.array([v0])
15 for t in tt[1:]:
16     a = (Q*E+Q*np.cross(v, B))/m
17     dr = v[-1]*dt+(a*dt**2)/2
18     dv = a*dt
19     r = np.row_stack((r, r[-1]+dr)) # 更新一个位置，并连接到位置数组的最后一行之后
20     v = np.row_stack((v, v[-1]+dv)) # 更新一个速度，并连接到速度数组的最后一行之后

```

4 结果

4.1 静磁场

首先考虑只存在静磁场，不存在静电场的情况，令 $B = (0, 0, 1)$, $r_0 = (-1, 0, 0)$, $v_0 = (0, 1, 0)$ ，时间步长 $dt = 10^{-3}$ ，时长 $T = 2\pi$ ，进行数值求解，粒子的运动轨迹如图2所示，可以看到粒子在绕坐标原点做圆周运动，该问题具有解析解，

$$\vec{r} = (-\cos(t), \sin(t)) \quad (11)$$

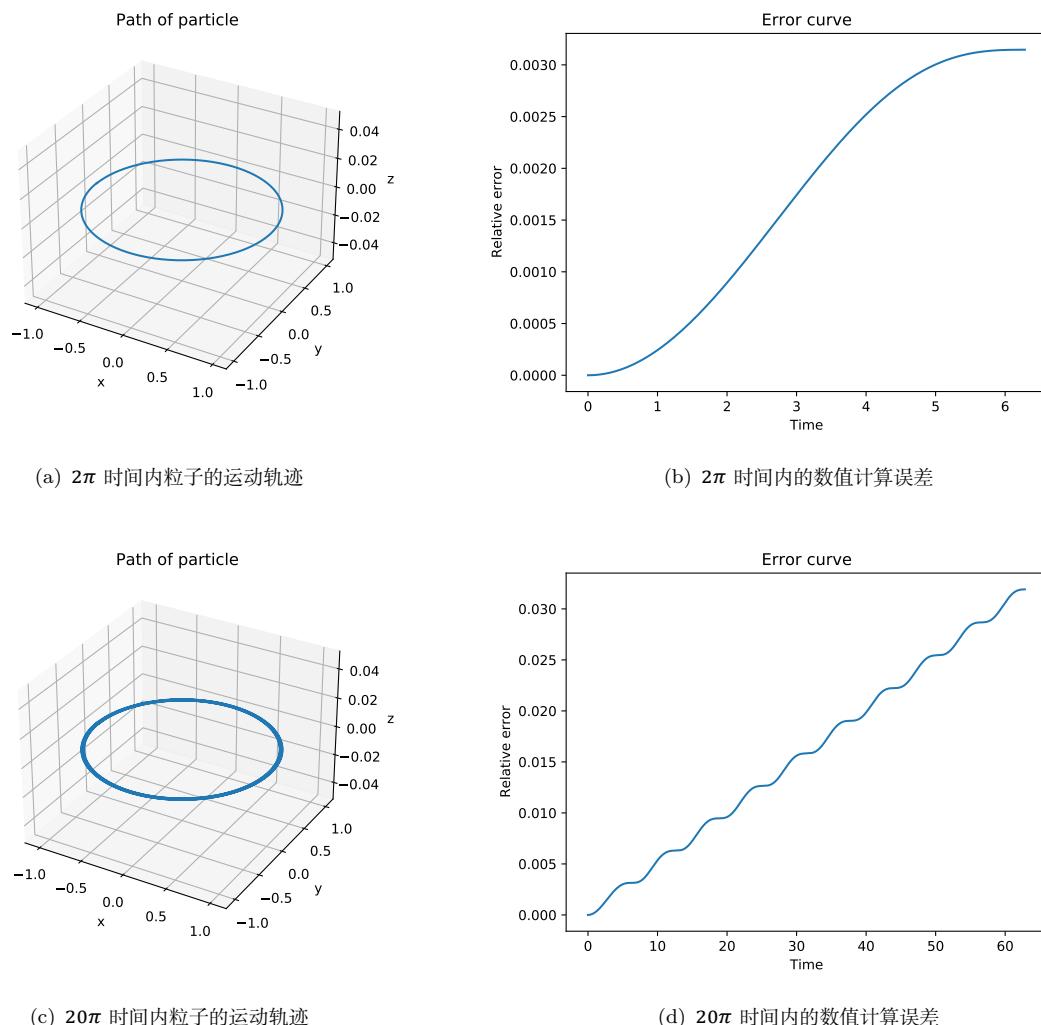


图 2: 只存在磁场时的数值计算结果