计算物理作业十六

于浩然 PB19020634 2021.12.23

1 作业题目

进行单中心 DLA 模型的模拟 (可以用圆形边界，也可以用正方形边界)，并用两种方法计算模拟得到的 DLA 图形的分形维数，求分形维数时需要作出双对数图.

2 算法简介

2.1 DLA 模型

DLA(扩散限制聚集) 模型是一种著名的二维随机生长图形。要产生这一图形，则需要:

- 首先在二维正方格子的中心放一个粒子作为核心，再在远处随机产生一个粒子使其做随机扩散运动;
- 当粒子运动到已存在粒子周围的八个格子中时，停止随机扩散，粒子成为核心的一部分;
- 在远处不断产生新的粒子重复上述过程，直到成千上万个粒子聚集成为分叉状的 DLA 生长图形.

2.2 盒计数法求分形维数

将一定尺寸 (边长为 ε) 的网格覆盖在分形图形上，计数网格中有图形像素的方格数目 $N(\varepsilon)$，直至网格尺寸达到像素大小为止. 为减少误差，应当使不同尺寸的网格能覆盖相同大小的图形. 作 $\ln N(\varepsilon) \sim \ln(1/\varepsilon)$ 图，若得到一条直线，则直线的斜率 D 即为图形的分维.

2.3 Sandbox 法求分形维数

Sandbox 法即为将一系列尺寸 $r(r > 1)$ 的不断增大的方框 (或圆) 覆盖到分形图形上，计数不同方框 (或圆) 中像素数 N. 作 $\ln N \sim \ln r$ 图，则直线部分的斜率即为分形维数 D.

3 编程实现

用 Fortran90 编写程序，分别在三个子程序中实现 DLA 模型的生成、盒子计数法求分维、Sandbox 法求分维，并装在一个模块 `MODULE DLA` 中.

- `SUBROUTINE Generator(particles, steps, r0, rmax, resc, filename)`

  基于随机数生成 DLA 图形的子程序. 使用两组随机数，如下：

  - `rand0`

    数组大小和粒子数 `particles` 相同，用于生成每个粒子的初始位置.

  - `rand`

    数组大小和预计最大步数 `steps` 相同，用于决定每一次随机行走的方向，具体实现见函数 `FUNCTION walk(dir)`.

  还有一个算法用于验证粒子是否已经运动到已存在粒子的附近：

  - 首先有一个用于记录网格占据状态的网格矩阵 (数组) `lattice`，当网格上某处有已存在的粒子则在此处 `lattice=1`. 粒子当前位置的坐标为 (x, y)(二维数组)，对网格矩阵的切片 `lattice(x-1:x+1, y-1:y+1)` 求和：若求和不为零，说明周围已经存在其他粒子，停止模拟.

  具体取参数为: 模拟的总粒子数 `particles=100000`;预计最大步数 `steps=500000`，当粒子超过这一步数还没有结合是死亡；初始随机点与中心点固定距离 `r0 = 400`，点会在这一半径的圆周上随机生成；结合图形的边界取为正方形框，其边长的一半为 `rmax = 256`；粒子逃逸距离 `resc=450`，当粒子的横坐标或纵坐标值比 `resc` 大时，粒子死亡.

- `SUBROUTINE Box(resc, rmax, filename)`

  读取前面子程序中生成的大小为 (-450:450, -450:450) 的 `lattice` 矩阵，为简化取网格步骤取其 (-255:256, -255:256) 部分. 取 $\varepsilon = 2^k$，容易得到各 ε 值对应的 $N(\varepsilon)$.

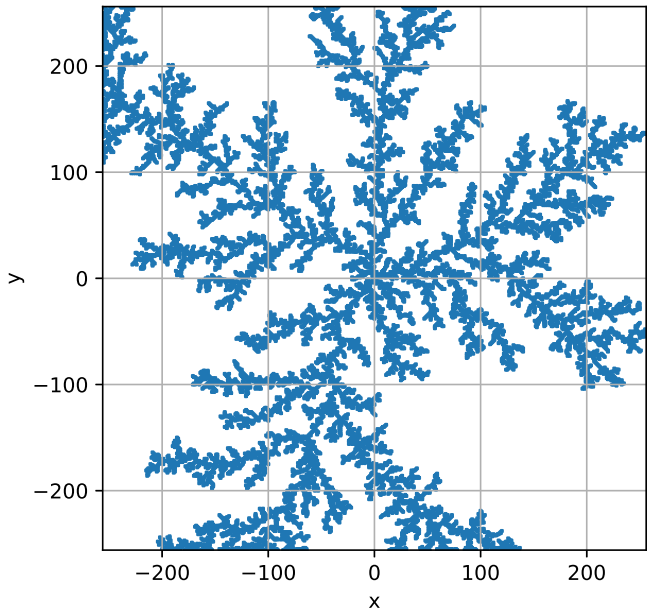
- `SUBROUTINE Sandbox(resc, rmax, filename)`

  与上面的子程序同理，取 $r = 2^k$.

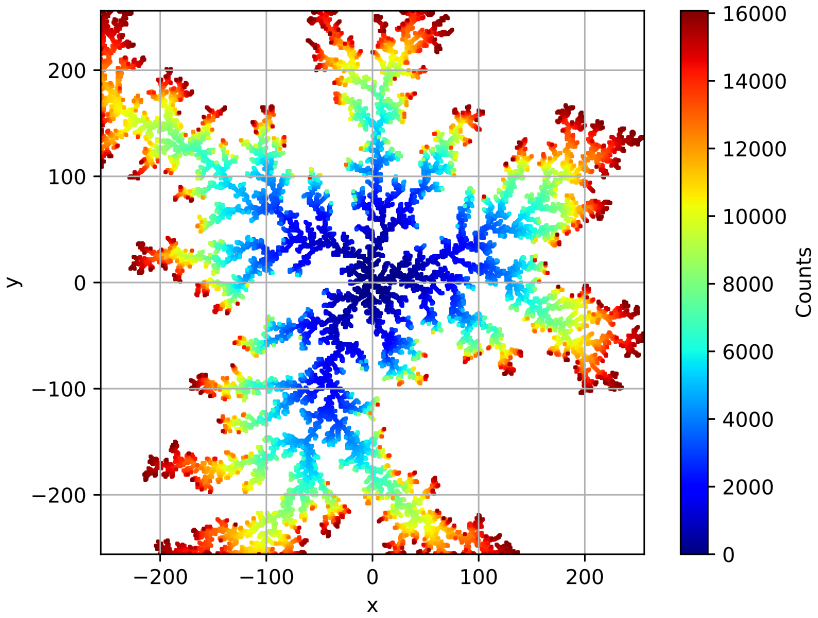
4 计算结果

4.1 DLA 生长图形

将计算所得的 DLA 模型绘制散点图展示如下：



(a)



(b)

图 1: DLA 生长模型 ($r = 256pxs$)

图 1(a) 清晰地展示了 DLA 图形的枝杈状结构，而在图 1(b) 中展示了点生成的先后顺序.

4.2 盒子计数法求分维

作图并显示拟合直线及其信息：

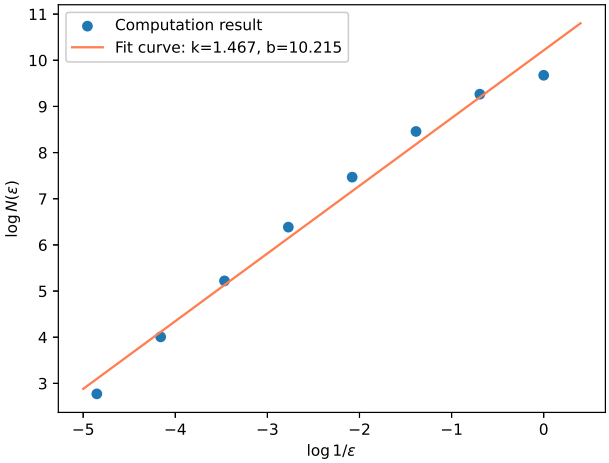


图 2: 盒计数法拟合曲线

由直线斜率可知：所得 DLA 图形的分维为 $D = 1.467$.

4.3 Sandbox 法求分维

作图并显示拟合直线及其信息：

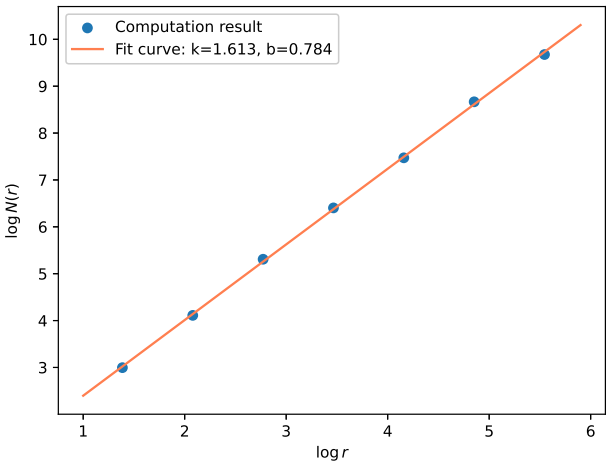


图 3: Sandbox 法拟合曲线

由直线斜率可知，所得 DLA 图形的分维为 $D = 1.613$.

5 结论

从所作的图像可看出直线拟合的非常好，得到了比较合理的结果. 计算所得的分维由于计算方法不同而存在一定差异，这可能随着模拟点的进一步增多而逐渐消失.

本次作业实现了 DLA 模型的模拟，进一步实践了随机行走的运用，并计算了其分维以更好地了解分形的性质.

6 源代码

6.1 Fortran90 代码

```
MODULE DLA
IMPLICIT NONE
REAL(KIND=8) :: pi = 3.14159265358979
CONTAINS
SUBROUTINE Generator(particles, steps, r0, rmax, resc, filename)
    ! 生成DLA图形的子程序
    INTEGER(KIND=4), INTENT(IN) :: particles, steps, r0, rmax,
       resc
    INTEGER(KIND=4) :: i, j, dir, r(2), pos(0:particles, 2),
       recorded
    INTEGER(KIND=4), DIMENSION(-resc:resc, -resc:resc) ::
       lattice
    REAL(KIND=8) :: seed, rand0(particles), rand(steps)
    CHARACTER(LEN=*) :: filename

    CALL RANDOM_NUMBER(seed)
    CALL Schrage(particles, INT(2147483647 * seed), rand0)
    lattice = 0 ! 用来记录坐标对应网格的占有情况
    lattice(-1:1, -1:1) = 1 ! 设置原点初始占据
    pos(0, :) = [0, 0] ! 记录DLA图形上像素点的坐标
    recorded = 1
    DO i = 1, particles
        r(1) = INT(r0 * COS(2 * pi * rand0(i)))
        r(2) = INT(r0 * SIN(2 * pi * rand0(i)))
        ! 用一个随机数产生粒子开始随机行走的初始点

        CALL RANDOM_NUMBER(seed)
        CALL Schrage(steps, INT(2147483647 * seed), rand)
```

```
25          ! 产生用于判断粒子运动方向的随机数序列
26          DO j = 1, steps
27              dir = INT(4 * rand(j)) ! 设dir=0,1,2,3分别对应左,下,
                    右,上
28              r = r + walk(dir) ! 粒子随机行走一步
29              IF (ABS(r(1)) > resc .OR. ABS(r(2)) > resc) THEN
30                  EXIT ! 若大于逃逸距离则停止计算
31              ENDIF
32              IF (SUM(lattice((r(1) - 1):(r(1) + 1), (r(2) - 1):(r
                    (2) + 1))) .NE. 0) THEN
33                  IF(ABS(r(1)) <= rmax .AND. ABS(r(2)) <= rmax)
                        THEN
34                      ! 如果点在正方形范围(-rmax:rmax, -rmax:rmax)
                            内则记录
35                      lattice(r(1), r(2)) = 1
36                      ! 若粒子附近的八个点不是完全空的, 则停止随机
                            扩散使其成为核心的一部分
37                      pos(recorded, 1) = r(1)
38                      pos(recorded, 2) = r(2)
39                      print *, r(1), r(2), real(i) / particles *
                            100, '%'
40                      ! 记录新添加到DLA图形上像素点的坐标
41                      recorded = recorded + 1 ! 添加到DLA图形上的
                            像素数+1
42                  END IF
43                  EXIT
44              END IF
45          END DO
46      END DO
47      OPEN (1, file=filename)
48      DO i = 1, recorded
49          WRITE (1, *) pos(i, :)
50      END DO
51      CLOSE (1)
52      OPEN (1, file='lattice.dat')
53      WRITE (1, *) ((lattice(i, j), j = -resc, resc), i = -resc,
            resc)
54      CLOSE (1)
55  END SUBROUTINE Generator
```

```
56
57 SUBROUTINE Sandbox(resc, rmax, filename)
58     CHARACTER(LEN=*) :: filename
59     INTEGER(KIND=4),INTENT(IN) :: resc, rmax
60     INTEGER(KIND=4) :: i, j, k, N(2:8), r(2:8)
61     INTEGER(KIND=4) :: lattice(-resc:resc, -resc:resc), mesh(-
           rmax:rmax, -rmax:rmax)
62     ! 读取Generator子程序中生成的格点占据矩阵
63     OPEN (1, file='lattice.dat')
64     READ (1, *) ((lattice(i, j), j = -resc, resc), i = -resc,
           resc)
65     CLOSE (1)
66     mesh = lattice(-rmax:rmax, -rmax:rmax) ! 取画出图像部分的像
           素点占据情况
67     N = 0
68     DO k = 2, 8
69         r(k) = 2**k
70         DO i = -r(k)+1, r(k)
71             DO j = -r(k)+1, r(k)
72                 IF(lattice(i, j) .EQ. 1) THEN
73                     N(k) = N(k) + 1 ! 当lattice值为1表示存在一个
                           像素, N值累加
74                 END IF
75             END DO
76         END DO
77     END DO
78     OPEN (1, file=filename)
79     DO k = 2, 8
80         WRITE (1, *) r(k), N(k)
81     END DO
82     CLOSE (1)
83 END SUBROUTINE Sandbox
84
85 SUBROUTINE Box(resc, rmax, filename) ! 盒子计数法求分维子程序
86     CHARACTER(LEN=*) :: filename
87     INTEGER(KIND=4), INTENT(IN) :: resc, rmax
88     INTEGER(KIND=4) :: i, j, k, N(0:7), eps(0:7)
89     INTEGER(KIND=4) :: lattice(-resc:resc, -resc:resc), mesh(-
           rmax+1:rmax, -rmax+1:rmax)
```

```
90      ! 读取Generator子程序中生成的格点占据矩阵
91      OPEN (1, file='lattice.dat')
92      READ (1, *) ((lattice(i, j), j = -resc, resc), i = -resc,
            resc)
93      CLOSE (1)
94      mesh = lattice(-rmax+1:rmax, -rmax+1:rmax) ! 取画出图像部分
            的像素点占据情况
95      N = 0
96      DO k = 0, 7
97          eps(k) = 2**k
98          DO i = -256 / eps(k), 256 / eps(k)
99              DO j = -256 / eps(k), 256 / eps(k)
100                 IF(SUM(lattice((i * eps(k) + 1):((i+1) * eps(k))
                        , (j * eps(k) + 1):((j+1) * eps(k)))) .NE. 0)
                        THEN
101                     N(k) = N(k) + 1 ! 若网格中非空则盒子计数+1
102                 END IF
103             END DO
104         END DO
105     END DO
106     OPEN (1, file=filename)
107     DO k = 0, 7
108         WRITE (1, *) eps(k), N(k)
109     ENDDO
110     CLOSE (1)
111 END SUBROUTINE Box
112
113 FUNCTION walk(direction) ! 将数字代号转换成位移坐标的函数
114     INTEGER(KIND=4) :: direction
115     INTEGER(KIND=4), DIMENSION(2) :: walk
116     SELECTCASE(direction)
117         CASE(0)
118           walk = [-1, 0]
119         CASE(1)
120           walk = [0, -1]
121         CASE(2)
122           walk = [1, 0]
123         CASE(3)
124           walk = [0, 1]
```

```
125     END SELECT
126 END FUNCTION walk
127 END MODULE DLA
128 
129 SUBROUTINE Schrage(num, z0, rand)
130     !Schrage随机数生成器子程序,将均匀随机数序列存放在数组rand中
131     IMPLICIT NONE
132     INTEGER(KIND=4) :: N = 1, num
133     INTEGER :: m = 2147483647, a = 16807, q = 127773, r = 2836,
            In(num), z0
134     REAL(KIND=8), INTENT(INOUT) :: rand(num)
135     In(1) = z0 !将传入值z0作为种子
136     rand(1) = REAL(In(1))/m
137     DO N = 1, num - 1
138         In(N + 1) = a * MOD(In(N), q) - r * INT(In(N) / q)
139         IF (In(N + 1) < 0) THEN !若值小于零, 按Schrage方法加m
140             In(N + 1) = In(N + 1) + m
141         END IF
142         rand(N + 1) = REAL(In(N + 1))/m !得到第N+1个随机数
143     END DO
144 END SUBROUTINE Schrage
145 
146 PROGRAM MAIN
147     USE DLA
148     IMPLICIT NONE
149     ! CALL Generator(100000, 500000, 400, 256, 450 ,'test.dat')
150     CALL Box(450, 256, 'box.dat')
151     ! CALL Sandbox(450, 256, 'sandbox.dat')
152 END PROGRAM MAIN
```

6.2 python 绘图代码

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib as mpl
4 import math
5 
```

```
plt.rcParams['savefig.dpi'] = 300
plt.rcParams['figure.dpi'] = 300

dat = np.loadtxt('0_2.dat')
x = dat[0:100000]
y = dat[100001:200001]
plt.xlabel('x')
plt.ylabel('y')
plt.plot(x, y, linewidth=0.1)
plt.savefig('0_2.eps')
plt.show()
plt.scatter(x, y, c=range(100000), cmap=mpl.cm.jet, s=0.1)
plt.colorbar(label="Counts", orientation='vertical')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('0_2_1.eps')
plt.show()

dat = np.loadtxt('1_0.dat')
x = dat[0:100000]
y = dat[100001:200001]
plt.xlabel('x')
plt.ylabel('y')
plt.plot(x, y, linewidth=0.1)
plt.savefig('1_0.eps')
plt.show()
plt.scatter(x, y, c=range(100000), cmap=mpl.cm.jet, s=0.1)
plt.colorbar(label="Counts", orientation='vertical')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('1_0_1.eps')
plt.show()

dat = np.loadtxt('5_0.dat')
x = dat[0:100000]
y = dat[100001:200001]
plt.xlabel('x')
plt.ylabel('y')
plt.plot(x, y, linewidth=0.1)
```

```
45 plt.savefig('5_0.eps')
46 plt.show()
47 plt.scatter(x, y, c=range(100000), cmap=mpl.cm.jet, s=0.1)
48 plt.colorbar(label="Counts", orientation='vertical')
49 plt.xlabel('x')
50 plt.ylabel('y')
51 plt.savefig('5_0_1.eps')
52 plt.show()
```