03：DLA

许传奇 PB16021546

1 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），并用两种方法计算你模拟得到的DLA图形的分形维数，求分形维数时需要作出双对数图。

2 原理与算法

2.1 原理

2.1.1 团簇的形成

在模型图像中心放置一个粒子，作为起始粒子，也作为起始团簇。

当新粒子运动到与中心粒子接触时，它们就聚集成更大团簇。当第二个新粒子与团簇接触时，也聚集到团簇中……

这样循环往复，团簇将变得越来越大，生成粒子聚集的图像。

2.1.2 新粒子的生成与并入团簇

取一条离中心较远的曲线，最好是中心对称的闭曲线，随机在曲线上任一位置产生一个新粒子。

产生的新粒子随即不停地随机行走，即粒子有概率向其他任意方向走一定的距离，直到粒子满足上面所提到的并入团簇的条件——与团簇接触时，就并入团簇。

当然，由于是随机行走，粒子可能最终与团簇接触，也可能在总体上离团簇越来越远。但在实际中，粒子数非常大，因此有相当多的粒子与团簇接触，团簇也变得足够大。

2.1.3 扩散限制聚集模型(Diffusion-limited Aggregation, DLA)

用计算机模拟上述的团簇生成的过程即为DLA模型。

但DLA模型与实际生成的模型存在一些区别。例如，计算机程序的离散性让粒子的扩散方向与距离并不是任意实数，也并不是足够多的粒子同时随机行走等。但模拟所得到的结果，与真实情况十分相似，因此计算机模拟DLA模型也是非常好的应用。

最后模拟的模型为雪花状图形，具有与分形图案类似的特征与性质，也可以近似计算其分形维数。

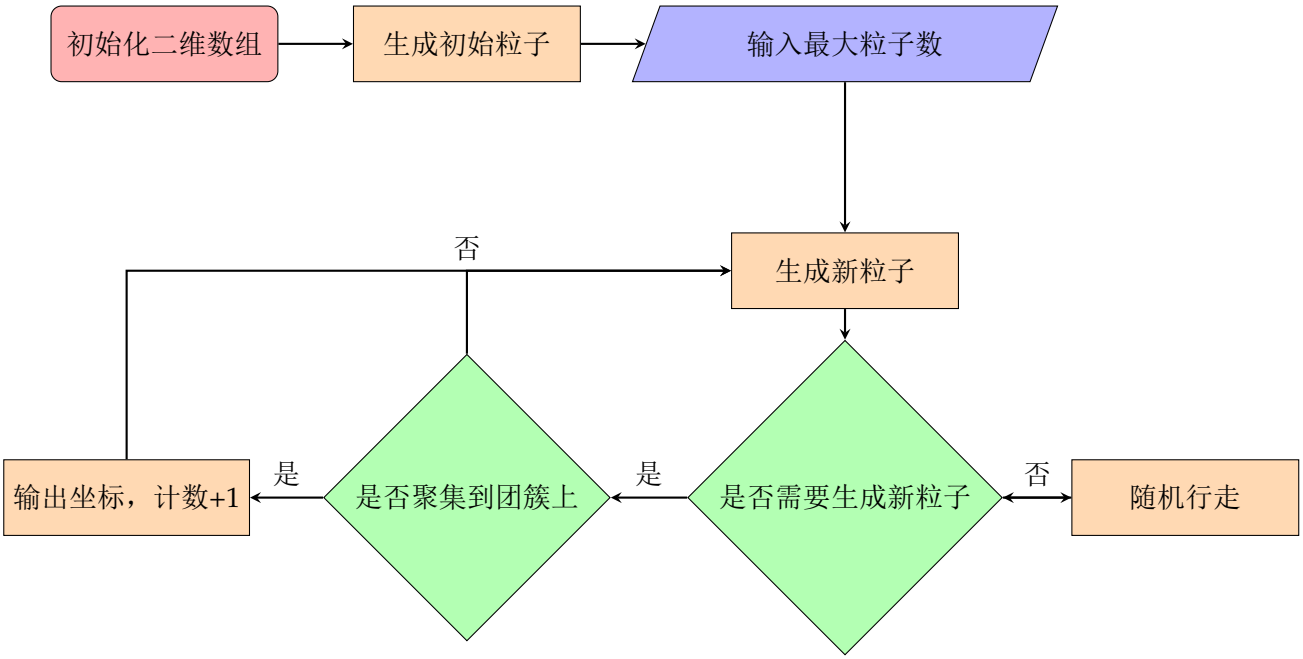
2.2 算法

2.2.1 DLA模型生成

1. 原理简述：

   运用产生新粒子的函数、随机行走的函数、判断是否需要新粒子的函数，总共3个函数来实现以下步骤。

   起始时设置初始粒子数为1，输入一个数作为最大粒子数。大致步骤如下：



   其中，判断“是否需要生成新粒子”和“是否聚集到团簇上”通过一个函数，即判断是否需要生成新粒子的函数的3个返回值来判断。

   通过该循环，即可依次输出属于DLA模型的点的坐标。

2. 代码模块简述：

   (a) DLA模型：

   用一个较大的二维数组来表示，为了方便，二维数组的横、纵坐标也是生成模型中点的横、纵坐标。当二维数组中元素为0时，代表此处没有粒子；当元素为1时，代表此处有粒子。初始时设置数组中心的元素为1，即表示起始的粒子。

   (b) 产生新粒子的函数：

   本实验中我们设置产生新粒子的边界为正方形。先随机生成0、1、2、3中的一个数，分别代表正方形的上、下、左、右四条边。然后再生成0、1、2、……、正方形边长中的一个数，代表边的从下到上，或者从左到右的这个数这么多单位长度的坐标。返回该点的横、纵坐标，即得到了生成的新粒子的横、纵坐标。

   (c) 随机行走的函数：

   随机生成0、1、2、3中的一个数，分别表示向上、下、左、右四个方向走一个单位长度。返回该点新的横、纵坐标，即代表做了一次随机行走。

(d) 判断是否需要生成新粒子的函数：

有两个判断条件：

第一，若粒子当前位置的上、下、左、右、左上、左下、右上、右下八个方向有粒子，则表示粒子与团簇接触。此时将粒子当前位置的二维数组的元素置为1，且粒子数+1，表示粒子并入团簇中。

第二，若粒子当前位置超出给定的一个逃逸距离，则认为粒子离团簇足够远，无法再与团簇接触并入团簇中。我们就重新生成一个粒子。

若粒子当前位置不是上述两种情况，则没有操作。

2.2.2 分形维数计算

通过对源程序中两个参数COEFFICIENT_BOXCOUNT和COEFFICIENT_SANDBOX的修改，我们可以控制图片的大小。两个参数的值越大时，图片越小。图片大小对下面两种方法计算分形维数得到结果的正确性非常重要，我们将在5.3的讨论中解释。

1. 盒计数法：

   输入盒长度为r。

   对表示DLA模型的二维数组的中心的一个大的正方形区域进行分割，即先进行两个循环，遍历分割这个大正方形得到的盒。

   然后在每个盒内再进行两个循环，循环每一个数组元素。当循环到数组元素为1的时候，退出这两层循环到盒的循环中，且计数+1。若该盒的数组元素都遍历了却没有一个元素为1，则将自动遍历下一个盒。

   当所有盒遍历完毕后，返回计数值。

   在主函数中，设置一个循环，输入不同的r，即可得到相应的r与计数N的关系，然后对这些点进行拟合，得到的斜率的绝对值即为用盒计数法计算的分形维数。

2. Sandbox法：

   输入正方形长度为r，正方形中心在起始粒子位置。

   进行两个循环，遍历该正方形内的数组元素。当数组元素为1时，计数+1；否则遍历下一个数组元素。

   循环完毕，返回计数值。

   在主函数中，设置一个循环，输入不同的r，即可得到相应的r与计数N的关系，然后对这些点进行拟合，得到的斜率即为用Sanbox法计算的分形维数。

3 源文件使用说明

打开03DLA.exe执行文件，输入最大粒子数，随后等待计算机计算。

计算完毕后，将得到data.txt文件，其中第一行到倒数第三行保存在团簇中的每个点的横、纵坐标。倒数第二行和倒数第一行一次保存团簇中的粒子数、产生新粒子的边界、粒子的逃逸边界和图片大小。同时得到两个文件Boxcount.txt和Sandbox.txt，分别是盒计数法求得的点和Sandbox求得的点。

注意，应该先在源代码里修改宏定义的两个参数COEFFICIENT_BOXCOUNT和COEFFICIENT_SANDBOX。两个参数是用来控制合适的盒计数法和Sanbox法的图片大小，这点我们将在5.3的讨论中解释。当然，如果不计算分形维数，则不必修改。

打开plot.py，编译并运行，可以得到DLA模型的图片。

打开animation.py，编译并运行，可以得到DLA模型生成的动态图。

打开CalculateDimension.py，编译并运行，可以得到盒计数法和Sanbox法计算的分形维数对应的拟合图片和数值。

4 计算结果及具体分析

4.1 DLA模型

我们运行程序生成DLA，首先设定最大粒子数为500，如下图所示：

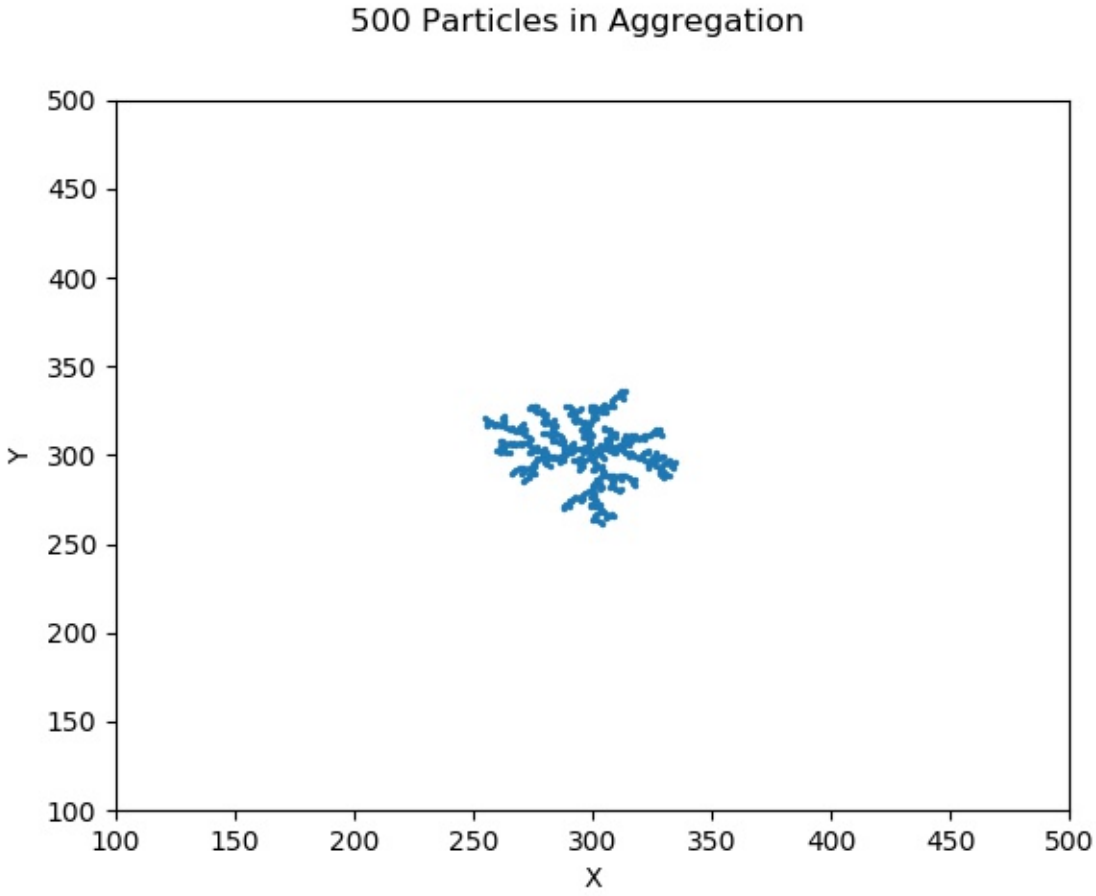


图 1: 500个粒子

生成的DLA模型较小，我们增大粒子数，设置成2000，如下图所示：

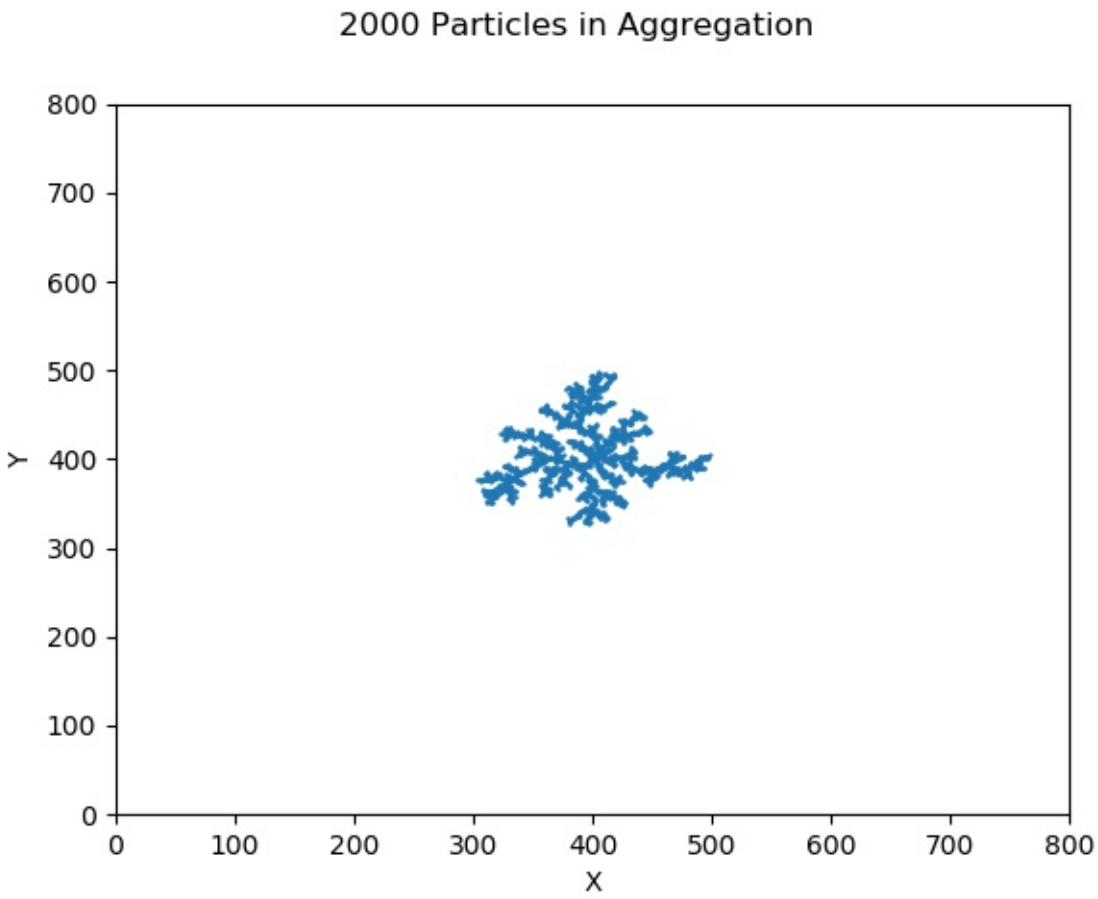


图 2: 2000个粒子

DLA模型已经初见规模，继续增大粒子数，设置成10000。如下图所示：

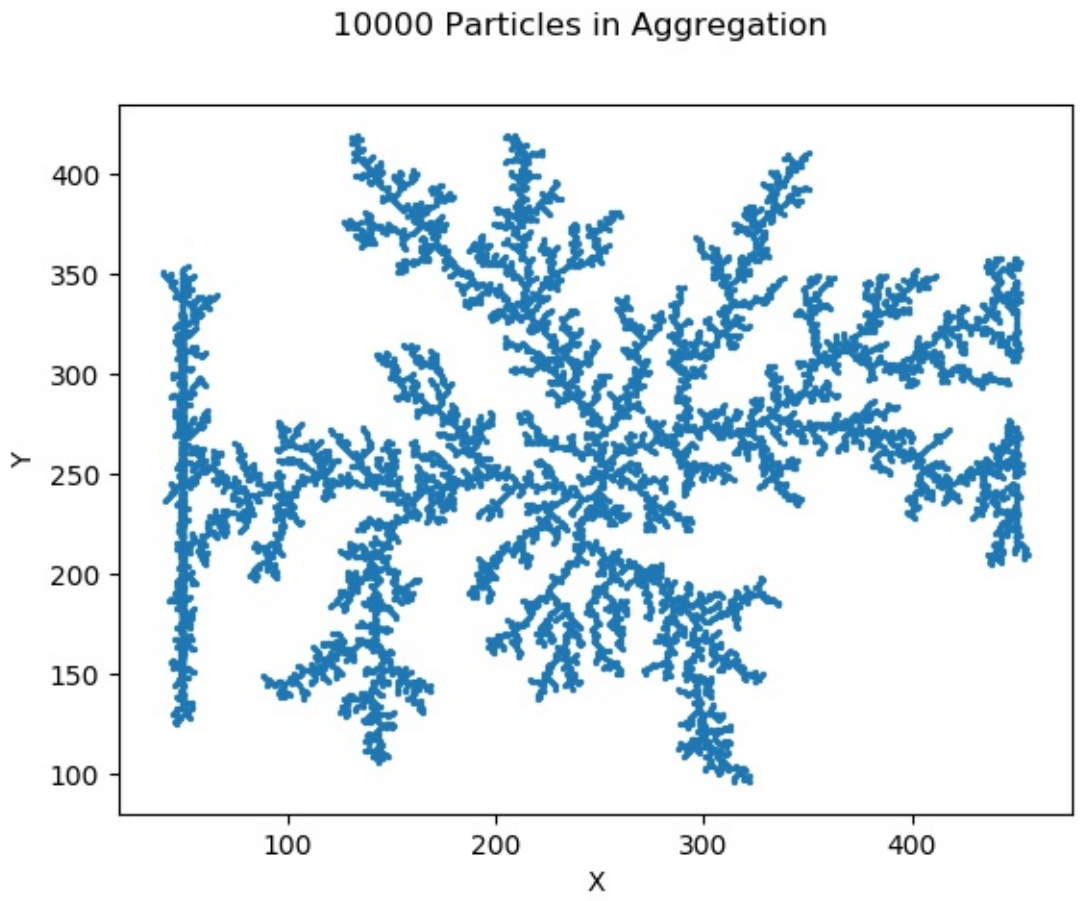


图 3: 10000个粒子（改进前）

出现了一个我们没有考虑到的问题。

由于生成新粒子的边界是固定的，因此当粒子数很大的时候，刚生成的粒子就会聚集到团簇上。可以看到上面的图3的左边和右边，粒子形成正方形边界的竖直的团簇，类似于附着在薄膜生长的容器的边缘。

为了解决这个问题，我们把生成新粒子的边界设置成可变的。当团簇上的新粒子达到边界时，我们就把生成新粒子的边界和逃逸距离同时增大。

这样就可以解决这个问题，如下图所示，是我们改进程序后生成10000个粒子的DLA模型的图案：

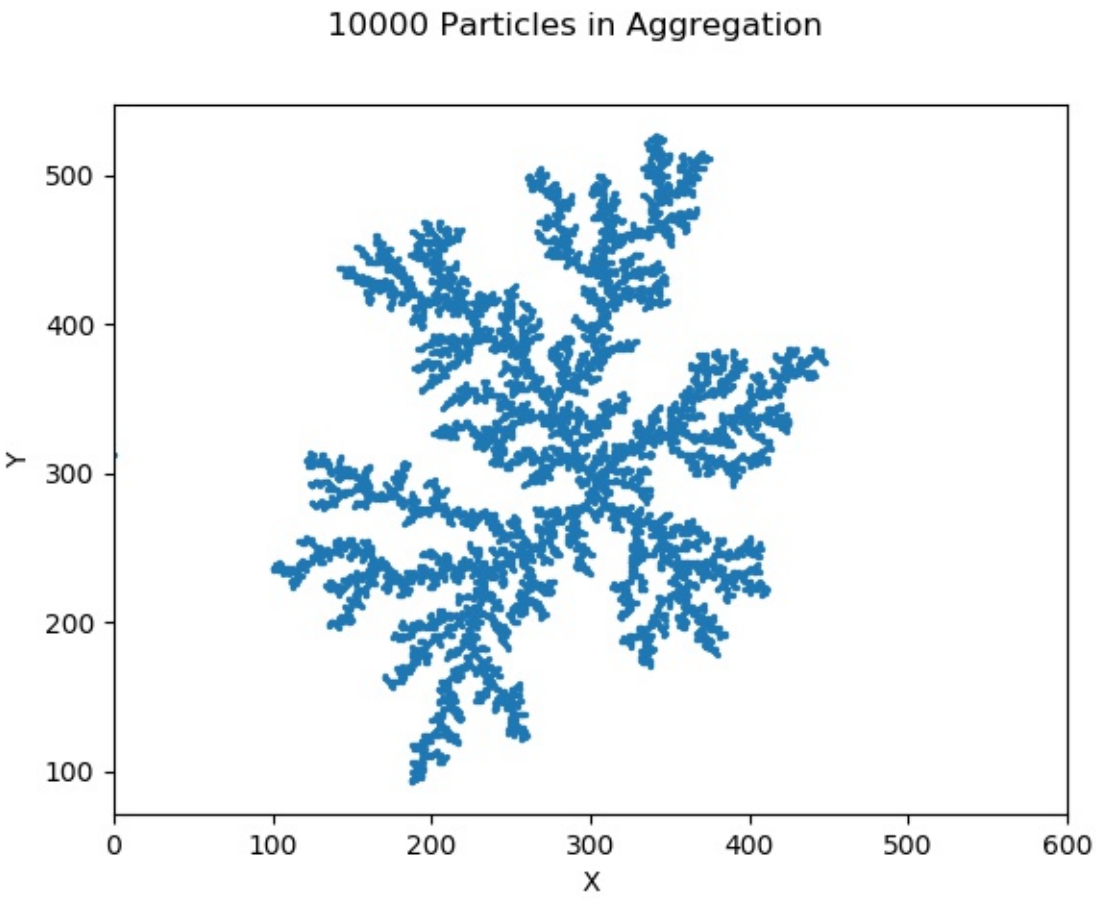


图 4: 10000个粒子（改进后）

可见已经解决了生成的粒子附着在容器边缘的情况。

另外，压缩包中附带了生成10000个粒子时的动态图，保存在10000ParticlesAnimation.mp4文件中。可以通过animation.py编译并运行得到。

注意：动画中的数据用的不是图4中的数据！

4.2 分形维数的计算

我们在源程序中将宏定义的两个参数COEFFICIENT_BOXCOUNT和COEFFICIENT_SANDBOX分别修改成15和4。运行程序，输入最大粒子数为2000求得的点运用CalculateDimension.py进行拟合，得到如下图片。

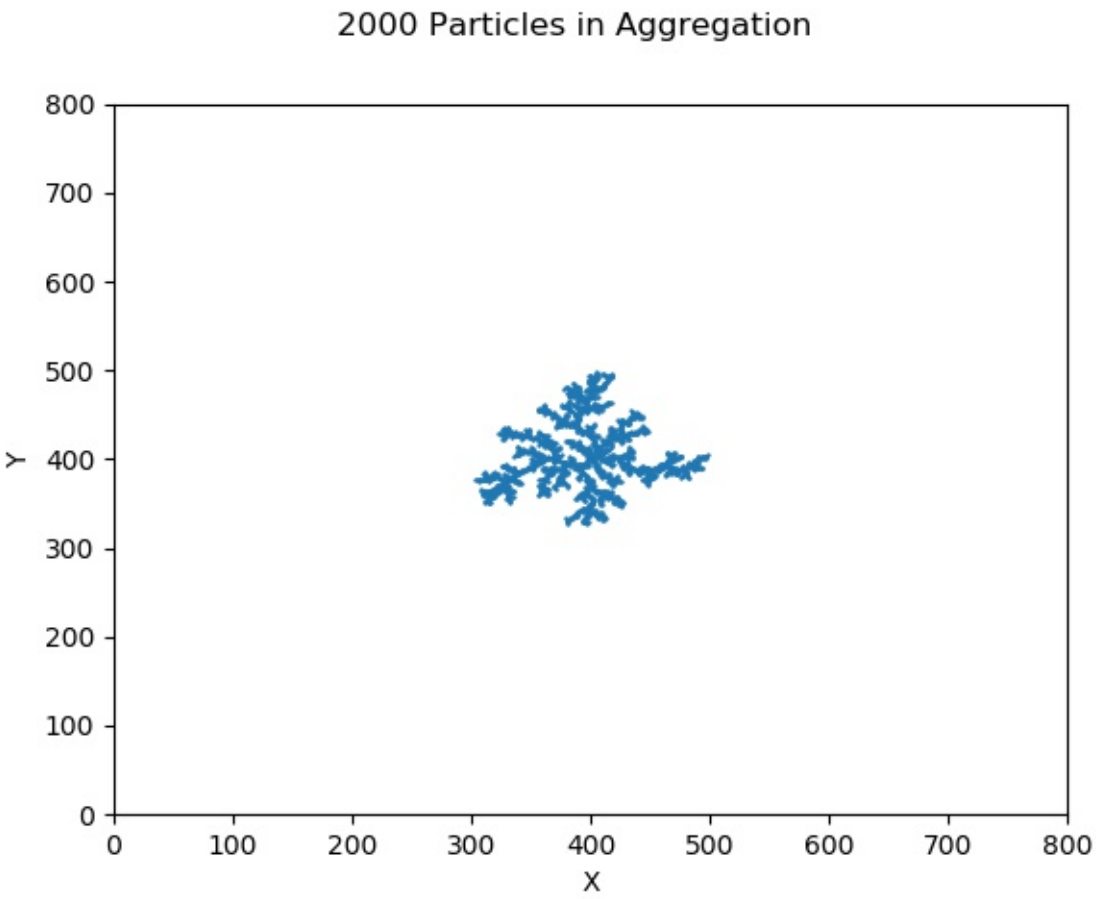

2000 Particles in Aggregation
800
700
600
500
400
300
200
100
0
Y
0
100
200
300
400
500
600
700
800
X

图 5: 2000个粒子

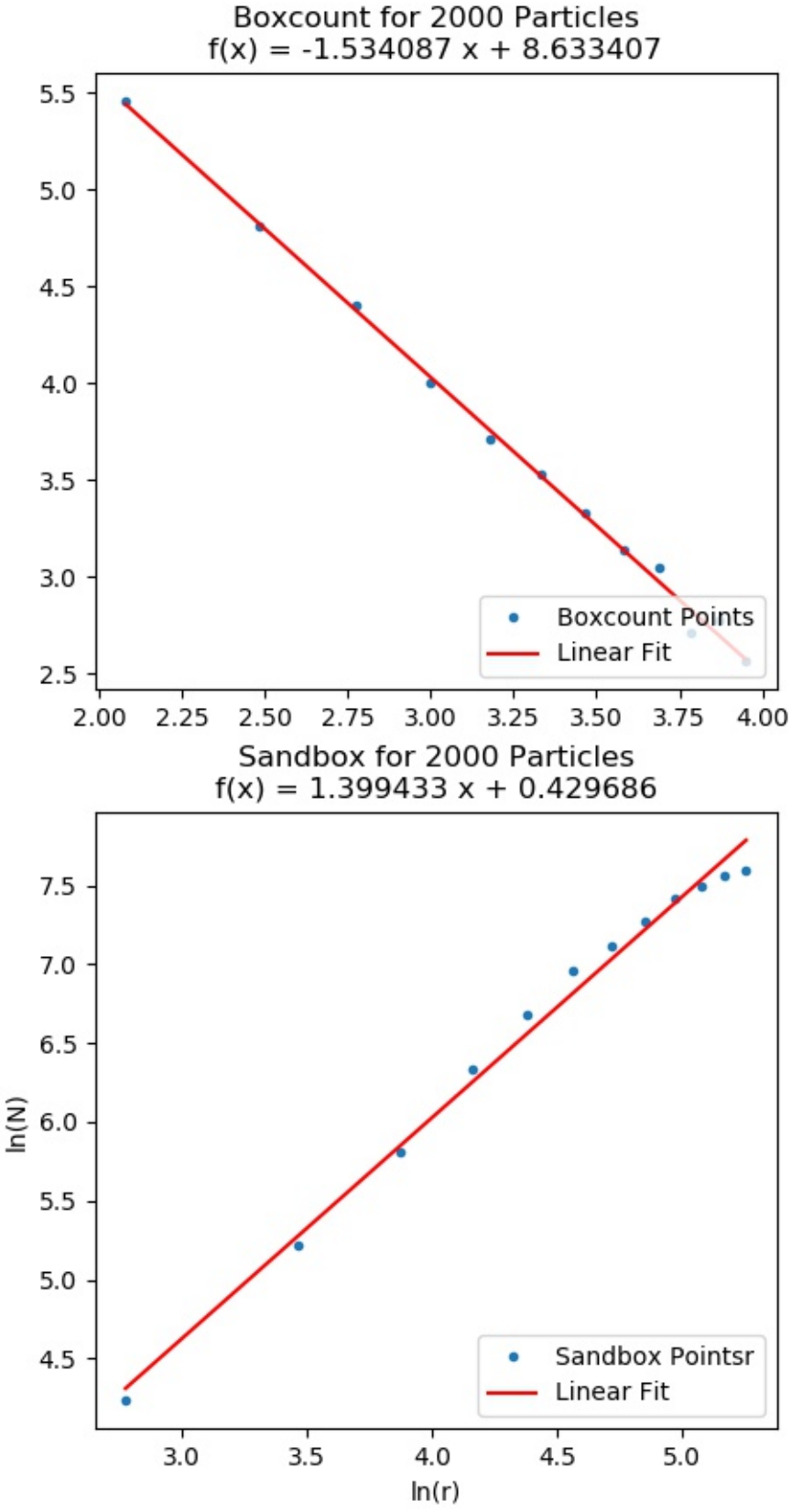


图 6: 2000个粒子求分形维数(修改前)

分形维数近似为1.534和1.399。

由于准确的分形维数应该是1.6-1.7，可见模拟得到的数值并不准确。

一方面我们可以修改宏定义中的参数来得到更正确的数值。

比如我们看两个方法中ln(r)大处的粒子已经发生偏转，这是盒计数法和Sandbox法本身的不足，我们将在5.3结果的讨论中提到。当我们把图片取得更小时，我们可以更加精确。

通过增大参数COEFFICIENT_BOXCOUNT和COEFFICIENT_SANDBOX的大小，可以使计算需要的图片更小，来省掉后面的值。

为了保证图像的一致性，我们直接手动去除2000Boxcount.txt和2000Sandbox.txt中处于后面的点的数据。因为如果重新生成图像，我们随机数种子会改变，图片会有所变化。

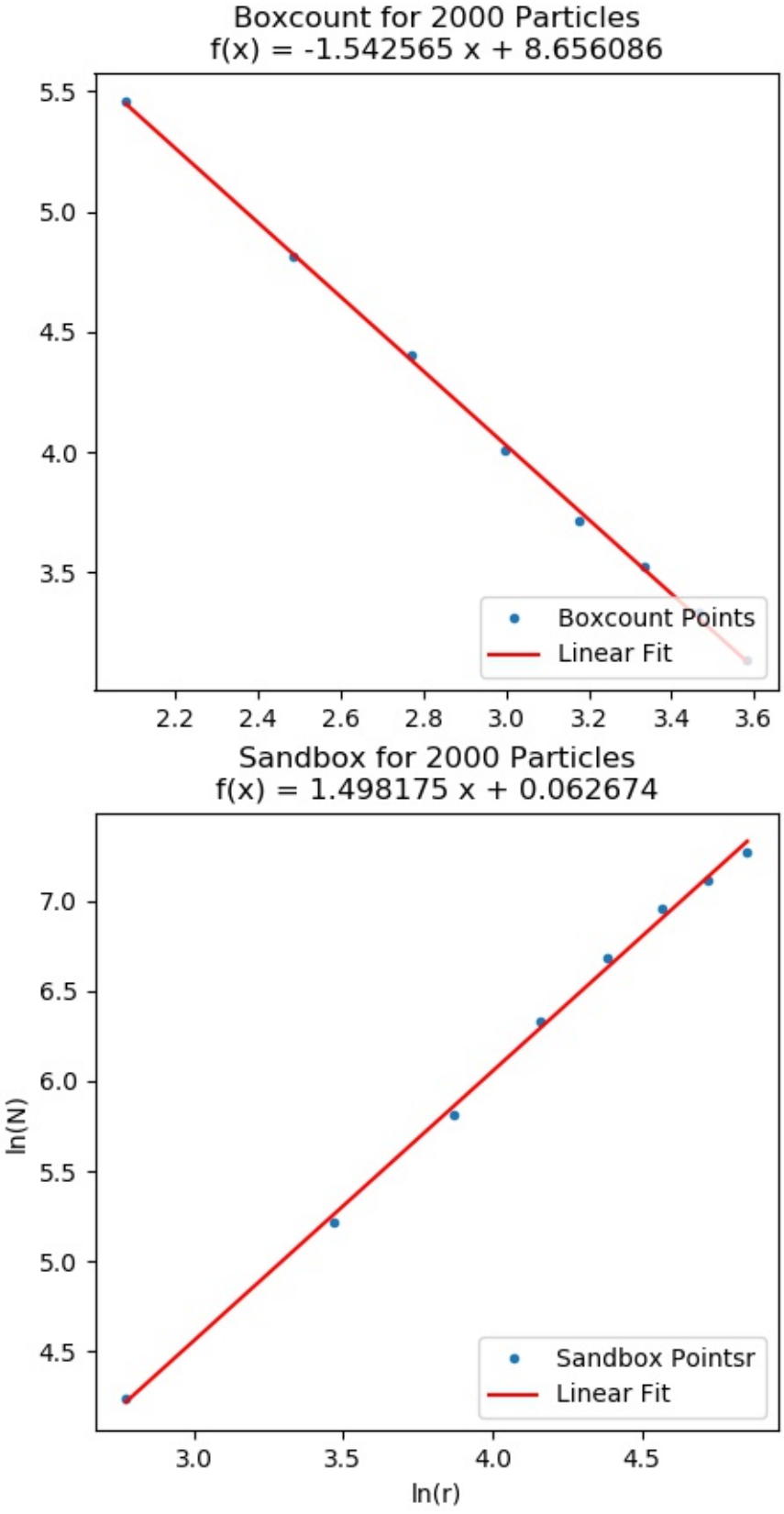


图 7: 2000个粒子求分形维数(修改后)

分形维数近似为1.542和1.498。

可以看到，两个计算的结果更加靠近真实值，但仍旧存在较大的误差。

这是因为粒子数很少，两个方法对于粒子数少的情况存在很大的缺陷。

修改宏定义的两个参数COEFFICIENT_BOXCOUNT和COEFFICIENT_SANDBOX到15和9。我们设置更大的粒子数，比如10000，得到下图：

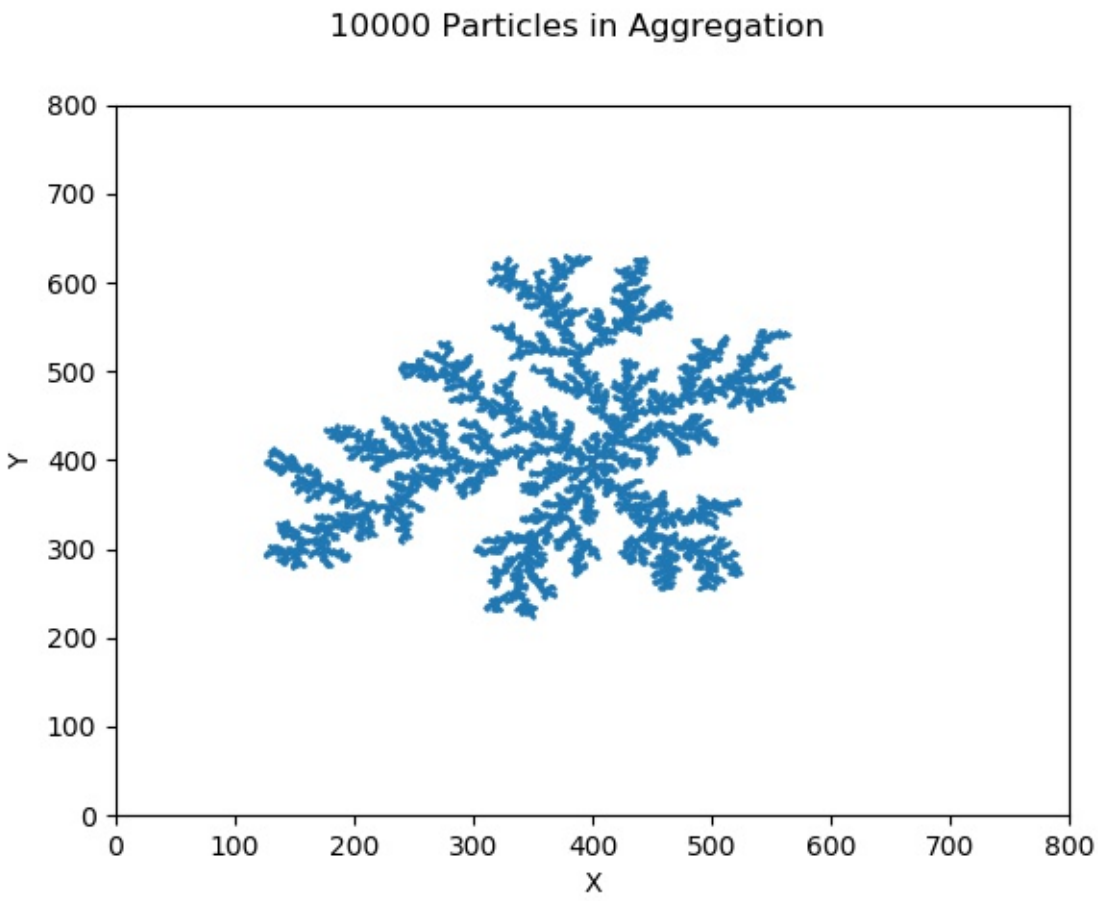


图 8: 10000个粒子

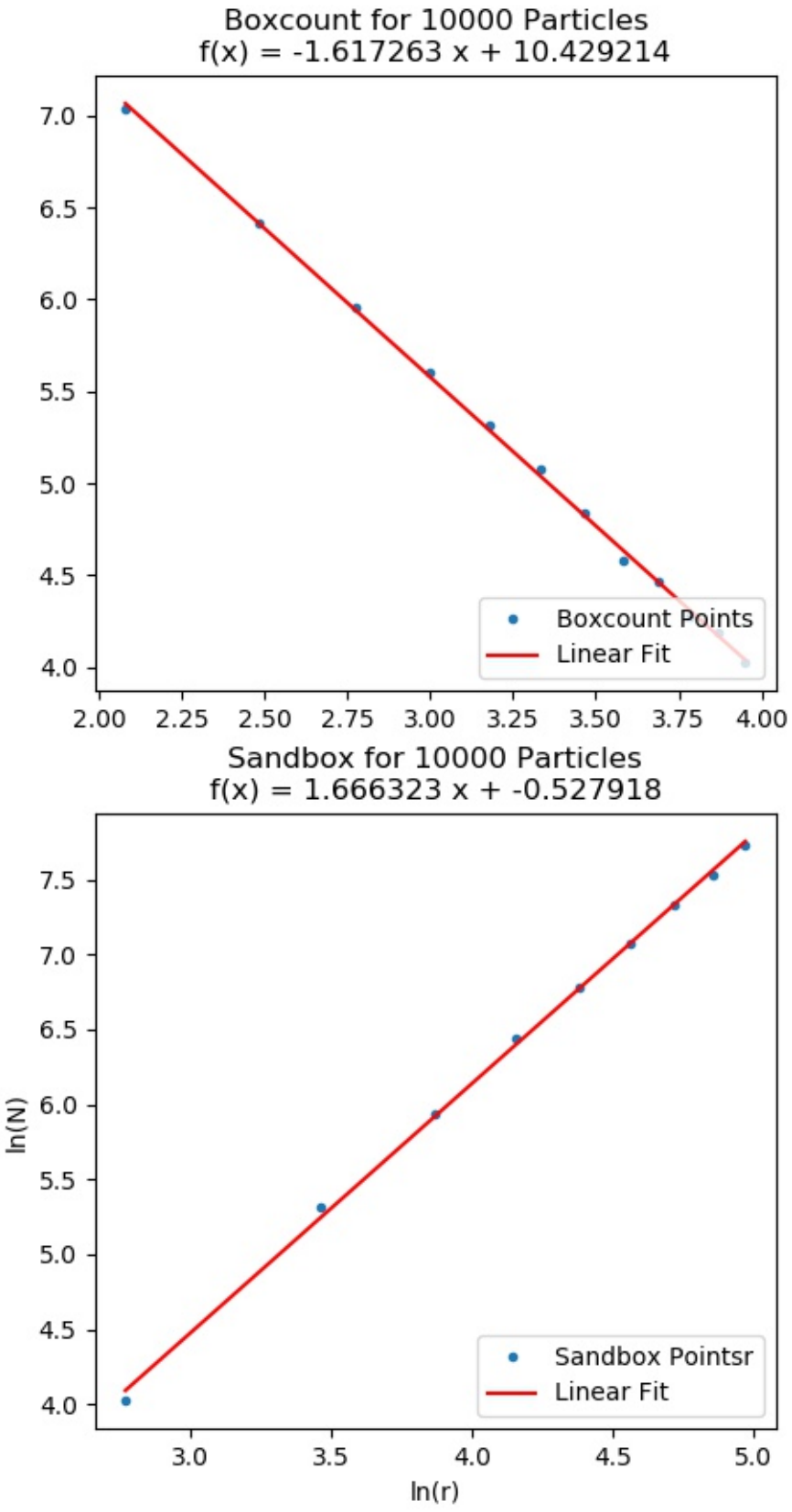


图 9: 10000个粒子求分形维数

这次求得的两个分形维数都处于正确的分形维数区间内，比较成功。

5 讨论

5.1 原理的讨论

本次实验用计算机来模拟薄膜生长的情况，可见计算机在物理学中的重要作用。

源程序中随机行走的程序限制了粒子运动的方向和距离。我们每次只能选择上、下、左、右四个方向中的一个，而且只能选择行走一个单位长度。显然实际中不存在这两种限制。

我们可以修改随机行走与判断是否并入团簇的函数来实现更加真实的情况。例如随机生成运动的角度和距离，虽然计算机并不能取到实数范围内的所有数，随机数生成也有周期性，但这样将更加精确地模拟随机行走。

同时，由于距离不再是整数，我们判断是否并入团簇的条件也需要修改。比如设置成当粒子离团簇的最近距离达到一定距离之内时，就认为粒子可以聚集到团簇上。

通过以上的办法修改，可以更加符合真实的情况。但由于我们本次实验模拟的程度已经非常高，这样修改将会极为复杂，且要求极高的计算速度，也不一定能将模拟得到的模型改进得多精确。因此，我们运用本次实验的原理即可。

5.2 算法的改进

由于本人的计算机知识并不是很丰富，下面的讨论可能会欠妥，源代码的编写也肯定会有很多需要改进的地方，希望以后学习中能够不断完善。

1. 二维数组的声明：

   声明二维数组时，需要定义在main函数的外面，即全局变量。因为如果定义在main函数里面，数组的空间是存放在目前线程的栈中，能分配到的最大内存不够大。因此定义的二维数组的两个维度都不能太大，否则会造成内存溢出。因此如果定义在main函数中，就会很难满足本实验中需要大数组的情况。此外，类型设置不是int而是unsigned char。这样做的目的是为了减少二维数组存储的内存。

2. 动态边界的设置：

   在图3中我们遇到了粒子贴着生成新粒子的边界聚集的情况。这是因为我们把生成粒子的边界设置成不变的，当粒子数很大时就会出现这种情况。我们改进的方法就是将生成新粒子的边界设置成可变的，即如果有粒子聚集到团簇上的时候处于生成新粒子的边界外时，我们就将生成新粒子的边界和逃逸的边界同时扩大。通过修改，我们得到图4所示的比较好的模拟结果。但也因此会增大运算时间，因为每个粒子聚集时都要判断一次是否达到了生成新粒子的边界。

3. 运算时间长的缺陷：

   程序运算时间方面有几个比较重要的缺陷，但由于缺乏相应的计算机知识，目前不知道如何解决。

   第一个就是粒子数大时需要花很长时间，一个重要的原因就是盒计数法是四层for循环，在main函数里调用盒计数法也存在一层for循环，这样时间复杂度就是n的五次方，但我还没有想到如果提升这个算法的性能。如果只能选择一种方法，我绝对不会选择盒计数法。

第二个就是判断粒子是否达到生成新粒子的边界，每个粒子都判断一下花了很长时间，因为大多数粒子并不会达到边界。

第三个，起始时生成粒子的边界、逃逸距离、图片大小等都会实际影响计算速度。另外有一点比较奇怪的是，这三个参数中如果我们设置很大的话，将会导致程序一直在计算，却得不到结果。我不知道是程序内存或者其他的问题，还是时间需要很长才能计算出来，至少我将这三个参数分别设置成700、800、1000，设置最大粒子数为20000时，运行了一个小时也没有得到结果。

4. 随机数的生成：

   需要在main函数里添加语句srand(time(NULL));来根据系统时间生成随机种子。否则会导致生成的随机数一样。这个语句也不能放在for循环中，因为程序运行太快，系统时间可以认为不变，这样也将得到一样的随机数。

5. 判断是否并入团簇的函数：

   判断粒子是否并入团簇的函数，我们运用的是八个方向。如果只用上、下、左、右四个方向，生成的图像就会有较大区别，因此我们设置了八个方向，更加符合实际情况。

5.3 结果的讨论

我们本次实验采用盒计数法和Sandbox法计算分形维数，但这两种方法都有非常严重的缺陷。

非常重要的一点是，应该选取多大的画布，但老师的讲义中并没有严格地定义两种方法。我为此想了很久，因此源程序中设置了两个宏定义COEFFICIENT_BOXCOUNT和COEFFICIENT_SANDBOX来控制两种方法的画布大小，通过不断运行程序并求结果来得到一个好的画布大小。

对于画布大小的影响，我们可以这样想：假设画布选取得无穷大，在盒计数法中，我们存在粒子的盒子数总是一个，即对应的是r非常大的请情况，这时候曲线已经变成直线，得到的分形维数是0；同样，在Sandbox法中，我们得到的粒子数总是总粒子数，这样也对应r非常大的情况，得到的分形维数也是0。可见，用这两种方法求分形维数时，首先要确定合适的画布大小，但在这两种方法的介绍中并没有提及如果选取，这是我的一大困惑。因为我是已知正确的分形维数来确定画布大小，还可以通过不断试验来确定比较好的大小。倘若我要求一个任意图形的分形维数，不同画布大小得到的结果相差比较大，我将无法保证我的结果的正确性。

另外认真分析这两种计算分形维数的方法，我们可以想到，对于多中心的情况，我们用盒计数法可能更好；而对于单中心的情况，我们用Sandbox法可能更好。

我们生成的DLA模型来计算分形维数时，得到的结果并不是一成不变的，这也符合正常的统计涨落。