REPORT

Runze Wang

1. Question

Simulate 2D Diffusion Limited Aggregation (DLA) and Dielectric Breakdown Model (DBM) patterns, and discuss image's fractal and dimensional features.

2. Method

2.1 DLA

Diffusion Limited Aggregation

Simulation Rules

A 2D square lattice is taken, and a particle is placed at the center of the lattice as the seed for growth. Another particle is released from a circular boundary far enough from the origin of the lattice and allowed to perform a random walk. The particle will eventually collide with the nearest neighbor position of the seed, at which point the particle will stick to the seed and stop moving. If the particle reaches the boundary of the lattice, it is considered a useless trajectory and discarded, and a new particle is generated. Therefore, useful particles that stick to the seed form a growing aggregate cluster.

Definitions:

- The maximum distance of the current aggregate cluster is **d**.
- The cluster radius is $r = \max 20, d$.
- The edge length of the 2D square matrix is N.
- The particle stickiness is $stickness \in (0, 1)$.
- The radius of the generating circle is $R = 2r$.

Algorithm:

1. Randomly generate a particle, x.
    - If $R < N/2$, generate a particle randomly on a circle with radius R.
    - Otherwise, generate a particle randomly on the boundary of the square lattice.
2. Let particle x perform a random walk and check if there are any aggregate particles in the eight surrounding points.
    - If the particle collides with the boundary of the lattice, return to step 1 and generate a new particle.
    - If there are aggregate particles nearby, determine whether to join the cluster or continue to perform a random walk based on the stickiness of x.
3. When the particle joins the cluster, update the cluster radius, r.
4. Return to step 1, or end the program when the cluster is large enough.

2.2 DBM

The Dielectric Breakdown Model

Simulation Rules

The process described by this model involves a constant potential ϕ_0 at the central point of a dielectric material (insulator), which continuously breaks down the surrounding medium, forming a growing conductor.

The growth rate v of breakdown in the medium is a function of the gradient of potential ϕ, i.e., $v = f(\nabla_n \phi)$, where n is normal to the interface. The potential throughout the medium follows Laplace's equation, and the boundary conditions are that the potential is ϕ_0 on the occupied grid and 0 at a distance. The growth mode is to occupy an empty grid on the boundary with a probability equal to the growth rate. Mathematically, this can be described as:

$$GrowthRate :: v_{i,j} = n|\phi_0 - \phi_{i,j}|^\eta \tag{1}$$

$$SelectionProbability : p_{i,j} = v_{i,j} / \sum v_{i,j} \tag{2}$$

The probability is summed over all unoccupied grids on the boundary, and a grid is selected for occupation based on this probability. After occupying a new grid, the boundary conditions change, and the potential distribution, growth rate, and selection probability need to be recalculated.

Definitions

- Grid state: `EMPTY` represents a free point, `CANDIDATE` represents a candidate point, and `FILLED` represents a grown point.
- Particle set `_particle_set`: stores a set of generated points, all of which have the state `FILLED`.
- Candidate set `_candidate_set`: stores a set of candidate points to be grown, all of which have the state `CANDIDATE`.

Algorithm

1. Using formula (1) and (2), randomly select a candidate point **x** from `_candidate_set` based on its growth rate.
2. Add the candidate point **x** to `_particle_set`, change its state from `CANDIDATE` to `FILLED`, and set its potential to ϕ_0.
3. Add all surrounding `EMPTY` points of **x** to `_candidate_set`, changing their state from `EMPTY` to `CANDIDATE`.
4. Remove **x** from `_candidate_set`.
5. Recalculate the potential of particles in `_candidate_set` based on Laplace's equation and the new boundary conditions.
6. Go back to step 1, or terminate the program when the cluster is large enough.

Note: In actual code implementation, it is not necessary to label points with states. The state can be determined based on whether a point is in `_particle_set` or `_candidate_set`.

Numerical Solution Method for Laplace's Equation:

The Laplace equation can be expressed as

$$\nabla^2\phi(x,y)=0,\phi|_{\partial D}=f(x,y)$$

For numerical solution, it can be written as

$$\phi_{i,j}=(\phi_{i-1,j}+\phi_{i+1,j}+\phi_{i,j-1}+\phi_{i,j+1})/4$$

In reality, the potential values of the surrounding points are not known, only the potential values on the boundaries are known. Therefore, particles at position (i, j) are allowed to randomly walk until they encounter the boundary, at which point the potential value on the boundary, denoted as $f(x,y)$, is recorded. After multiple random walks, the average potential value is taken as the potential value at point (i, j):

$$\langle\phi_{i,j}\rangle=\frac{1}{N}\sum_{n}^{N}f_n(x,y)$$

2.3 DBM_FAST

Background

For actual dielectric breakdown models, the main bottleneck of the algorithm is to re-calculate the potential size of **all candidate points** after each iteration. When the boundary range **N** is large enough to grow **n** points, the time complexity required is $O(n*N^3)$ 1. For a grid point with a side length of $N=300$, the time will be very slow, and it takes about 18 minutes to grow **n=300** points.

Therefore, based on the idea of a paper about **fast Laplace growth simulation** 1, I propose the following improvements.

Idea

Each "grown" point is insulated, that is, the charge is not redistributed but fixed at the lattice point. For the potential of candidate points, it is a linear superposition of the electric potential of these `FILLED` growth points.

Consider that the center potential of the boundary is 1 and the potential at infinity is 0. Then, each point charge can be treated as a positive charge, and the potential is

$$\phi(r)=\frac{R_1}{r}$$

where the length of each small lattice point is h, and the radius of the point charge is $R_1=h/2$.

Therefore, for the `CANDIDATE` point i, its potential is the superposition of the electric charge j of all `FILLED` points:

$$\phi_i=\sum_{j=1}^{n}\frac{R_1}{r_{i,j}} \tag{3}$$

The growth rate should also be correspondingly corrected. Let the maximum potential of the `candidate` be ϕ_{max}, and the minimum potential be ϕ_{min}:

$$v_i = \left(\frac{\phi_{max} - \phi}{\phi_{max} - \phi_{min}} \right)^{\eta} \tag{4}$$

The reason why it can be accelerated is that each time it is iterated, the previous potential can be used for iteration, that is:

$$v_i = \left(\frac{\phi_{max} - \phi}{\phi_{max} - \phi_{min}} \right)^{\eta} \tag{4}$$

r_{t+1} is the distance between the new position of the `FILLED` point and each `CANDIDATE` point.

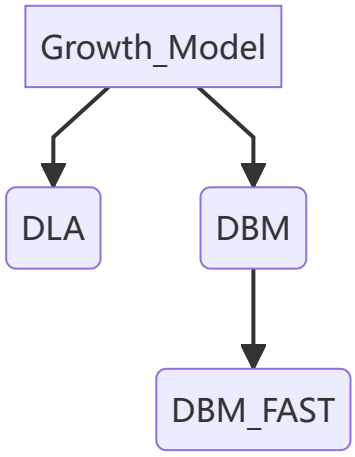
Algorithm

1. According to formula (4), the growth rate of the candidate point is a probability. After normalization, randomly select a candidate point **x** from the `_candidate_set`.
2. Add candidate point **x** to `_particle_set`, and its state changes from `CANDIDATE` to `FILLED`.
3. Remove **x** from `_candidate_set`.
4. Iterate the potential in `_candidate_set` according to formula (5).
5. Add `EMPTY` points around **x** to `_candidate_set`, change the state from `EMPTY` to `CANDIDATE`, and calculate the potential of the new candidate point according to formula (3). In one iteration, no more than 5 points like this.
6. Return to step 1; or when the cluster is large enough, the program ends.

2.4 Code Framework

Due to the need for efficiency in the code involving a large number of operations to check whether points are in a set, we have chosen `set` as the primary data structure. Its operations of searching with `in`, adding with `add`, and removing with `remove` are all with a time complexity of $0(1)$.

Since the three growth models have similar ideas, a class inheritance framework is designed.



The lower-level classes inherit the interface of the upper-level classes, reducing redundant code.

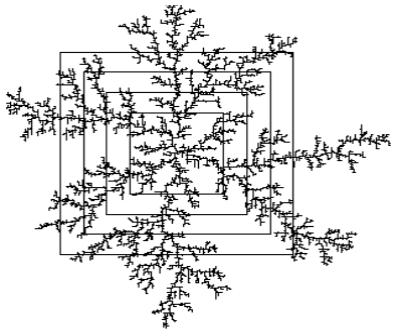
2.5 Fractal and Dimensional Features

Sandbox Method

Using the Sandbox method to calculate the fractal digits, the formula is

$$N(r) \sim r^D$$
$$D = \frac{\ln N}{\ln r} + C$$

N is the number of pixels in the square box, r is the side length of the box, and C is other constants. The statistical process is shown in the figure below



Increase the side length of the box according to the power of $\sqrt{2}$, count the number of internal points, and get a logarithmic coordinate graph, and calculate the slope to be the number of fractal digits D

密度-密度相关函数法

平面上分形图形的密度-密度 相关函数定义:

$$C(r) = \left\langle \sum \frac{\rho(r')\rho(r'+r)}{N} \right\rangle \sim r^{-\alpha}$$

$\rho(r')$ 是图形的密度函数，有图形象素处为 1，无 图形处为 0；N 是总象素数。C(r) 的几何意义是：原始图形和平移 r 后的图形重叠部分的象素数和全部象素数的比值，即在相距 r 处发现另一象素的概率。在实际计算中除了对 N 个不同的 r′ 求平均之外，还对不同方向、 长度相同的 r 求平均

特例: 固定 r′， 并把它取为图形的中心，即r′=0，此时

$$C(r) = \left\langle \sum \rho(0)\rho(r) \right\rangle \sim r^{-\alpha}$$

则要做的就是按照 $\sqrt{2}$幂次增长圆的半径，统计内部重叠点数，绘制出对数坐标，得到斜率 α

在C(r) 在回转半径R内积分，R 足够大时，积分值很接近于和图形总象素数 N 成正比

$$\int_0^R C(r)\mathrm{d}^{dim}r \sim N$$
$$N \sim R^{dim-\alpha}$$
$$D = dim - \alpha$$

dim是欧氏空间维数，此题中为2；D是分形维数。

3. Experiment

3.1 DLA

如下图所示，考虑了黏度分别为 **0.01,0.1,0.5,1**的DLA模型，生长了3000个点

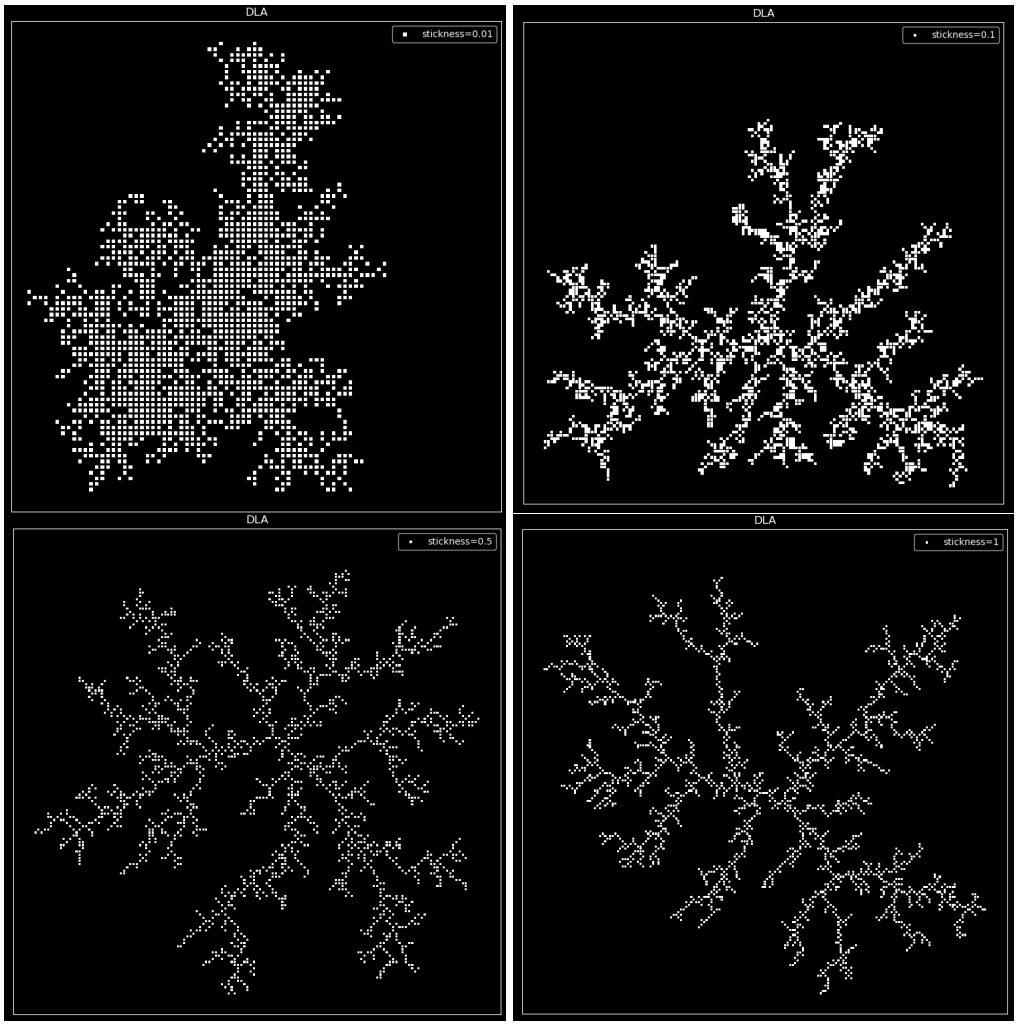


图1 DLA不同stickness的对比（3000点）

DLA分析：

由图可见，随着黏度的提高，DLA生长的图像会愈发稀疏，当黏度为1时，粒子进入一个沟槽之前，很有可能碰上一根在外的触须，因而粒子无法进入沟槽内，形成屏蔽效应。这一结构反映出生长过程的特征，即越是尖端处生长得越快，从而形成枝蔓向外延伸，越是平坦处生长得越慢，从而出现沟槽中的空隙疏松结构。这样的形貌只有当粒子粘结到集团而无任何优先选择的方向时才会出现

3.2 DBM

如下图所示，考虑了eta分别为 **0，3，6，10** 的DBM模型，由于代码效率受限，只生长了300个点

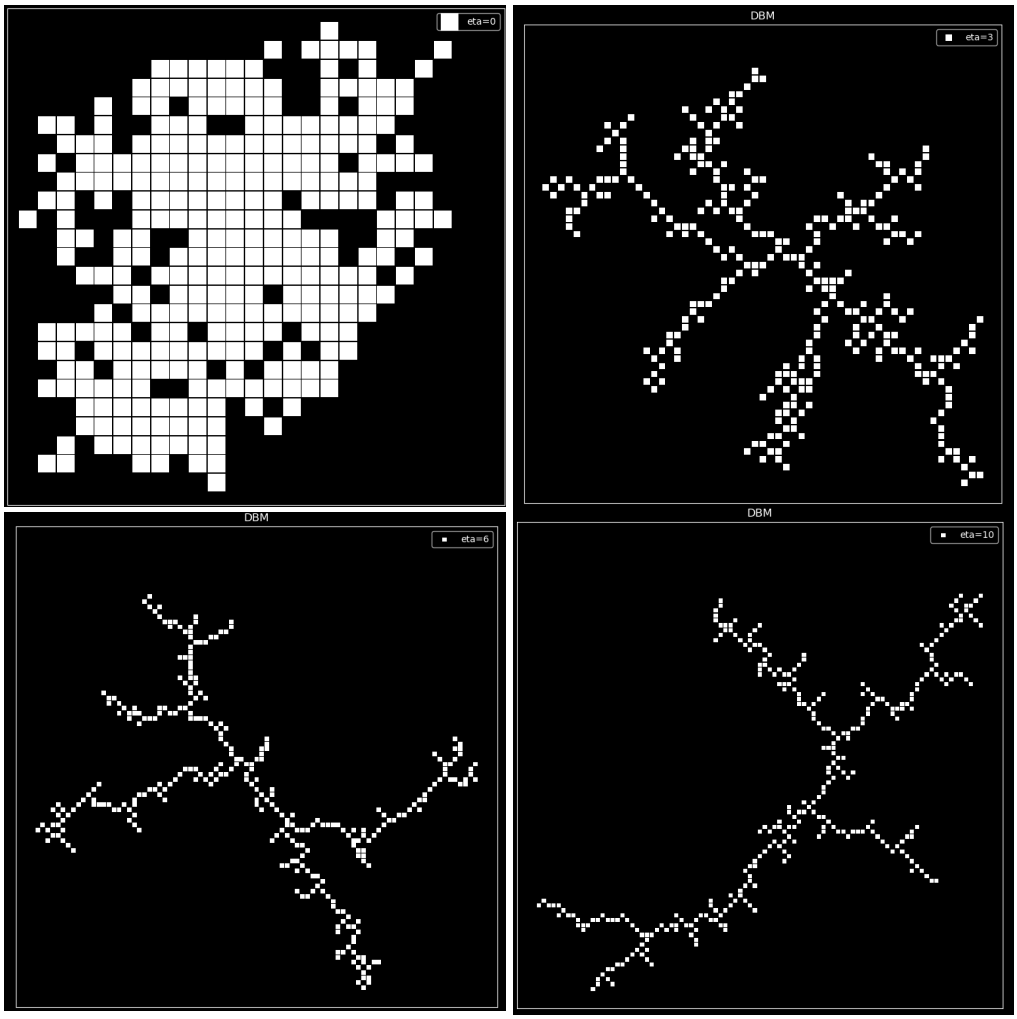


图2 DBM不同eta的对比（300点）

DBM分析：

由图像可见，随着 η 的增大，根据生长速率 $v_{i,j} = n|\phi_0 - \phi_{i,j}|^{\eta}$ 为概率而选择，会导致图像的分支逐渐变少，图像从平面的二维均匀平面逐渐收敛到一维图像，如同从球状闪电到线状闪电一样。

通过分析也可以很容易解释上面的图像，当 η 变大时，对于较大的势能梯度 $|\phi_0 - \phi_{i,j}|$，其比重会更加明显，从而导致依概率选择时，更倾向于选择梯度变化更大的方向，故更容易呈现出分支较少的线形图像。

3.3 DBM_FAST

改善了算法后，得到eta分别为 **0，3，6，10** 的DBM模型，生长3000个点，有以下图像

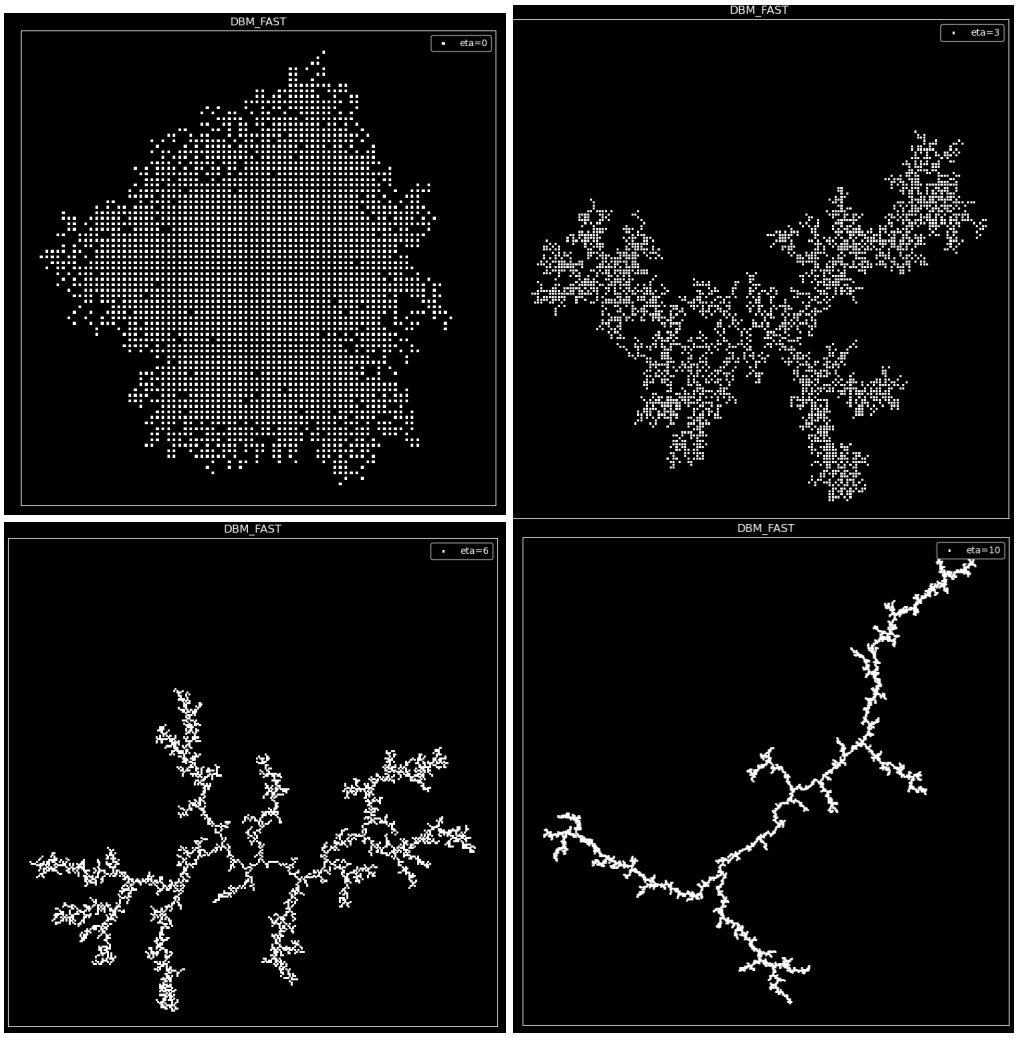


图3 DBM FAST算法不同eta的对比（3000个点）

DBM FAST分析：

由图像可见，随着 η 的增大，图像也同样得从二维的平面逐渐收敛到一维的线性，和DBM的趋势一致，从而可以说明，我们改善后的算法没有改变图像的本质区别，并且大大加速了算法的速度。

与DBM 对比

下图是300个点的DBM FAST算法

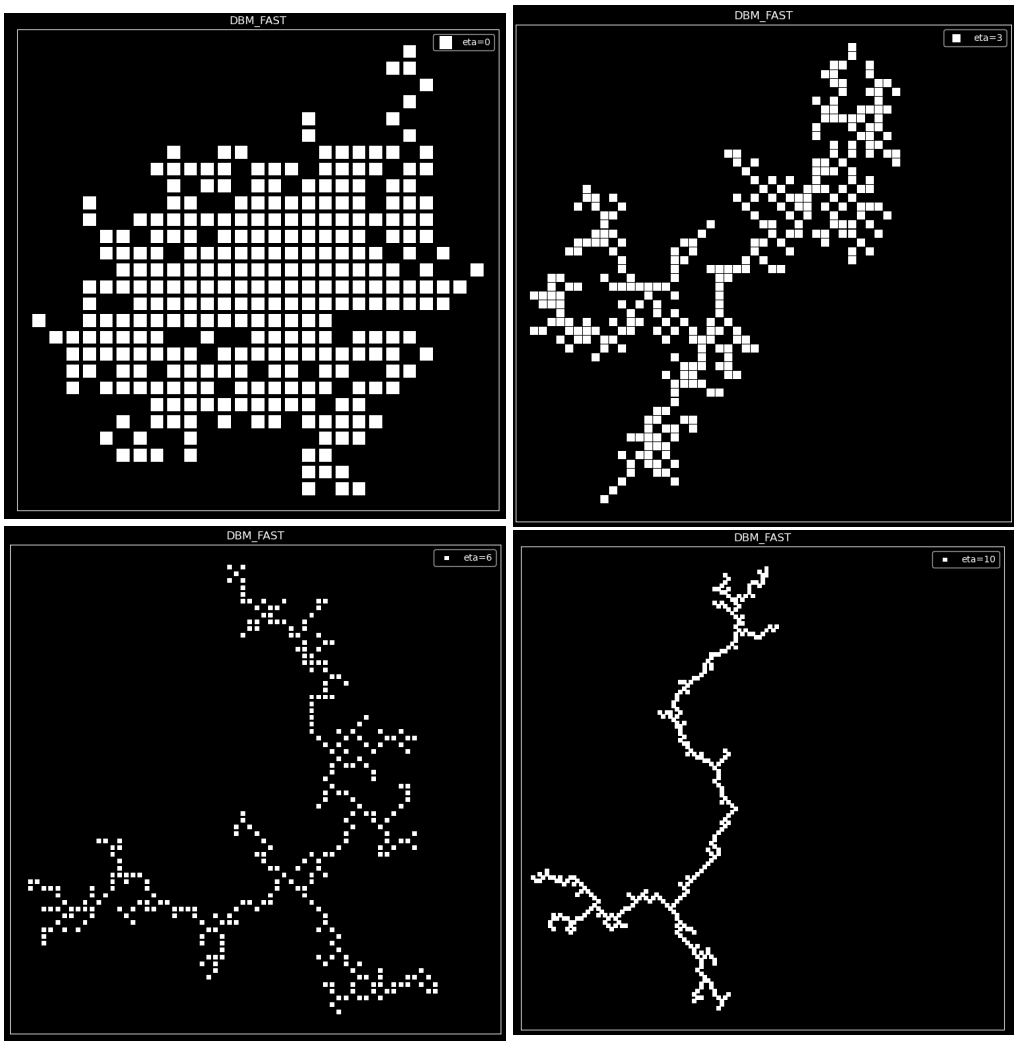


图3 DBM FAST算法不同eta的对比（300个点）

可以更好的与DBM图像对比，二者几乎一致。

并且时间得到加速

```
DBM 300 points time cost: 18.0min, 30.58253574371338sec
DBM FAST 300 points time cost: 0.0min, 0.7947542667388916sec
```

几乎加快了1000倍

与DLA对比

二者在一定参数选择下都呈现出树状生长的形状。

可以看到，DLA算法始终保持了较多的分支数目，不会随着参数的改变，而逐渐收敛到一维。DBM算法，会随着 η 的改变，点与点相关性会变得明显，收敛到一维而不再保持分支现象。

4. Summary

本次实验对DLA和DBM模型进行对比，分析二者的不同。

提出了加速Laplace数值算法的方式，并通过实验验证其有效性。

在实验过程中，采取了良好的数据结构模式 **集合set**，提高了程序速度。

5. Reference

[1] 快速Laplace生长模拟