Hanoi University of Science and Technology
School of Information and Communication Technology
===

Scientific Computing

Project Report

Diffusion-Limited Aggregation (DLA)

GROUP 9

Trần Đức Nam	20215228
Nguyễn Chính Minh	20215224
Trịnh Giang Nam	20215229
Nguyễn Trọng Huy	20210451

Class: 131111

Teacher: Prof. Vu Van Thieu

SOICT, semester 2022.2

Hanoi, July 2023

TABLE OF CONTENTS

I. INTRODUCTION

1.1 Diffusion Limited Aggregation

Diffusion Limited Aggregation (DLA) is a process used to desribe the random scalar growth of particles.

This theory, proposed by T.A. Witten Jr. and L.M. Sander in 1981, is applicable to aggregation in any system where diffusion is the primary means of transport in the system. DLA can be observed in many systems such as electrodeposition, Hele-Shaw flow, mineral deposits, and dielectric breakdown.

1.2 Application of DLA

Diffusion Limited Aggregation (DLA) is a model for non-equilibrium growth, where growth is determined by diffusing particles. It can be a model for a Bacillus subtilis bacteria colony in a petri dish. The idea is that the colony feeds on nutrients in the immediate environment, that the probability of growth is determined by the concentration of nutrients and finally that the concentration of nutrients in its turn is determined by diffusion.

1.3 Problem description

Among many applications of DLA, this report will present the simulation process of the development of a virus-infected organism in a finite environment containing food. The virus will continuously consume food and grow.

II. THEORETICAL BASE

2.1 Theory

During the description process, bacteria will develop from an initial cell. The growth of bacteria in a food environment is probabilistic and depends on the food concentration of the surrounding cells.

The food concentration in the environment also continuously changes, and cells containing bacteria will consume all the available food. The difference in food availability will be altered by the propagation equation.

The propagation equation simulates the change in food concentration. Since this change is much faster compared to the virus's growth rate, the propagation time can be considered very small. We will use a **time-independent propagation equation** to determine the food concentration.

$$\frac{\partial c}{\partial t} = \mathcal{D}\nabla^2 c$$

Where $\mathcal{D}$: Spreading coefficient.

c : Concentration.

Since it is a time-independent equation, the derivative of concentration with respect to time is assumed to be zero.

$$\nabla^2 c = 0$$

The above equation can be solved directly or using the iterative method. Iterative method Successive Over Relaxation will be covered later in the report.

2.2 SOR Method (Successive Over-Relaxation method)

We did a white-black order. For updating black points, since we only need the white point, only the white points need to be exchanged. So white-black ordering means that we first update all white points in parallel, followed by a parallel update of all black points. Within each domain updating can be done with the row-wise ordering scheme.

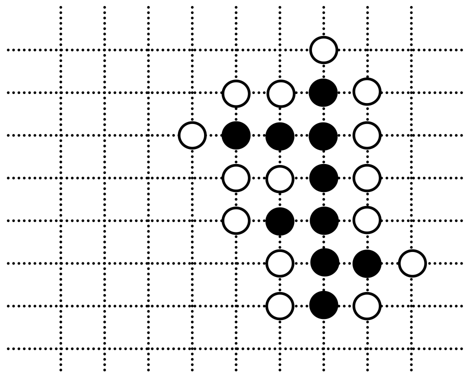


Figure 1. White-black ordering

By mixing the Gauss-Seidel result with the current value, the SOR iterative method uses the following formula:

$$c_{l,m}^{(n+1)} = \frac{\omega}{4}\left[c_{l+1,m}^{(n)} + c_{l-1,m}^{(n+1)} + c_{l,m+1}^{(n)} + c_{l,m-1}^{(n+1)}\right] + (1-\omega)\, c_{l,m}^{(n)} .$$

Figure 2. Formula of SOR iterative method

where w is the correction parameter, $(1 < w < 2)$ and in this case, we choose $w = 1.5$.

2.3 DLA Problem application

The basic algorithm to simulate the DLA process is as follows:

- Step 1. Solve the propagation equation to get the food concentration matrix. The concentration matrix will be generated by the SOR method.

- Step 2. Let the virus reproduce (according to the probability). Each cell next to the bacteria will have a probability of being infected, that probability is proportional to the concentration of food there.

- Step 3. Go back to step 1.

Probability of the virus developing:

$$P_{((i,j)\in Food \rightarrow (i,j)\in Virus)} = \frac{(c_{i,j})^{\eta}}{\sum_{(i,j)\in Food}(c_{i,j})^{\eta}}$$

The exponent η determines the shape of the virus reservoir, usually taken from 0.5 to 2. In the report, η will be tested with several different values.

III. INSTALLATION

In this project, the programming is installed on C language, GNU compiler via TMD GCC tool on windows environment. And we use Matlab to illustrate the results in graph.

3.1 Data modeling

Virus is modeled by VirusPlace on C:

```
typedef struct Point {
        long x;
        long y;
}Point;
```

In which **x** and **y** are respectively the row and column coordinates of the virus.

The **virus[]** array contains a list of the viruses that have appeared, initialized with a single virus.

The variable **nVirus** stores the current number of viruses.

The **candidate[]** array contains a list of candidates that can potentially be infected by the virus in the current step, and the variable **nCandidate** holds the number of candidates.

The **c[][]** array represents the remaining food in the area, initialized to 1, except for a single cell containing the first virus.

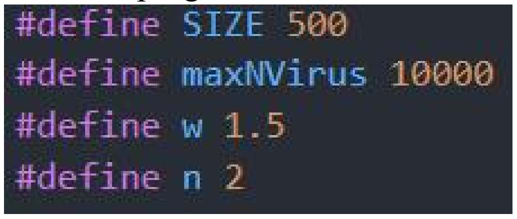
The **chance[]** array holds the probabilities of the candidates, and the number of elements is also equal to **nCandidate**.

The **grow[][]** array is used to mark the positions of the viruses.

3.2 Program Design

<u>Input/output data:</u>

- Input: position of the first virus (we stored it in "input.txt" file)
- We define some constants in the program.



```
#define SIZE 500
#define maxNVirus 10000
#define w 1.5
#define n 2
```

Figure 3 Constants term

  - In practice, n should belong to range [0,2]

- Output: write matrix of the virus to “output.txt”

<u>Procedure:</u>

addVirus(u,v,x)	Marks the presence of a virus at cell (u, v), which is generated from candidate[x]. Additionally, adds the surrounding cells of (u, v) to the candidate array.
init()	Initializes the Grow[][] array as false. Initializes $C_{i,j}$ as 1. Initializes the first virus using addVirus().
sor()	Performs the Successive Over-Relaxation (SOR) iteration method to compute the entire C[][] matrix.
eat()	Mark the value c[i][j] which has virus to 0.
computeProbality()	calculate probability of each candidate.
growth()	generate random virus based on a random number between 0 and 1.
solve()	Continues looping until the desired number of viruses is reached: sor() -> eat() -> computeProbability() -> growth().
main()	Read the data, calls the solve function, and then print the data

3.3 Simulation in MATLAB

- Store the output file in “result.txt”
- Run the file “SOR.m” and see the illustration.

IV. RESULTS

Case 1: the first virus was in the center of the board.

We will test for the maximum number of viruses is 10000. We set up as following:

- Size of matrix: 500x500.
- Omega: 1.5 (w).
- The first virus was in the center of the board.

Now we change the value of η to see the difference.

- η = 0:

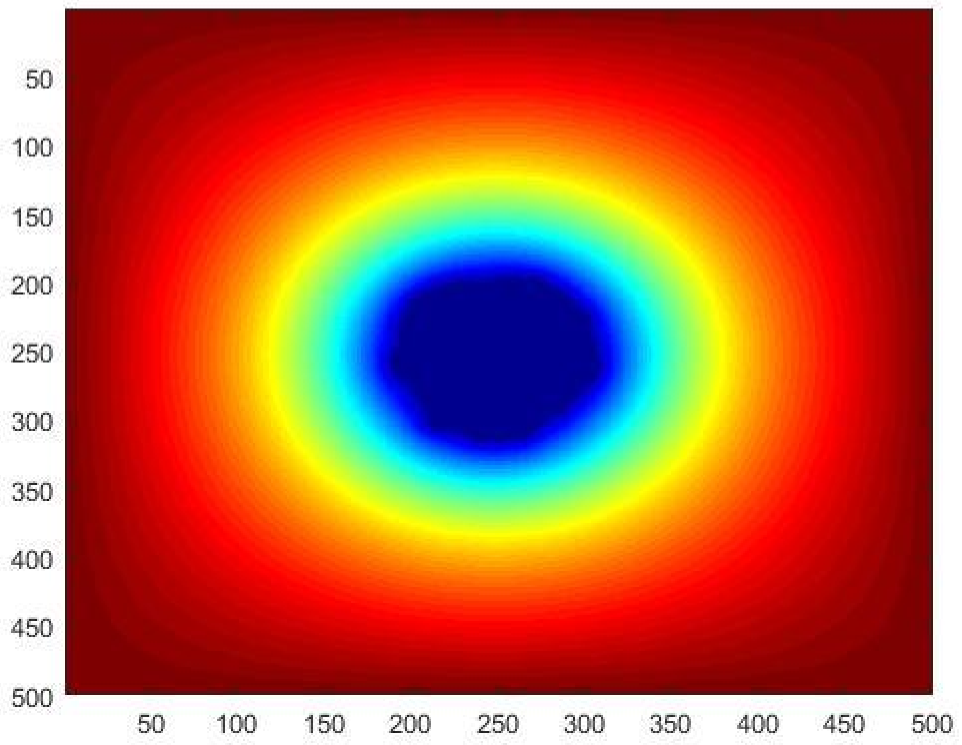
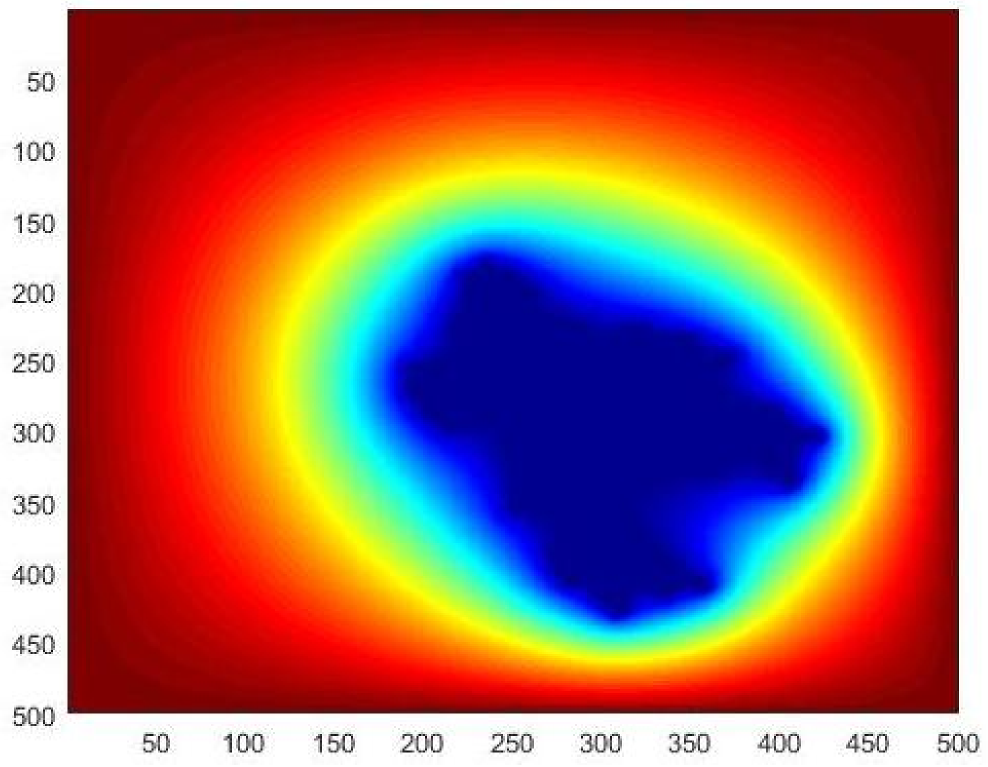


Figure 4. Testcase1: n = 0

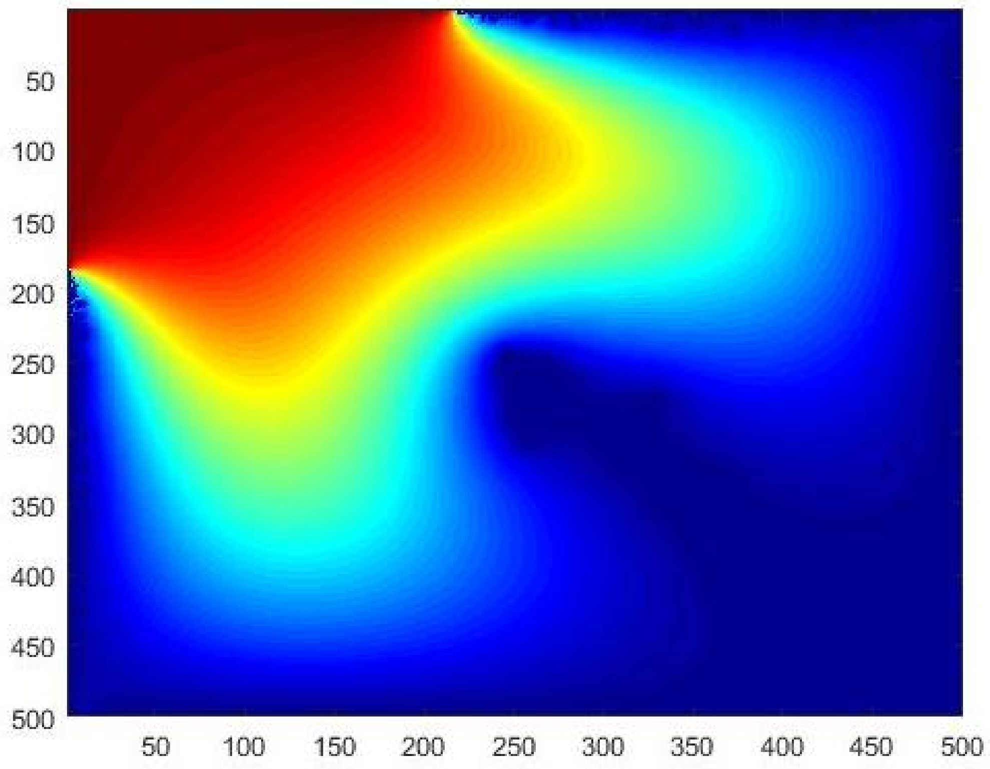
The positions that appear redder indicate a higher concentration of food. The positions that appear lighter red or yellow represent areas with less food. The darkest blue positions indicate the areas where viruses have appeared.

Now, let's change the value of η to see how the virus change their behaviour of growth under this circumstance.

- η = 1:

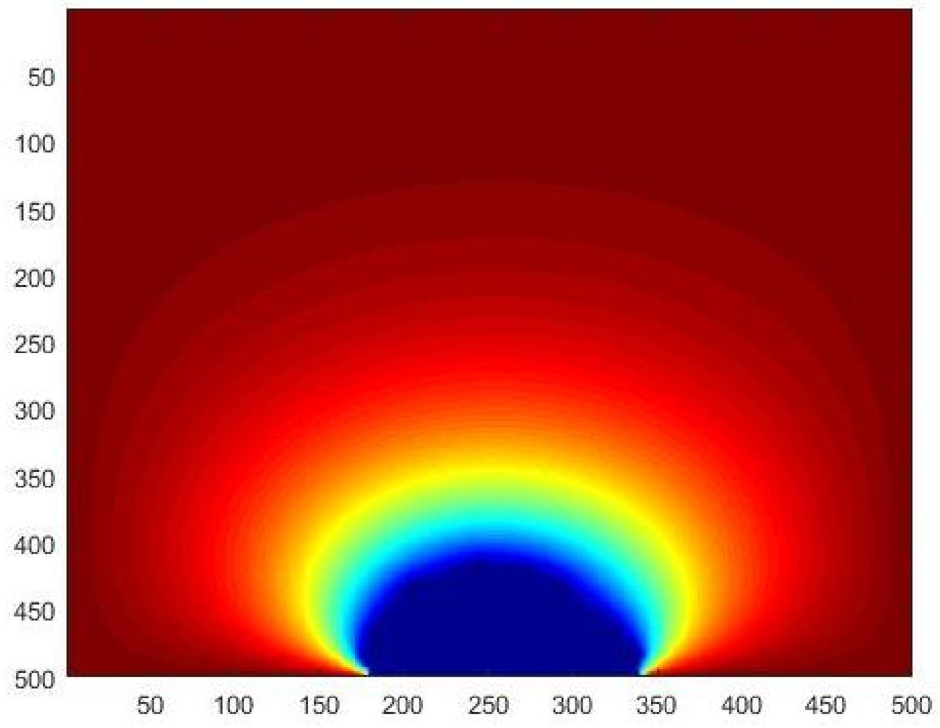

50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500

- η = 2:

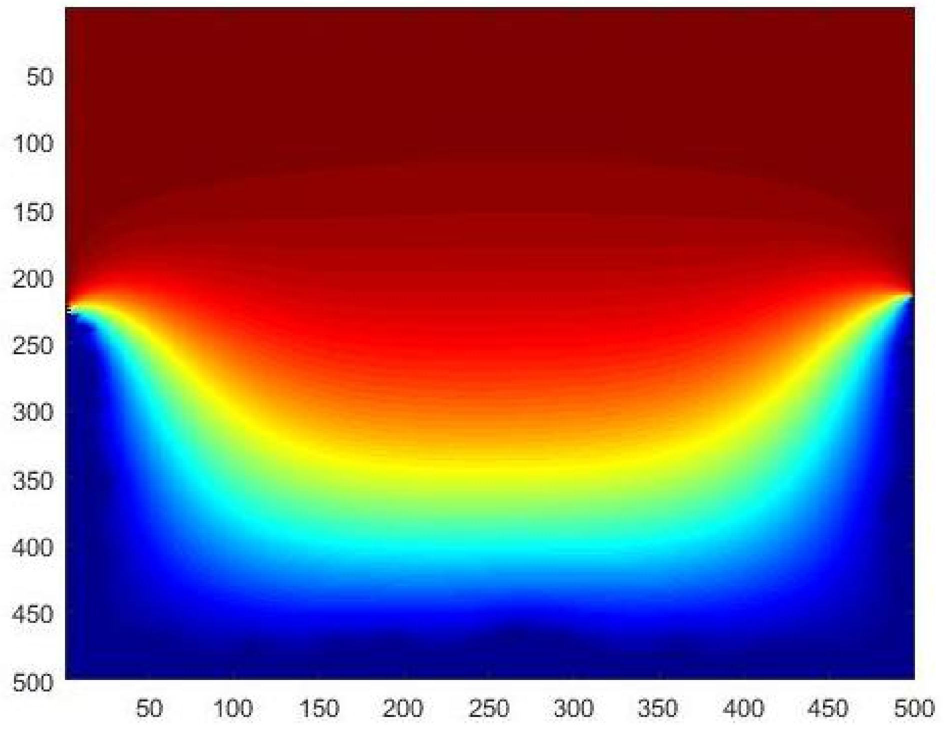

50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500

Case 2: the first virus was in the center of the bottom of the board.

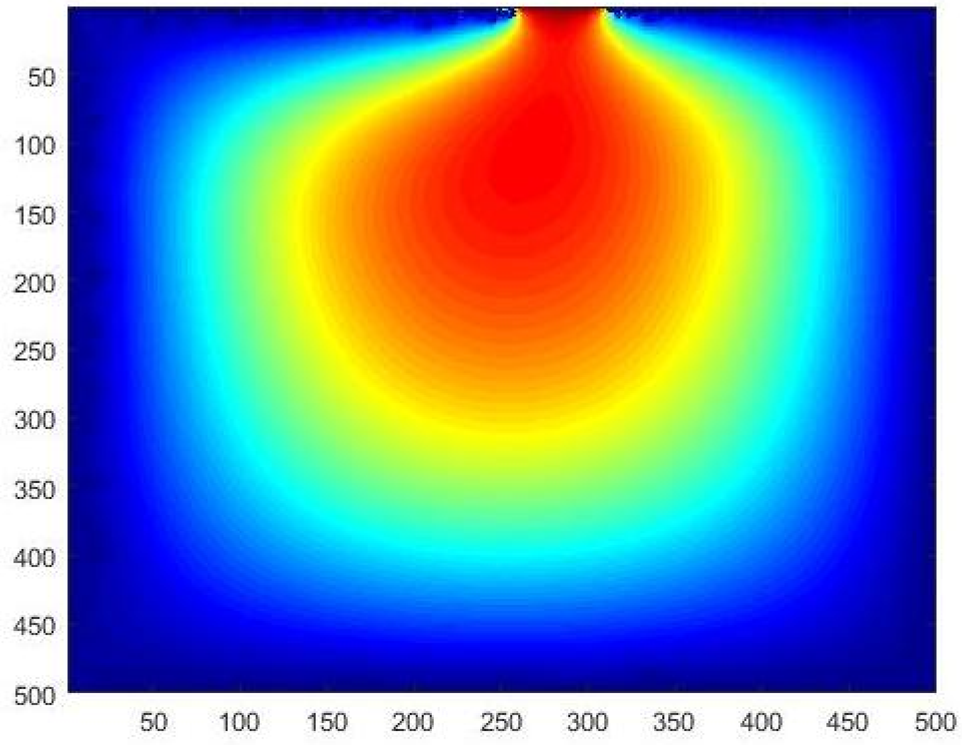
- η = 0:



- η = 1:

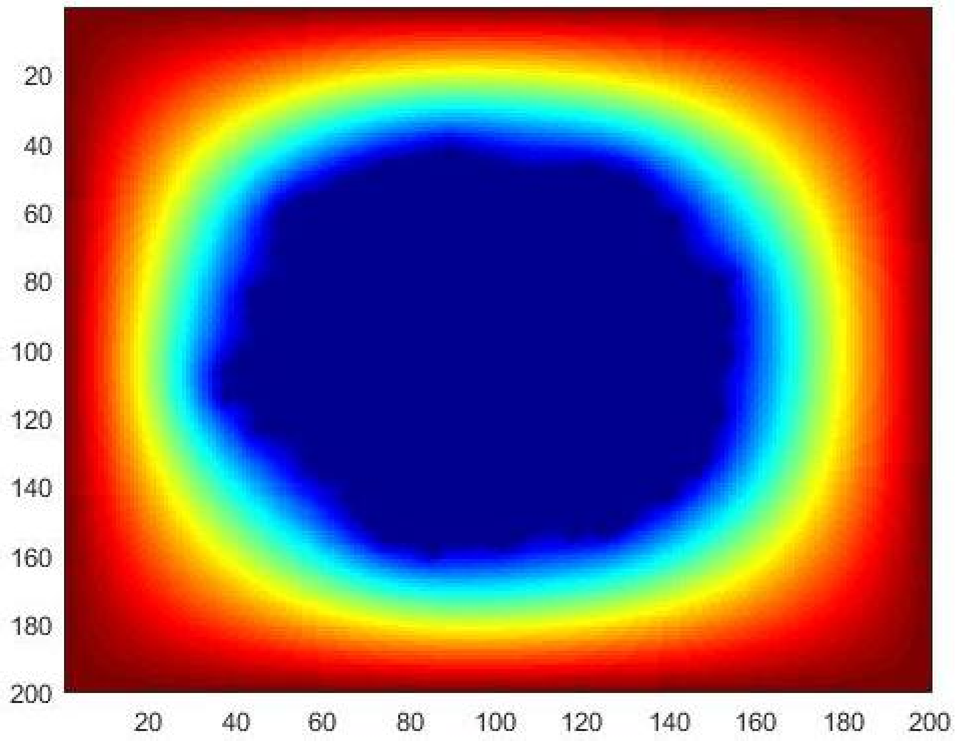


- η = 2:

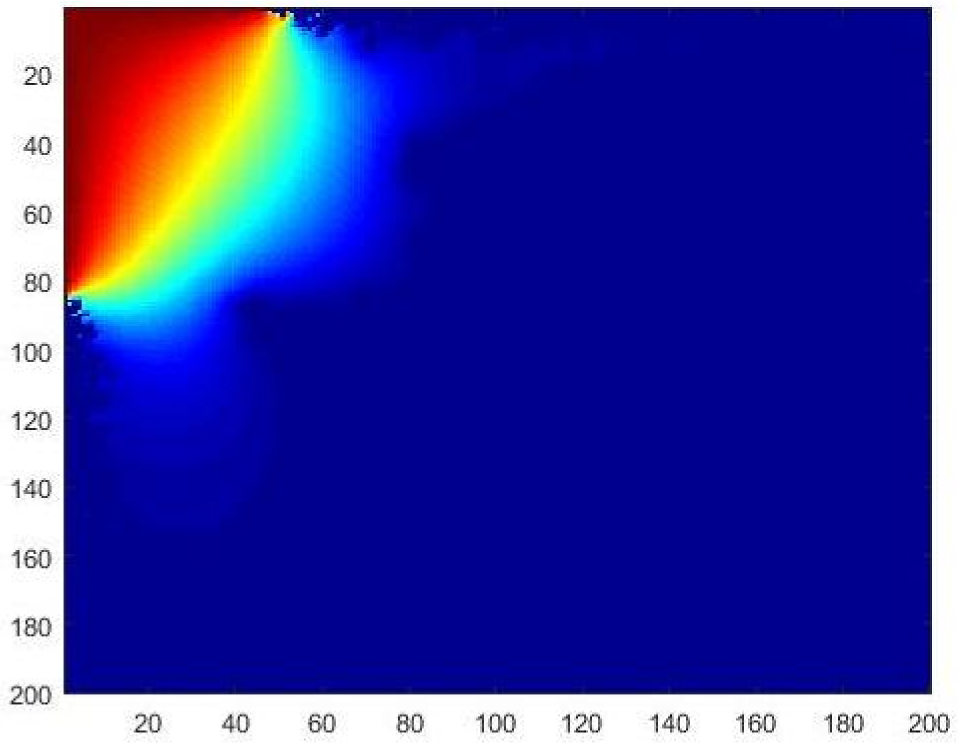


Case 3: Now we test again the case 1 but in the board of 200x200.

- η = 0:



- $\eta = 1$:



<u>*Comments:</u>

The exponent η influences the development of the virus.

- For $\eta=0$, the probability of new virus appearance is the same for all positions, resulting in a symmetric pattern.

- For $\eta=1$, the probability of a new virus appearance is proportional to the available food concentration at that location, which is reasonably aligned with natural circumstances. The pattern typically exhibits only a few branching developments.

- For $\eta=2$, the probability is proportional to the square of the available food concentration. Consequently, locations with higher food concentration will have a significantly higher probability of new virus appearance. As a result, the virus develops with very few branching extensions.

V. CONCLUSION

The report presented the principles and algorithms of the DLA simulation method using SOR Method, and also presented the direct test results as well as some our evaluations and conclusions.

REFERENCES

[1] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/Diffusion-limited_aggregation. [Accessed July 2023].

[2] T. A. Witten, L. M. Sander, "Diffusion-limited aggregation," in *PHYSICAL REVIEW B*, 1983.