Contents

[1.Introduction 2](#_Toc108650886)

[1.1 Members: 2](#_Toc108650887)

[1.2 DLA problem: 2](#_Toc108650888)

[1.3 Practical applications: 2](#_Toc108650889)

[1.4 The problem posed: 2](#_Toc108650890)

[2. Theories 3](#_Toc108650891)

[2.1 Theorem statement: 3](#_Toc108650892)

[2.2 SOR iterative method: 3](#_Toc108650893)

[2.3 Application to DLA problem: 4](#_Toc108650894)

[3.Set-up: 6](#_Toc108650895)

[3.1 Data structure: 7](#_Toc108650896)

[3.2 Program structure: 7](#_Toc108650897)

[3.3 Visualizing: 7](#_Toc108650898)

[4. Test results: 7](#_Toc108650899)

[5. Simulating: 7](#_Toc108650900)

# 1.Introduction

## 1.1 Members:

**Đỗ Hoàng Anh – 20200012**

**Hoàng Huy Chiến – 20200084**

**Nguyễn Hoàng Hải – 20200193**

**Nguyễn Phương Quang – 20205191**

**Topic: Diffusion-Limited Aggregation.**

## 1.2 DLA problem:

Diffusion-Limited Aggregation (DLA) is a process, specialized to describe the random scalar growth of individuals. It is a model for non-equilibrium growth, where growth is determined by diffusing particles.

## 1.3 Practical applications:

DLA is used not only to describe the growth of living organisms (such as the growth of viruses, algae, and cells) but also to describe other physical processes (such as dissolution, osmosis, etc.).

## 1.4 The problem posed:

DLA can model 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.

# 2. Theories

## 2.1 Theorem statement:

During the simulation, the first bacteria will be placed in a single cell. The bacterial growth in the environment is likely to depend on the concentration of nutrients in the cells surrounding it.

The concentration of nutrients in the environment is also continuously changing. Each growth, cells with bacteria will be eaten up, the difference in concentration of nutrients will be changed by the diffusion.

The diffusion equation simulates the change of the concentration of nutrients. We assume that the nutrients are only transported by free diffusion in the environment. We will now derive an equation for the concentration c, by considering mass balance in a small volume and invoking Fick’s law that relates a diffusion flux in a linear way with the concentration gradient.

Consider that situation, as in Figure 1, where a concentration gradient exists and that due to this gradient a net flux of solutes exists. A flux is an amount of material that passes through a certain area per time unit. This flux, **J**, is therefore measured in units of number (of molecules)/m2s. Fick’s law states that the flux and the concentration gradient depend on each other in a linear way,

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

where D is the diffusion coefficient (in unit m2/s). Fick’s law is valid to a high degree of accuracy.

Diagram

Description automatically generated Shape, rectangle

Description automatically generated

Figure . Due to a concentration ∇c gradient a flux **J** exists Figure 2. A small volume dxdydz

Next, consider an infinitesimally small volume dV = dxdydz. (Figure 2) and calculate the amount of material that diffuses into and out of the cube. We do this here by treating the three Cartesian directions independently and next adding all contribution to get the total flux. For the amount of material flowing into the cube per unit time in the x-direction we may write

and likewise in the other two directions. Next, the total increase in amount of material per unit time must be equal to the total amount of material diffusing into the cube, i.e.

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

where t denotes time. Dividing Equation (2) by dV and taking the limit dx, dy, dz → 0 we find

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

Finally, by combining Equation (3) with Fick’s law (Equation (1)) we obtain the diffusion equation

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

This change in the concentration of nutrients is much faster than rate of the bacterial growth, diffusion time can be ignored, so we use the time-independent equation to determine the concentration of nutrients. Since it is a time-independent equation, the derivative of concentration with respect to time is assumed to be zero.

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

## 2.2 SOR iterative method:

The Equation (5) is a Laplace equation, we can solve it using Successive Over Relaxation (SOR) iterative method.

The SOR iterative method uses the following formula to calculate :

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

where ω is the correction parameter. The parameter ω determines the strength of the mixing. One can prove that for 0 < ω < 2 the method is convergent.

For ω = 1 we recover the Gauss-Seidel iteration, for 0 < ω < 1 the method is called Successive Under Relaxation. For 1 < ω < 2 we speak of Successive Over Relaxation, or SOR.

Specifically, in the report, we will survey with ω = 1.9.

However, this formula is for calculating row-wise order. For parallel simulation, we use the Red-Black ordering.

The idea of the reordering of the computations is as follows. First, color the computational grid as a checkerboard, with red and black grid points. Next, given the fact that the stencil in the update procedure only extends to the nearest neighbors, it turns out that all red points are independent from each other (they only depend on black points) and vice-versa. So, instead of the row-wise ordering we could do a red-black ordering, were we first update all red points, and next the black points (see Figure 3). We also call this Gauss-Seidel iteration (SOR is extended from Gauss-Seidel), because although the order in which grid points are updated is now different, we also do the computation in place, and use new results as soon as they become available.

Diagram

Description automatically generated

Figure . Row wise ordering (left) versus red-black ordering (right).

This new red-black ordering restores parallelism. We can now first update all red points in parallel, followed by a parallel update of all black point. Pseudo code for this method is shown in Figure 4 behind

Text

Description automatically generated

SOR

SOR

Figure . The pseudo code for parallel SOR iteration with red-black ordering.

## 2.3 Application to DLA problem:

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

1. Solve the Laplace equation to get the distribution of nutrients, assume that the object is a sink (c = 0 on the bacteria cells).

2. Let the bacteria grow (according to the probability).

3. Go back to step 1.

Specifically:

Step 1: Will be done by a parallel SOR iteration.

Step 2: Requires three steps

1. Determine growth candidates;
2. Determine growth probabilities;
3. Grow.

A growth candidate is basically a lattice site that is not part of the object, but whose north, or east, or south, or west neighbor is part of the object.

Figure . The object and possible growth sites.

Possible configuration of the object is shown in Figure 5; the black circles form the current object, while the white circles are the growth candidates.

The probability for growth at each of the growth candidates is calculated by

 (7)

The parameter determines the shape of the object. For η = 1 we get the normal DLA cluster, i.e., a fractal object. For η < 1 the object becomes more compact (with η = 0 resulting in the Eden cluster), and for η > 1 the cluster becomes more open (and finally resembles say a lightning flash).

Modeling the growth is now a simple procedure. For each growth candidate a random number between zero and one is drawn and if the random number is smaller than the growth probability, this specific site is successful and is added to the object. In this way, on average just one single site is added to the object.

# 3.Set-up:

## 3.1 Data structure:

* Viruses are described by the VirusPlace type in C:

typedef struct VirusPlace{

    int x;

    int y;

}Virus;

Where x and y are the row coordinates, column coordinates of the Virus, respectively.

* Constants

#define ENVIRONMENT\_SIZE 256

#define OMEGA 1.5

#define ETA 2

#define VIRUS\_LIM 1500

#define FIRST\_VIRUS 254, 128

#define TOLERANCE 0.0001

* The array c[][] represents the food remaining in the area, initialized to 1, minus the only cell containing the first virus:

*//Matrix of nutrient concentration*

double c[ENVIRONMENT\_SIZE][ENVIRONMENT\_SIZE];

* The chance[] array contains the list of probabilities of the candidates, the number of which is also nCandidate:

*//Array that stores the probability to become a virus of candidates*

double chance[ENVIRONMENT\_SIZE\*ENVIRONMENT\_SIZE];

* The array grow[][] is used to mark the locations of viruses and candidates:

*//Matrix of viruses' and candidates' location*

int grow[ENVIRONMENT\_SIZE][ENVIRONMENT\_SIZE];

## 3.2 Program structure:

**Calculate:**

* void addVirus(int *x*, int *y*, int *index*)

To add a candidate has recently become virus to virus[]

* void init()

Generating the initial environment and add the first virus **to it**

* void sor()

Calculating c[][] using SOR method

* void eat()

To set all nutrient concentration on virus places to 0

* void computeProbability()

Calculating the probability to become a virus of each candidate

* void growth()

To consider if a candidate would become a virus depending on probabilities and add it to the environment

* void solve()

Combination of functions that run each growth

## 3.3 Visualizing:

The result matrix of concentration of nutrients and that of bacteria places is write in the files “output\_nutrients.txt” and “output\_bacteria.txt”

We use MATLAB to visualize two above matrices by the following script:

Text

Description automatically generated

# Test results:

Place the first bacteria at the middle of the bottom of the environment

The image on the left shows the distribution of nutrients, in this image, the cell with the white color is where concentration of nutrients is zero (have just been eaten by bacteria). The image on the right shows the places of the bacteria (red cells).

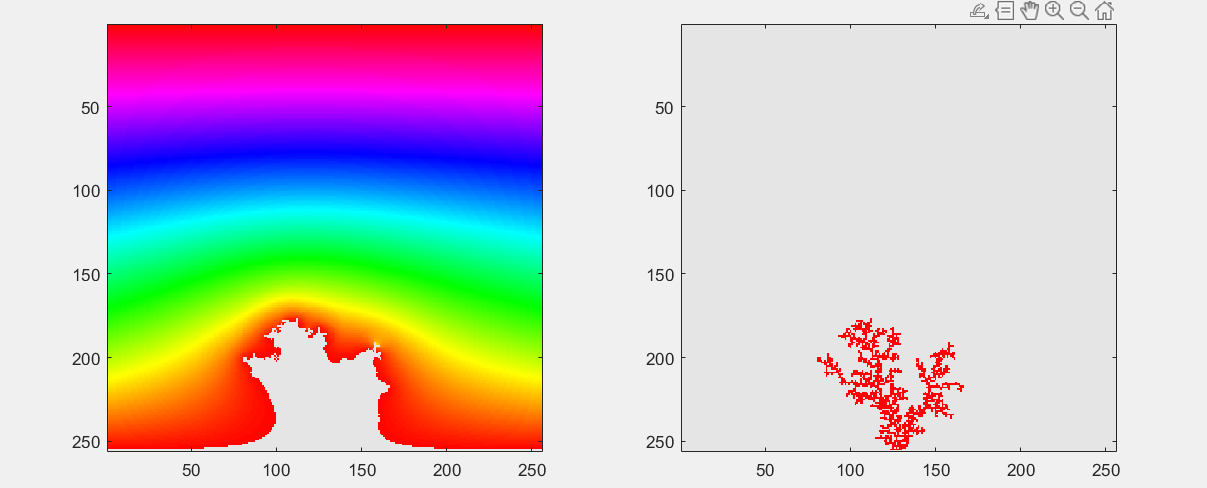
For η = 0,

Chart

Description automatically generated

This is called the Eden growth. Now, the bacteria grow independently from the distribution of nutrients, so a very compact growth form is obtained.

For η = 1,



The bacteria grow depending on the probabilities computed from distribution of nutrients and form a normal DLA cluster or a typical fractal pattern (a tree shape with lots of branches).

For η = 2,

Chart

Description automatically generated

The bacteria are now having more motivation to move to where the concentration of nutrients is high. So, the growth forms a more open tree shape with less branches or we can call a lightning flash shape.

**Bonus: Another way to visualize**

Using the following MATLAB script, we will obtain a clearer image of the final environment

Text

Description automatically generated

Results: With this script, the white cells are where the bacteria are, and gradient colors is the image of the distribution of nutrients.

For:

η = 0;

Chart

Description automatically generated

η = 1;

Chart

Description automatically generated

η = 2;

Chart

Description automatically generated

# Simulating:

By removing the comment symbol from the block of codes about FILE \*f0 or file ‘output\_simulate.txt’ in the source code (put it in comments since it takes time). We could use file ‘‘output\_simulate.txt’ on MATLAB to simulate the growth of bacteria (script is in the file simulate.m).

Text

Description automatically generated

Figure 6. Block of codes about ‘output\_simulate.txt’ is surrounded by /\* \*/

Chart

Description automatically generated

Figure 7. Bacteria growth simulating results