Hanoi University of Science and Technology

**School of Information and Communication Technology**

====================================

Scientific Computing

Project Report

Diffusion-Limited Aggregation (DLA)

GROUP 9

**Students:**

|  |  |
| --- | --- |
| 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 3](#_Toc139613434)

[1.1 Diffusion Limited Aggregation 3](#_Toc139613435)

[1.2 Application of DLA 3](#_Toc139613436)

[1.3 Problem description 3](#_Toc139613437)

[II. Theory 4](#_Toc139613438)

[2.1 Theory 4](#_Toc139613439)

[2.2 SOR Method (Successive Over-Relaxation method) 4](#_Toc139613440)

[2.3 Application to DLA problem 5](#_Toc139613441)

[III. Installation 6](#_Toc139613442)

[3.1 Data modeling 6](#_Toc139613443)

[3.2 Program Design 6](#_Toc139613444)

[3.3 Simulation in MATLAB 7](#_Toc139613445)

[IV. Results 8](#_Toc139613446)

[V. Conclusion 11](#_Toc139613447)

[REFERENCES 12](#_Toc139613448)

[1]

# **Introduction**

## Diffusion Limited Aggregation

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

## 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.

## Problem description

Among many applications of DLA, this report will cover the process of simulating growth of virus-producing individual, in the environment containing food and the virus will not stop eating food and growing.

# **Theory**

## Theory

As the description says, bacteria will grow from a single cell. Bacterial growth in the food environment is a probability of dependency on the food concentration of the cells surrounding it.

Food concentration of the environment is also constantly changing, cells with virus will be eaten up, the difference in food is changed by propagation equation simulates the change of food concentration, because this change is very fast compared to the growth rate of virus so propagation time can seem to be very small. We will use **the time-independent propagation equation** for food concentration determination.

Were : Spreading coefficient.

: Concentration.

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

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

## SOR Method (Successive Over-Relaxation method)

We did a red-black order. For updating black points, since we only need the red point, only the red boundary points need to be exchanged. So red-black ordering means that we first update all red 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.

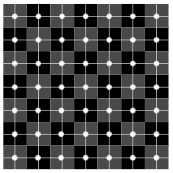


Figure 1. Red-black ordering

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

Text

Description automatically generated with low confidence

Figure 2. Formula of SOR iterative method

where *w* is the correction parameter, (1 < *w* < 2) and in this case, we choose *w* = 1.9

## Application to DLA problem

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:

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

# **Installation**

In this project, the programming is installed on C++ language, GNU compiler via TMD GCC tool on windows environment.

## Data modeling

Virus is modeled by VirusPlace on C++:

Struct VirusPlace {

long x;

long y;

double prob;

};

In which x and y are coordinate of row and column of the cell respectively, prob is the probability that a cell can be virus.

vector< > **virus** is the list of appeared virus, is initialized by one virus. The number of current viruses is controlled by **virus.size()**

vector< > **candidate** contains list of the possible virus candidate in the current step, variable **nCandidate** is the number of candidates.

Array c[][] represents the food concentration, initially declared by 1 except for the first cell contains virus.

Array grow[][] marks the position of virus and candidate. (1 for virus and 2 for candidate).

## Program Design

Input/output data:

* Input: position of the first virus
* We define some constants in the program.

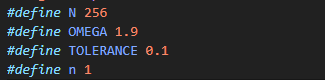


Figure 3 Constants term

* + In practice, n should belong to range [0,2]
* Output: write matrix of the virus to “result.txt”

Procedure:

|  |  |
| --- | --- |
| **AddVirus(u,v)** | marks that the spot (u,v) is a virus and add that spot to the list of viruses |
| **Initialize()** | initialize the value of matrix c[][], then make the first virus by calling addVirus(N-2, N/2). |
| **SOR()** | use SOR method to calculate the value of matrix c[][]. |
| **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. |
| **Main()** | has a while() loop {sor(), eat(), computeProbality(), growth() } |

## Simulation in MATLAB

* Store the output file in “result.txt”

# **Results**

We set:

* Size of matrix : 256x256
* Maximum virus: 2000
* Omega: 1.9
* Tolerance: 0.1

A picture containing chart

Description automatically generated

Figure 4. Testcase1: n = 2

Chart

Description automatically generated

Figure 5. Testcase 2: n = 1

Chart

Description automatically generated

Figure 6. Testcase 3: n = 0

**\*Comments:**

- As the value of parameter *η* decreases, the structure changes and the branches become thicker.

- In our program, we calculate the mean difference and set it as the stopping condition for the SOR method (instead of the maximum difference in the references [1]). By doing so, the SOR algorithm might not perform too many iterations than required to obtain reasonably accurate solution.

- For n = 0, the probability of each cell in the matrix is equal, therefore the virus will expand to all the neighbors around.

- For n = 1, the probability of each cell in the matrix is proportional to the value c[][], thus, the virus will expand towards the top of matrix where the value c[][] is much higher.

- For n = 2, the probability of each cell in the matrix is proportional to the square of value c[][], meaning that the virus moves upwards faster than the previous case.

# **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. |