Υπολογιστική Φυσική Στερεάς Κατάστασης
DLA

Σεϊτανίδου Δήμητρα

12 Μαΐου 2020

Δημιουργία του DLA

Αρχικά θέλουμε να δημιουργήσουμε μία εικόνα ενός DLA, από τυχαίο περίπατο που πραγματοποιούν σωματίδια, ακολουθώντας τα παρακάτω βήματα. Ορίζουμε ένα δισδιάστατο πλέγμα 601×601 και μέσα σε αυτό έναν εξωτερικό νοητό κύκλο με διάμετρο 600 και έναν εσωτερικό με διάμετρο 400. Στο κέντρο του πλέγματος υπάρχει το αρχικό σωματίδιο (particle zero). Από τυχαίο σημείο στην περιφέρεια του εσωτερικού κύκλου ξεκινάνε ένα-ένα σωματίδια και πραγματοποιούν τυχαίο περίπατο μέχρι να βρεθούν δίπλα σε προϋπάρχων σωματίδιο και να κολλήσουν δίπλα του. Ο εξωτερικός κύκλος υπάρχει για να ελέγξουμε μήπως κάποιο σωματίδιο πάει να ξεφύγει μακριά από το DLA. Αν γίνει κάτι τέτοιο τότε καταστρέφουμε αυτό το σωματίδιο και παίρνουμε ένα καινούργιο από την περιφέρεια του εσωτερικού κύκλου. Σταματάμε την διαδικασία ανάπτυξης του DLA όταν φτάσουμε κοντά στον εσωτερικό κύκλο.

Στο δισδιάστατο πλέγμα του κώδικα ο άδειος χώρος ορίζεται από μηδενικά, ενώ τα σωματίδια που αποτελούν το DLA ορίζονται από τη μονάδα. Ο εξωτερικός κύκλος καθώς και τα όρια του πλέγματος ορίζονται από -1 έτσι ώστε εύκολα να καταλαβαίνουμε πότε πρέπει να καταστρέψουμε ένα σωματίδιο και να πάρουμε καινούργιο.

Αυτά τα βήματα ακολουθήσαμε για να φτιάξουμε το DLA. Την τελική μορφή του πλέγματος την αποθηκεύουμε σε ένα αρχείο για να μπορέσουμε μετά να το απεικονίσουμε. Παρακάτω δίνεται ο κώδικας.

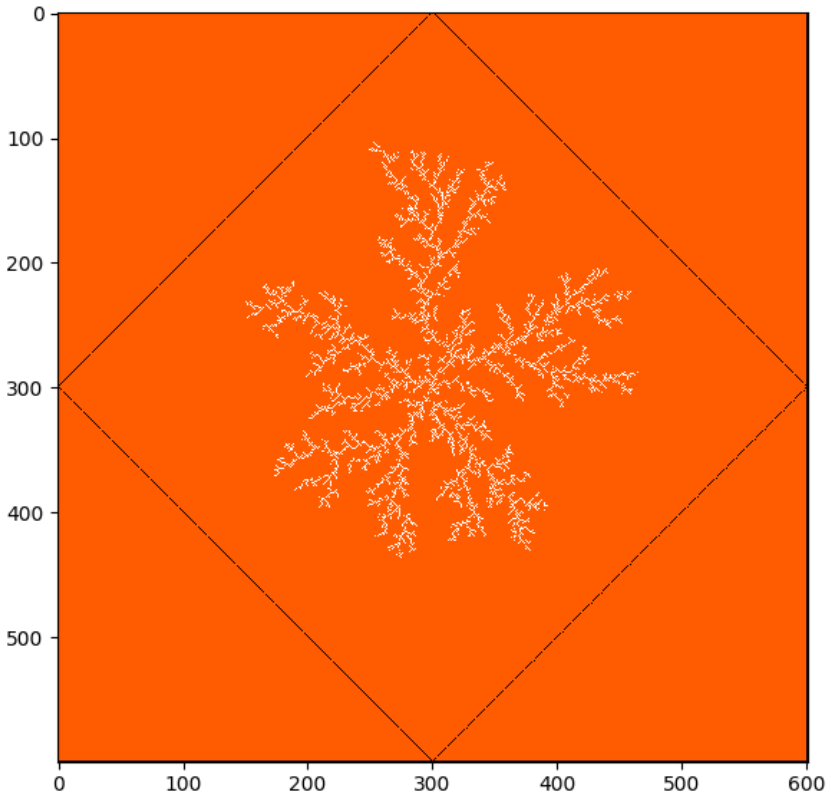
```
1  #define _USE_MATH_DEFINES
2
3  #include <cmath>
4  #include <iostream>
5  #include <fstream>
6  #include <cstdlib>
7  #include <vector>
8  using namespace std;
9
10 int main(int argc, char const *argv[]) {
11   int i,j,j0,j1,iindx,jindx,steps,edge;
12   int n=601;
13   int m=400;
14   double x;
15   bool stop;
16   vector<vector<int> > grid(n,vector<int> (n));
17
```

```
ofstream f("dla.dat");
srand(4372);

grid.assign(n, vector < int >(n, 0));
grid[n/2][n/2]=1;  // particle zero

// create outer circle
j0=n/2;
j1=j0+1;
grid[0][j0] = -1;
j0=j0-1;
for(i=1;i<n/2;i++){
    grid[i][j0] = -1;
    grid[i][j1] = -1;
    j0 = j0-1;
    j1 = j1+1;
  }
j0=0;
j1=n-1;
for(i=n/2;i<n;i++){
  grid[i][j0] = -1;
  grid[i][j1] = -1;
  j0 = j0+1;
  j1 = j1-1;
}

//create stop condition on the edges of the grid
for(i=0;i<n;i=i+n-1){
  for(j=0;j<n;j++){
    grid[i][j] = -1;
  }
}

for(j=0;j<n;j=j+n-1){
  for(i=0;i<n;i++){
    grid[i][j] = -1;
  }
}

loop:do {
  steps = 0;
  // generation of particles from inner circle
  x = ((double) rand() / (RAND_MAX))*2*M_PI;
  iindx = round(n/2 + m/2*sin(x));
  jindx = round(n/2 + m/2*cos(x));
  stop = false; // stop if particle attaches
  do {
    x = ((double) rand() / (RAND_MAX)); // random walk
    if(x<0.25){
```

```
67          jindx = jindx + 1;
68        } else if(x<0.5){
69          iindx = iindx + 1;
70        } else if(x<0.75){
71          jindx = jindx - 1;
72        } else {
73          iindx = iindx - 1;
74        }
75        if(grid[iindx][jindx]==-1) { // if particle exceeds bounds take new
       particle
76          goto loop;
77        }
78        if ((grid[iindx+1][jindx]==1) || (grid[iindx-1][jindx]==1)||(grid[
      iindx][jindx+1]==1)||(grid[iindx][jindx-1]==1)||(grid[iindx+1][jindx
      +1]==1)||(grid[iindx+1][jindx-1]==1)||(grid[iindx-1][jindx+1]==1)||(
      grid[iindx-1][jindx-1]==1)) {
79          grid[iindx][jindx] = 1; // attach if there is a neighboring
      particle
80          stop = true;
81        }
82        steps = steps + 1;
83      } while(stop==false);
84    } while(steps>2); // stop if too close to the inner circle
85
86  //write result on file
87    for(i=0;i<n;i++){
88      for(j=0;j<n;j++){
89        f << grid[i][j] << "\t";
90      }
91      f << endl;
92    }
93
94    return 0;
95  }
```

Τώρα για την απεικόνιση του DLA φτιάχνουμε ένα μικρό πρόγραμμα στην python και με την εντολή plt.imshow() φτιάχνουμε το heatmap του δισδιάστατου πλέγματος.

Σχήμα 1: DLA



Ο κώδικας που χρησιμοποιήθηκε για να βγει το παραπάνω σχήμα είναι ο εξής:

```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_table("dla.dat")

plt.imshow(data, cmap='hot', interpolation='nearest')
plt.show()
```

Μορφοκλασματική διάσταση

Στο δεύτερο κομμάτι της άσκησης καλούμαστε να προσδιορίσουμε την μορφοκλασματική διάσταση του DLA. Γνωρίζουμε ότι ισχύει η σχέση:

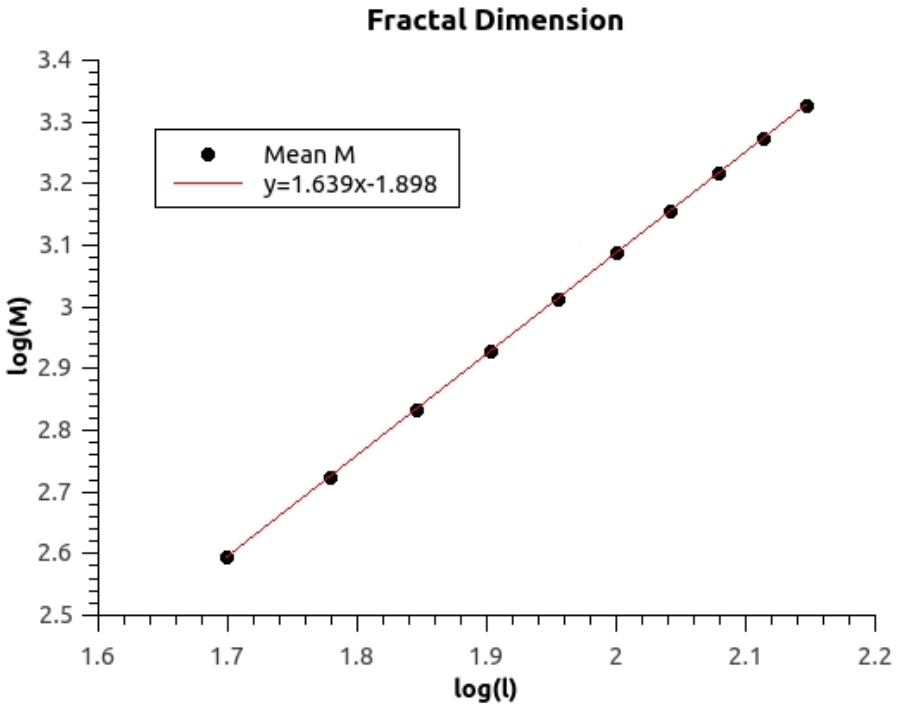
$$M = l^{df} \tag{1}$$

όπου M το πλήθος των σωματιδίων που υπάρχουν σε τετράγωνο διάστασης l και df η μορφοκλασματική διάσταση. Η διάσταση l πρέπει να είναι αρκετά μικρότερη της διάστασης του πλέγματος ($l << L$) για να αποφύγουμε τη καταμέτρηση σε άδειο χώρο.

Ο τρόπος με τον οποίον υπολογίζουμε την μορφοκλασματική διάσταση είναι ο εξής. Διαλέγουμε 10 l, ξεκινώντας από το 50 μέχρι το 140 με βήμα 10, και μετράμε το M. Επειδή η σχέση ανάμεσα στα l και M είναι νόμος δύναμης, φτιάχνουμε το διάγραμμα του $\log l$ με το $\log M$ και υπολογίσουμε την ευθεία ελαχίστων τετραγώνων. Η κλίση της ευθείας είναι το df.

Για να είμαστε σίγουροι ότι τα αποτελέσματα μας είναι σωστά εκτελούμε πολλά runs (100) και σε κάθε run υπολογίζουμε το M, για τα αντίστοιχα l. Στο τέλος για κάθε l βρίσκουμε τον μέσο όρο του M, όποτε καταλήγουμε με 10 M και 10 l. Κάνουμε το διάγραμμα και υπολογίζουμε το df όπως είπαμε παραπάνω.

Σχήμα 2: Μορφοκλασματική διάσταση



Η μορφοκλασματική διάσταση του DLA είναι ίση με $\mathbf{df = 1.639}$.
Ο κώδικας που χρησιμοποιήσαμε για τους παραπάνω υπολογισμούς δίνεται παρακάτω.

```
#define _USE_MATH_DEFINES

#include <cmath>
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <vector>
using namespace std;

int main(int argc, char const *argv[]) {
  int ii,i,j,k,ki,j0,j1,iindx,jindx,steps,edge,count,sum;
  int n=601;
  int m=400;
  int Ni=100;
  int Mk[10][Ni];
  double x;
  bool stop;
  vector<vector<int> > grid(n,vector<int> (n));

  ofstream data("df.txt");
  srand(4372);

  for(ii=0;ii<Ni;ii++){
    grid.assign(n, vector < int >(n, 0));
    grid[n/2][n/2]=1;  // particle zero

    // create outer circle
    j0=n/2;
    j1=j0+1;
    grid[0][j0] = -1;
    j0=j0-1;
    for(i=1;i<n/2;i++){
        grid[i][j0] = -1;
        grid[i][j1] = -1;
        j0 = j0-1;
        j1 = j1+1;
      }
    j0=0;
    j1=n-1;
    for(i=n/2;i<n;i++){
      grid[i][j0] = -1;
      grid[i][j1] = -1;
      j0 = j0+1;
      j1 = j1-1;
    }

    //create stop condition on the edges of the grid
    for(i=0;i<n;i=i+n-1){
      for(j=0;j<n;j++){
```

```
      grid[i][j] = -1;
    }
  }

  for(j=0;j<n;j=j+n-1){
    for(i=0;i<n;i++){
      grid[i][j] = -1;
    }
  }

  loop:do {
    steps = 0;
    // generation of particles from inner circle
    x = ((double) rand() / (RAND_MAX))*2*M_PI;
    iindx = round(n/2 + m/2*sin(x));
    jindx = round(n/2 + m/2*cos(x));
    stop = false; // stop if particle attaches
    do {
      x = ((double) rand() / (RAND_MAX)); // random walk
      if(x<0.25){
        jindx = jindx + 1;
      } else if(x<0.5){
        iindx = iindx + 1;
      } else if(x<0.75){
        jindx = jindx - 1;
      } else {
        iindx = iindx - 1;
      }
      if(grid[iindx][jindx]==-1) { // if particle exceeds bounds take
new particle
        goto loop;
      }
      if ((grid[iindx+1][jindx]==1) || (grid[iindx-1][jindx]==1)||(grid
[iindx][jindx+1]==1)||(grid[iindx][jindx-1]==1)||(grid[iindx+1][jindx
+1]==1)||(grid[iindx+1][jindx-1]==1)||(grid[iindx-1][jindx+1]==1)||(
grid[iindx-1][jindx-1]==1)) {
        grid[iindx][jindx] = 1; // attach if there is a neighboring
particle
        stop = true;
      }
      steps = steps + 1;
    } while(stop==false);
  } while(steps>2); // stop if too close to the inner circle

  ki=0;
  for(k=50;k<150;k=k+10){ // select area size k
    count=0;
    for(i=n/2-k/2;i<n/2+k/2+1;i++){ // particle zero is at center
      for(j=n/2-k/2;j<n/2+k/2+1;j++){
```

```
          if(grid[i][j]==1){
            count = count +1; // count particle in area
          }
        }
      }
      Mk[ki][ii] = count;
      ki=ki+1;
    }
  }

  k = 50;
  for(i=0;i<10;i++){
    sum = 0;
    for(j=0;j<Ni;j++){
      sum = sum + Mk[i][j]; // compute mean for each area size
    }
    data << k << "\t" << sum/Ni << endl;
    k = k +10;
  }

  return 0;
}
```