フラクタルクラスターの新しいモデル

田畑　志保
羽田野研究室

平成１３年2月２８日

概 要

本研究で扱うフラクタルクラスターは２種類ある。その一つ、DLA クラスターはランダムウォークによってある１点 (種) に向かって粒子が集まる凝集現象である。一方、PLSA モデルはランダムにばらかまれる粒子間の距離 r によって粒子の付着する確率が決まる。PLSA モデルは 1999 年に発表されたばかりの新しいモデルである。このモデルによってフラクタルクラスターが作られることを粗視化によって確認した。

さらに、本研究では DLA クラスターと PLSA モデルの要素を組み合わせ、新しいモデルを提案し、そのフラクタル次元を測定した。

目 次

1　はじめに

自然界にはフラクタルな性質を持つものがたくさんある。星の分布や川の蛇行や分岐の様子などもフラクタルな性質を持つ。そのような自然現象を理論的に取り扱うためには、まずどのようにすればフラクタルな図形を作ることができるか考えなければならない。そこで、本研究ではフラクタルなクラスターを作るモデルをテーマとした。一つはべき乗則に従うモデル、もう一つはランダムウォークをするモデルを用いた。特に、ランダムウォークをさせてフラクタルクラスターが出来ることは興味深い。さらに、この２つのフラクタルクラスターを組み合わせ、新しくフラクタルクラスターのモデルを作った。

2　フラクタル

ここではフラクタルとは何か、どういう性質を持っているかについて触れる。また、フラクタルを定量的に表す量としてフラクタル次元がある。その測定の仕方を述べる。

2.1　フラクタル図形

フラクタル図形は、特徴的な長さ (例えば、球を扱うならばその半径、人間を扱うならばその人の身長というように、そのものに附随する代表的な長さを指す。) を持たないことが特徴である。ゆえに、スケールを大きくしても小さくしても同じような形が見える性質 (自己相似性) を持つ。フラクタル図形は以下で述べるように整数でない次元を持った図形と捉えることができる [1, 2]。

2.2　粗視化とフラクタル次元

今後、「フラクタル次元を測る」ということが何度か出てくるので少し触れておく。フラクタル図形の次元を測る方法はいくつかあるが、本研究では粗視化を用いてフラクタル次元を測ることにする。例えば、複雑な海岸線の距離を測りたいならば、曲線の一端を始点とし、その点を中心にして半径 r の円を描く。その円と曲線が最初に交わった点と始点とを直線で結ぶ。そして、その交点を新たに始点とみなし、以下同じ作業を繰り返す。このようにして長さ r の折れ線によって海岸線の長さを近似するときに必要な線分の総数を $N(r)$ とする。基準となる長さ r を変えれば、当然 $N(r)$ は変化する。もしも、海岸線がまっすぐならば、

$$N(r) \propto 1/r = r^{-1}$$

の関係を満たすはずである。しかし、ギザギザの海岸線では、

$$N(r) \propto r^{-D} \quad D > 1$$

となることがある。この D が海岸線のフラクタル次元である。

本研究では二次元平面上の粒子の分布の次元を測るため、上記の考え方を踏まえ、粒子が存在する範囲を 1 辺が r の正方形のセルに分割し、平面上において少なくとも 1 つの粒子を含む正方形の数を数え、それを $N(r)$ とした。セルの 1 辺を変えて同じようにカウントする。$N(r)$ と r の間に

$$N(r) \propto r^{-D}$$

が成り立つならば D 次元的であると捉えることができる。そして、両辺を対数にすると

$$\log N(r) \propto -D \cdot \log r + C \tag{1}$$

C:定数

となり、両対数グラフに描くと直線の傾きがフラクタル次元となる。

3 PLSA モデル

この節では PLSA モデルについてと、モデルによって作られるクラスターについて述べる。

PLSA モデル [3] は 1999 年に発表されたばかりのモデルである。PLSA とは、「Power-Low Sequential Adsorption」の頭文字を取ったもので、べき乗法則に従って次々に吸着していく、という意味を持つ。都市の人口分布やバクテリアコロニーの成長のモデルに適している。

このモデルによって作られるクラスターをコンピューターで実際に再現し、フラクタルクラスターであるかどうか確認する。また、フラクタル次元を粗視化によって測定した。

クラスターのシミュレーション方法を述べる。まず、座標の中心に種を一粒おく（原点を中心に決める）。次に粒子の座標を乱数で適当に与える。その後、粒子の付着する確率を決める。粒子は

$$\text{確率 } P(r) = \begin{cases} 1 & r \leq d \text{ のとき} \\ (d/r)^{\alpha} & r > d \text{ のとき} \end{cases} \tag{2}$$

d : 粒子の直径

r : 粒子間の最短距離

という確率で付着する。r は問題とする粒子とそれに最も近い粒子との距離で、d は粒子の直径である。なお、$\alpha \geq 0$ とする。

例えば、3 個目の粒子を付けたい時は、種と 3 個目の粒子、すでに付着している 1 個目と 3 個目の粒子、2 個目と 3 個目の粒子の距離をそれぞれ測る。そして、一番短いものを式 (2) の r に代入して確率 $P(r)$ を計算する。ここでもう一度、乱数 R を発生させ、確率 $P(r) \geq R$ ならば問題の粒子は付着する。一方、$P(r) < R$ なら付着せずに粒子をなかったことにして、座標選びからやり直す、というようにシミュレーションする。

こうして図 1 のようなクラスターを得た。このクラスターのフラクタル次元を、第 2.2 節で述べた粗視化の方法によって測定した (図 2)。その結果、式 (1) の直線が確認され、クラスターがフラクタルであることが確認された。フラクタル次元は直線の傾きから $D \simeq 1.5$ と求められる。

ここでパラメータとして、粒子数 M の代わりに粒子密度

$$\eta_0 = \pi(d/2)^2 \cdot M/L \tag{3}$$

M : 粒子数

L : 座標面積

を用いる。$0 \leq x \leq 1, 0 \leq y \leq 1$ であるので、$L = 1.0$ である。図 1 の計算では $\alpha = 2.5, M = 10^5, d = 0.002$ であるから $\eta_0 = \pi/10 \simeq 0.314159$ である。一方、$\eta_0 = 0.01, \alpha = 2.5$ の時、フラクタル次元が 1.5 次元になることが参考文献 [3] よりわかっていた。η_0 の値は 30 倍ほど異なるが、次元は一致した。これはモデルの普遍性を示していると考えられる。

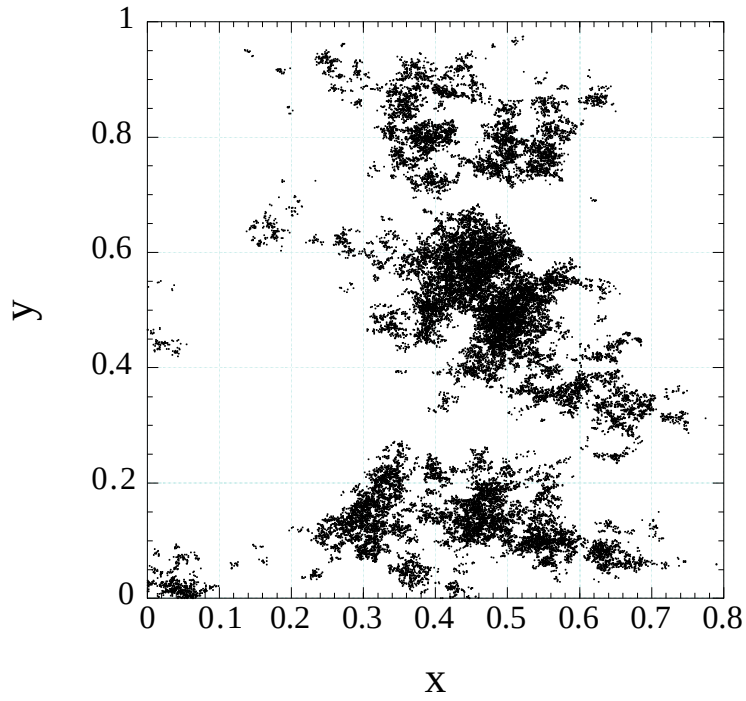

図 1: PLSA モデルにより作られたクラスター。(粒子数$= 10^5$,$\alpha = 2.5$,$d = 0.002$)

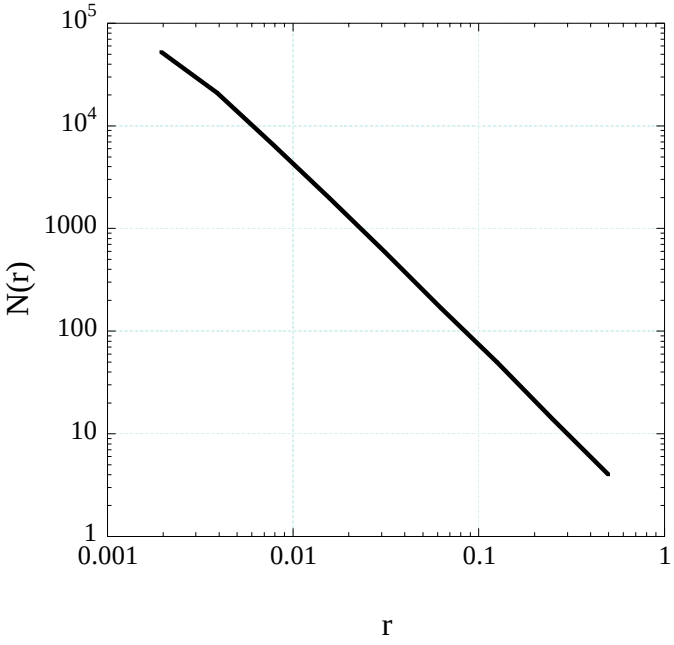

図 2: 粗視化によるフラクタル次元の解析。この場合、直線の傾きが 1.5 なので、1.5 次元のフラクタルクラスターだと考えられる。

4　DLA クラスター

この節では、DLA クラスターの特徴やこれによって作られるクラスターを紹介する。また、このクラスターのコンピューター上でのシミュレーション方法も述べる。

4.1　DLA クラスターと金属葉

DLA クラスターは「Diffsion-Limited Aggregation」の略で、拡散に支配された凝集という意味を持つ。DLA クラスターの代表的な例は「金属葉」である。金属葉とは、有機液体と金属塩水溶液との２液界面上で電析を行うとできる。具体的な作成方法の例として、ガラスの容器に硫酸亜鉛水溶液 (濃度は２モル程度) を入れる。その上に酢酸ブチルを注ぎ、界面を作る。そして、先を平らにしたシャープペンシルの芯の先端を界面の位置にセットする。容器の内側に亜鉛板をリング状に置き、陽極とする。陰極はシャープペンシルの芯である。両者の間に５Ｖ程度の電圧をかけると金属葉が出来上がる (図３)。遠方から浮遊してきた金属原子が陰極に付着して成長してクラスターを作るのである。こうして出来上がった金属葉は約 1.7 次元ほどのフラクタルクラスターを形成する [3, 4]。

4.2　コンピューターシミュレーションによる DLA クラスター

以下のような方法で DLA クラスターをシミュレーションすることができる。正方格子を考える。

1 座標の中心に種を置く。

2 乱数で新たな粒子の座標を選ぶ。

3 選ばれた座標に粒子を置くが、この粒子の上下左右の少なくとも１つに既に付着している他の粒子があれば、そこに付着することが決定する。しかし、そうでなければ新たに乱数 $(0 \leq R \leq 1)$ を発生させ、ランダムウォークさせる。本研究では４方向のランダムウォークのため、$0 \leq R < 0.25$ ならば上のセルに移り、$0.25 \leq R < 0.5$ ならば右のセルに、$0.5 \leq R < 0.75$ ならば下のセルに、$0.75 \leq R \leq 1$ ならば左のセルに移る、というルールを定めた。これがランダムウォークである。

4 ランダムウォークをしてまた上下左右に他の粒子がいなかったらランダムウォークを再度、行う。上下左右のいづれかに粒子がいたら、付着する。

5 2〜４の作業を繰り返す。

粒子数 10^3 としてシミュレーションを行った。実際の金属葉 (図３) とシミュレーションの金属葉 (図４) を比較してみると、特徴がよく似ていることがわかる。

図 3: 実際に化学的な手法で作成した金属葉。約 1.7 次元。

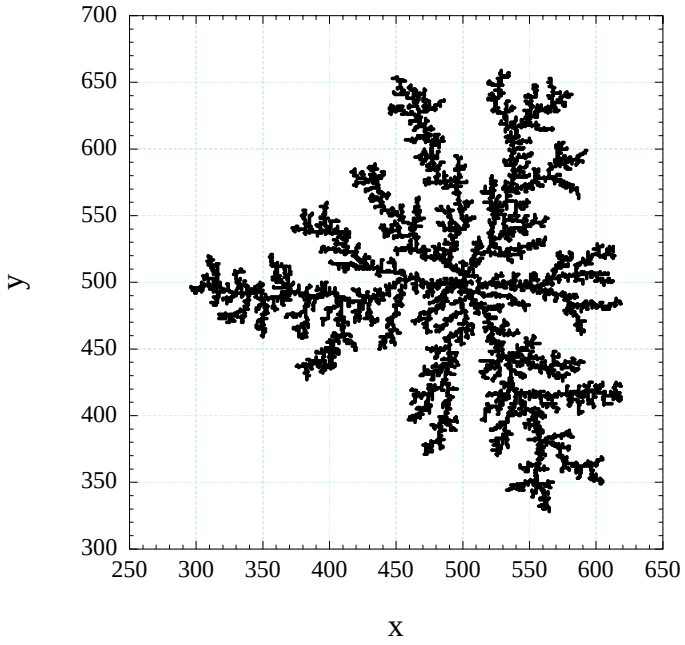

図 4: コンピューターでシミュレーションした結果。粒子数は 10^4。

5　新しいモデル

この章では3章のPLSAモデル、4章のDLAクラスターを組み合わせ、新しいモデルを提案する。

5.1　新しいモデルの概要

PLSAモデルは粒子が近くにいなくても付着するが、ランダムウォークをしないので、モデルに粒子の運動が取り入れられない。それに対し、DLAクラスターはランダムウォークをするので動きはあるが、粒子が近くにいないと付着しない。そこで、この2種類のフラクタルクラスターの特性を組み合わせ、動きがあり、かつ、粒子が近くにいなくても付着するモデルを考えた。人口分布を例に取ると、ある地点に住まいを構えるが引っ越しを考えた時にすぐに遠くの地に移住するのではなく、割と近い土地を選んで引っ越すといったようなイメージである。

5.2　新しいモデルのシミュレーション方法

格子上モデルでモデルを定義する。PLSAモデルと同様に、座標の中心に種を置き、次からの粒子の座標を乱数で選ぶ。粒子は

$$\text{確率 } P(r) = \begin{cases} 1 & r \leq d \text{ のとき} \\ (d/r)^{\alpha} & r > d \text{ のとき} \end{cases} \tag{4}$$

d　: 粒子の直径

r　: 粒子間の最短距離

という確率で付着する。ここで既に付着している粒子との距離を測り、一番短い距離を最短距離rとして式(4)に代入する。そして、新たに発生させた乱数Rと,式(4)の$P(r)$の大小比較をする。$P(r) \geq R$ならば粒子はその場に付着する。ここまではPLSAモデルと全く同じである。$P(r) < R$だった場合、付着せずにランダムウォークをする。ランダムウォークの仕方はDLAクラスターと同様である。ランダムウォークをした粒子とその周辺の粒子との距離をまた新たに測り、式(4)に代入して計算をする。この作業を繰り返す。

5.3　新しいモデルと他のモデルの関係

式(4)においてαが小さい値をとるとき、$P(r)$は長い尾を引くので(図5)、粒子は最初に乱数で与えられた場所にそのまま付着する可能性が高い。従って、そのモデルはPLSAモデルに近いふるまいをする。それに対して、αが大きい値を

とるとき、$P(r)$ は $r > d$ では、付着する確率が急激にゼロに近くなるので (図 6)、他の粒子に接するまでランダムウォークする可能性が高い。従って、このモデルは DLA クラスターに近いふるまいをする。

このように、ここで提案した新しいモデルは、PLSA モデルと DLA クラスターを統一したモデルであるということができる。実際に第 6 章の結果でもそれを確認することができる。

6 結果

$\alpha = 3.0, 3.5, 4.0, 5.0, 6.0$ の 5 つの値で新しいモデルによるクラスターをシミュレーションした。α が小さい時は、ランダムウォークをあまりしないので PLSA モデルに近いクラスターができる。α が大きい時は、ランダムウォークをしやすいので DLA クラスターに近いクラスターを作る。

図 7 から図 11 は α を 3.0 から 6.0 まで値を変えてシミュレーションをした結果のクラスターである。実際に α が 3.0 の場合、粒子がばらついて PLSA モデルに近いクラスターができ、α が 6.0 の場合、DLA クラスターに近いクラスターを作ることに成功した。ここでは η_0 は考えなかったが、実際には PLSA モデルの要素が入っているので関連しているはずである。粒子数 $M = 10^3$ ということもあって、η_0 は小さい値をとると考えられる。

粗視化によるフラクタル次元の解析については、先に述べたように

$$N(r) \propto r^{-D} \tag{5}$$

という関係が成り立つならば、D をフラクタル次元と捉える。図 5 から図 11 のクラスターに対して、粗視化によって $N(r)$ を計算した結果が図 12 である。図 12 の直線の傾きからフラクタル次元を求めることになるが $\alpha = 3.0$ の時は直線とは呼び難い。これは、η_0 が大きいとフラクタル次元が大きくなることが報告されているので、粒子数が少なかったことによるものだったと考えている。その他の場合は、かなり直線に近い傾きが得られ、フラクタルクラスターであることが確認された。

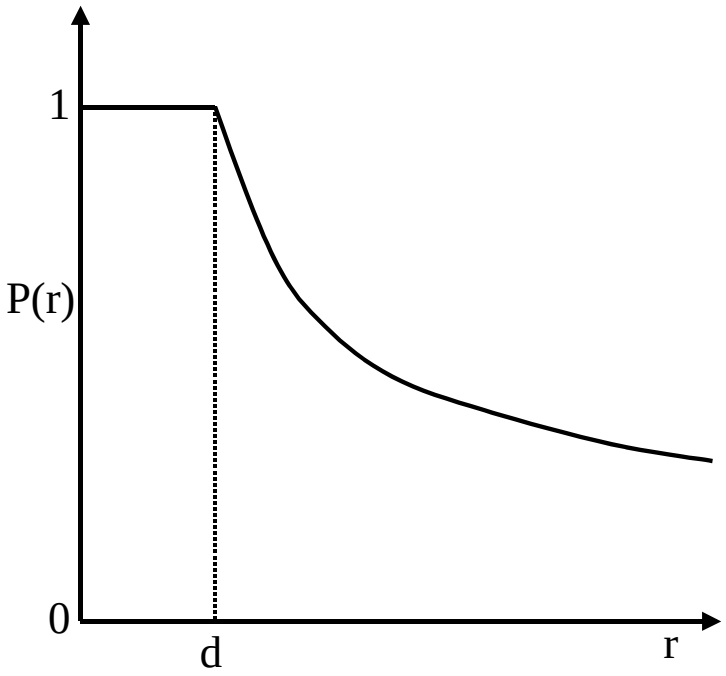

図 5: α が小さい値をとる時は、粒子が最初に乱数で与えられた場所に定着する確率が高い。

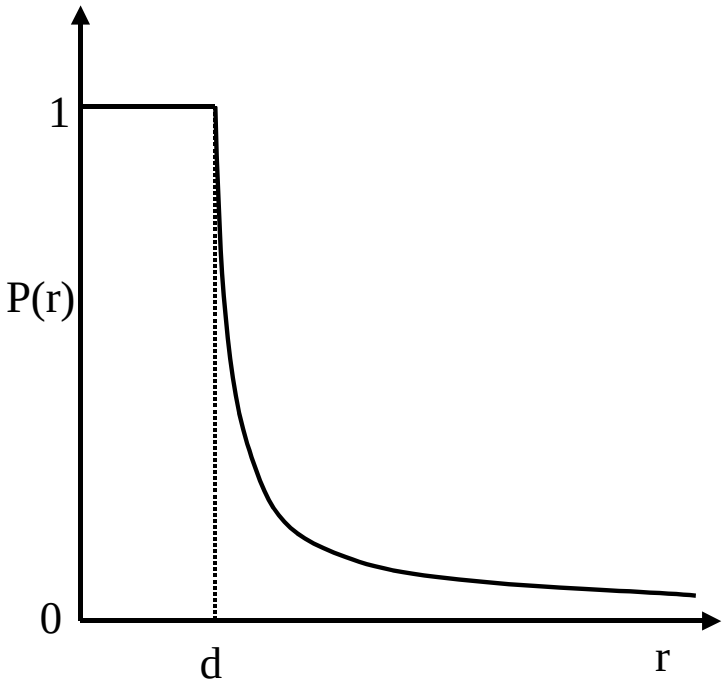

図 6: α が大きくなってくると、粒子が最初に乱数で与えられた場所に定着する確率は低くなり、ランダムウォークをしやすくなる。

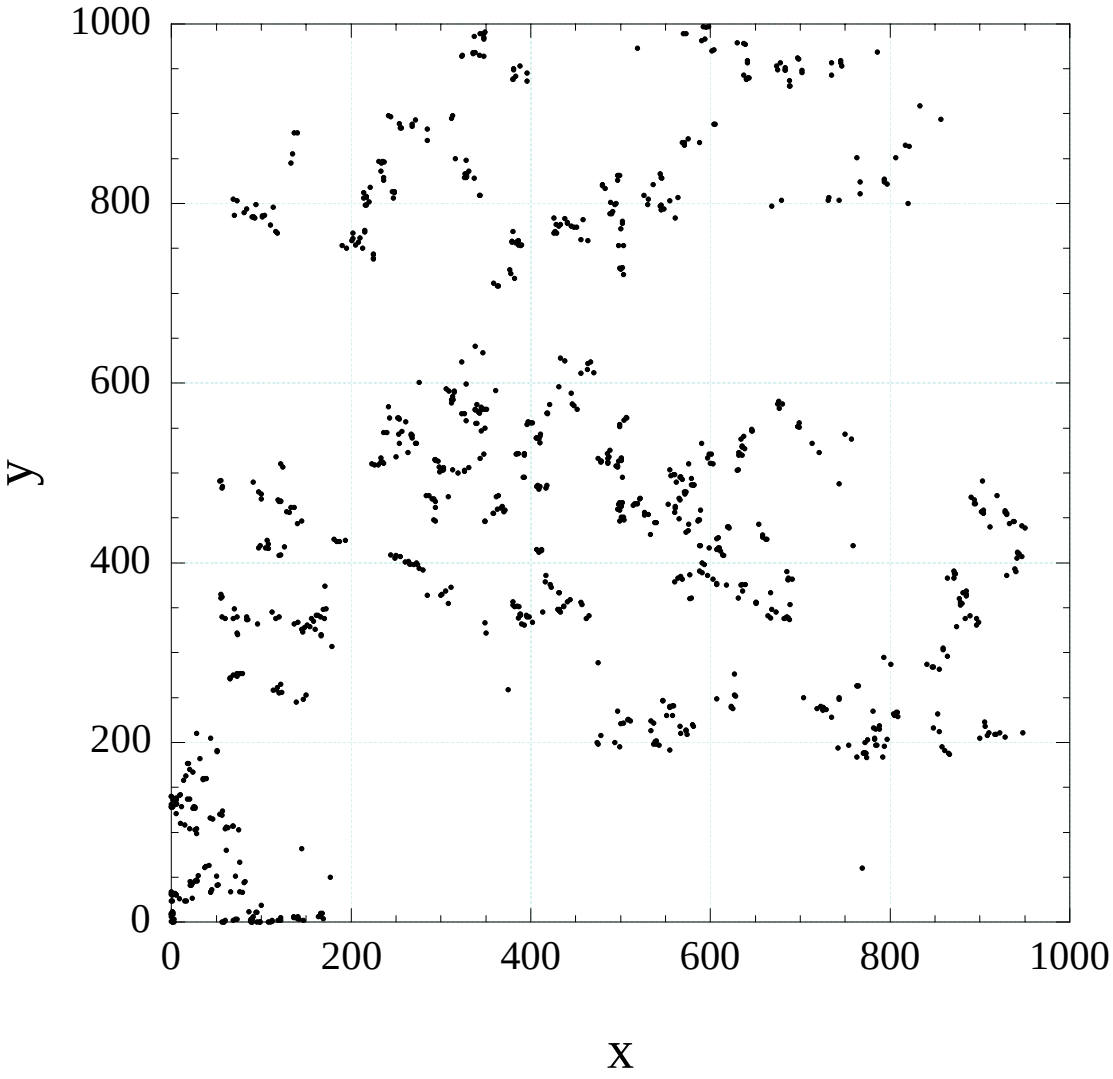

図 7: 式 (4) で $\alpha = 3.0$ とした場合。粒子数 $= 10^3$,$d = 1.0$。

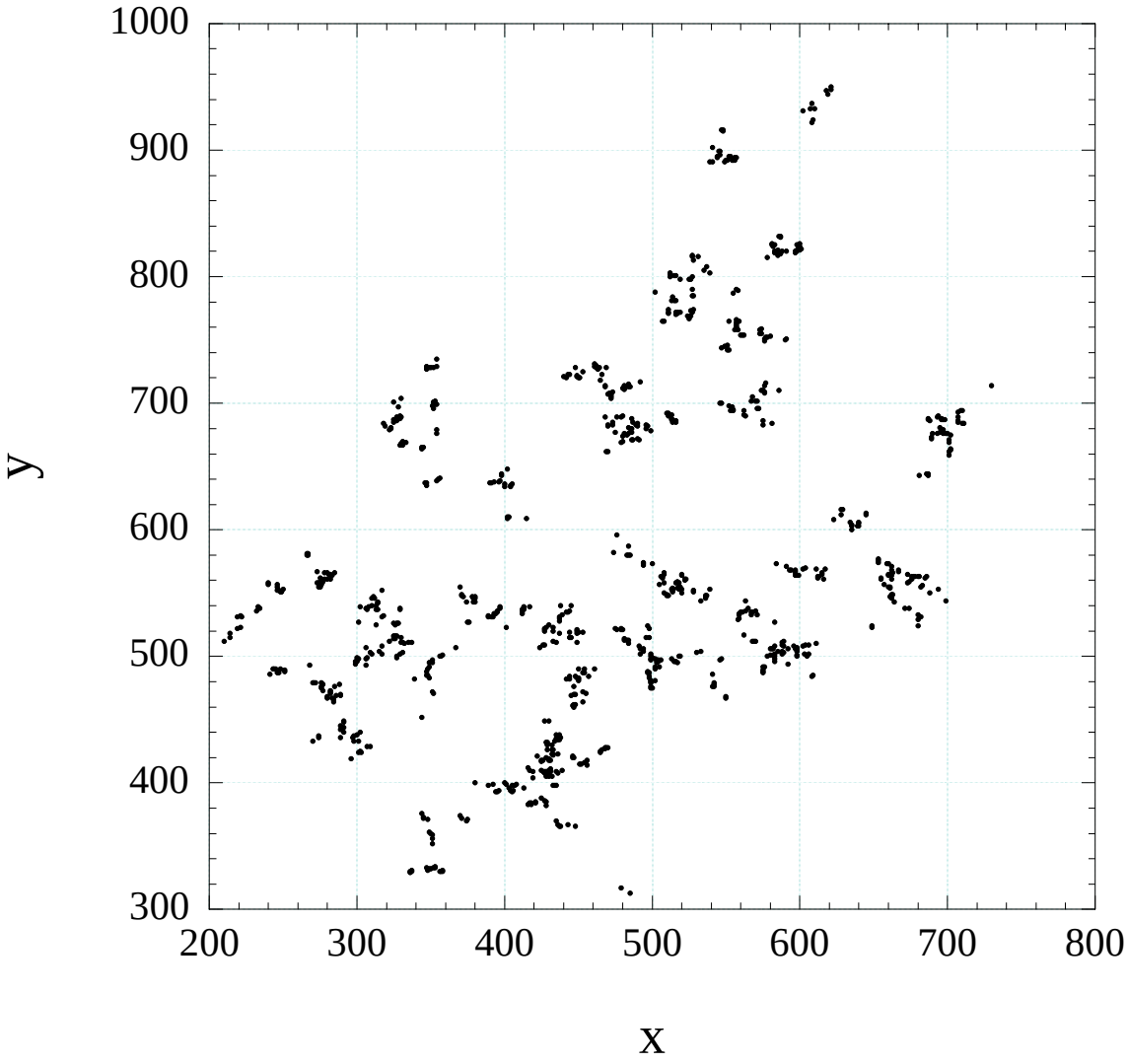

図 8: 式 (4) で $\alpha = 3.5$ とした場合。粒子数 $= 10^3$,$d = 1.0$。

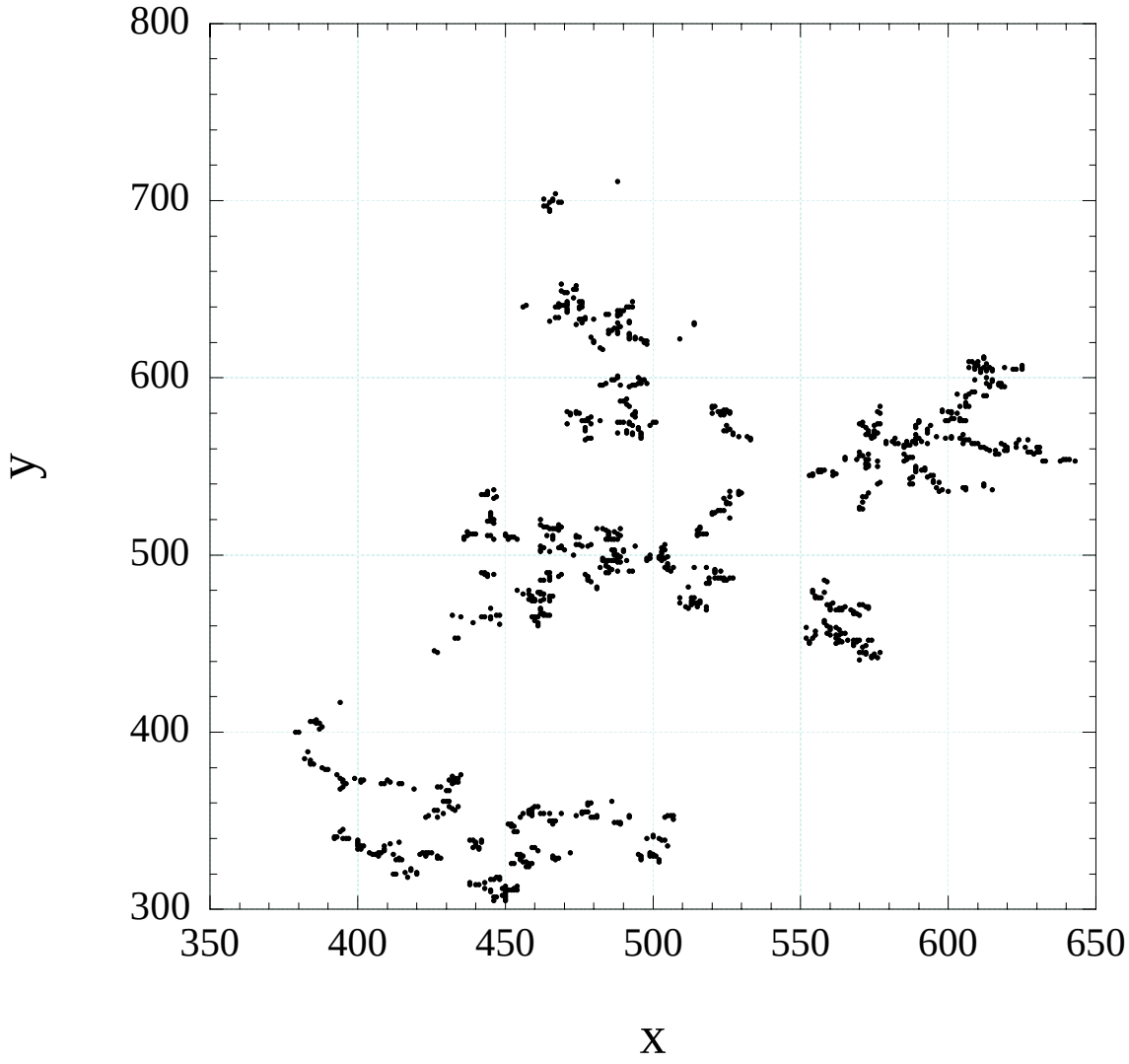

図 9: 式 (4) で $\alpha = 4.0$ とした場合。粒子数 $= 10^3$,$d = 1.0$。

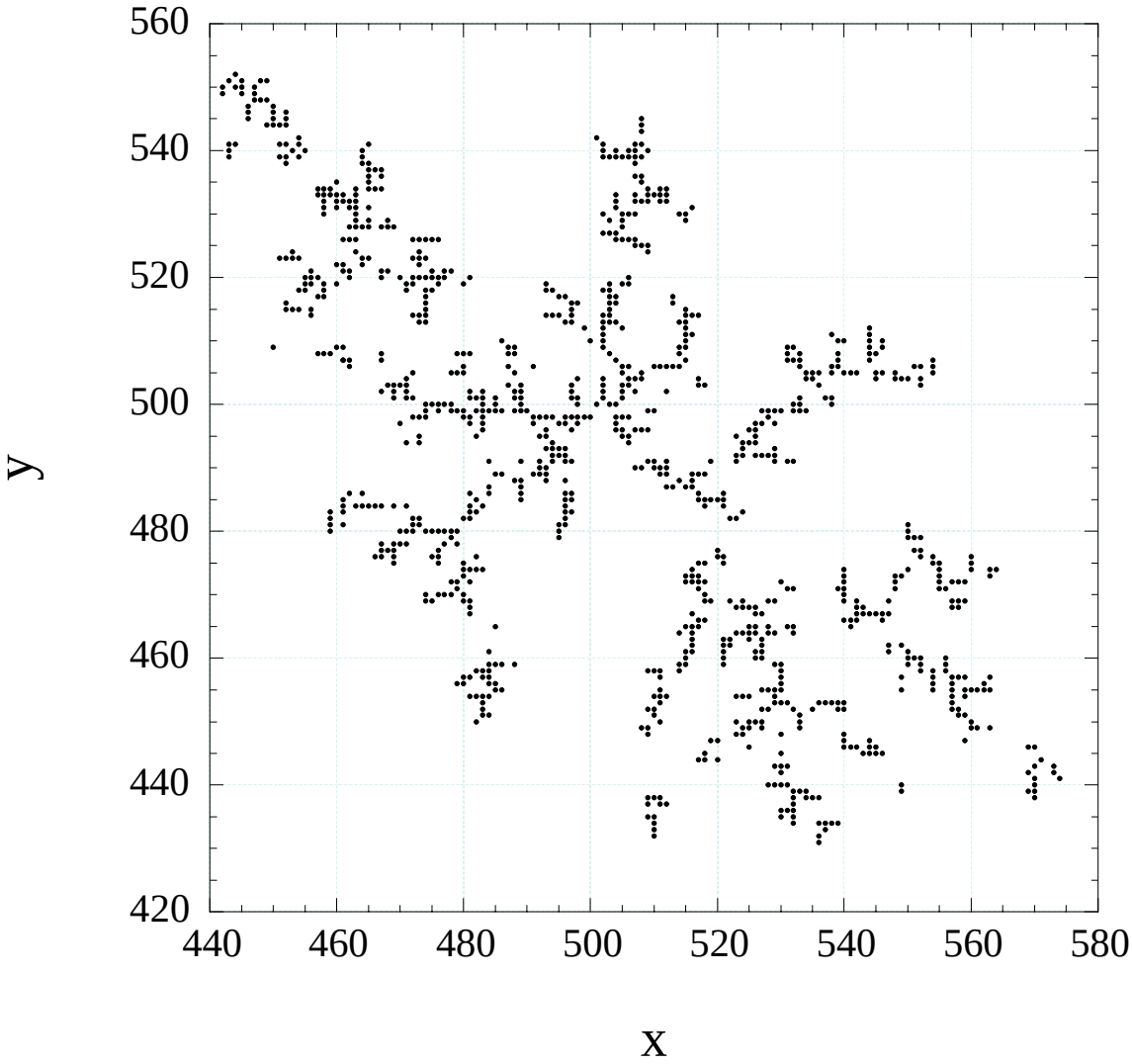

図 10: 式 (4) で $\alpha = 5.0$ とした場合。粒子数 $= 10^3$,$d = 1.0$。

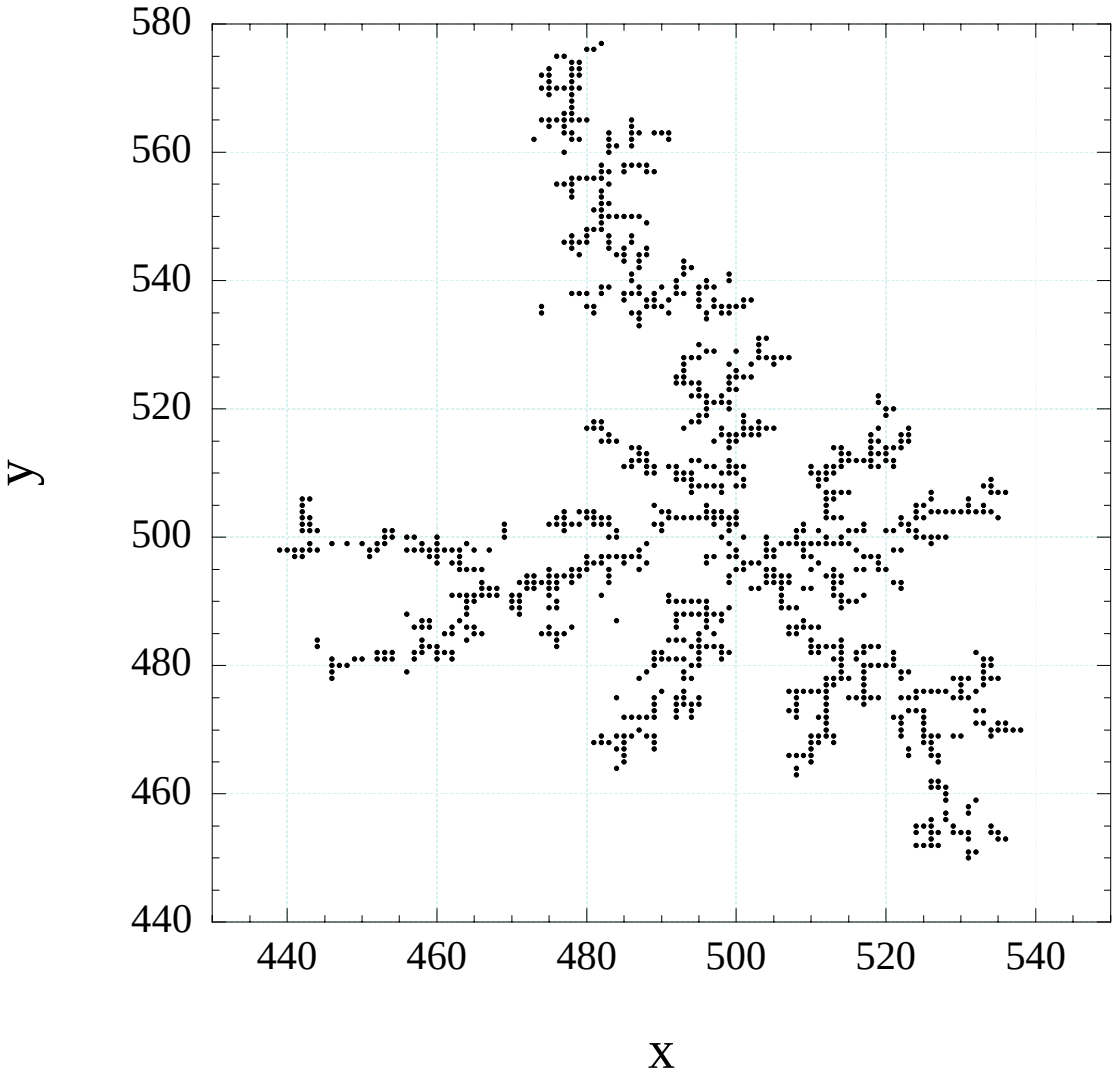

図 11: 式 (4) で $\alpha = 6.0$ とした場合。粒子数 $= 10^3$,$d = 1.0$。α が大きくなるほど金属葉に近いクラスターになることがわかる。

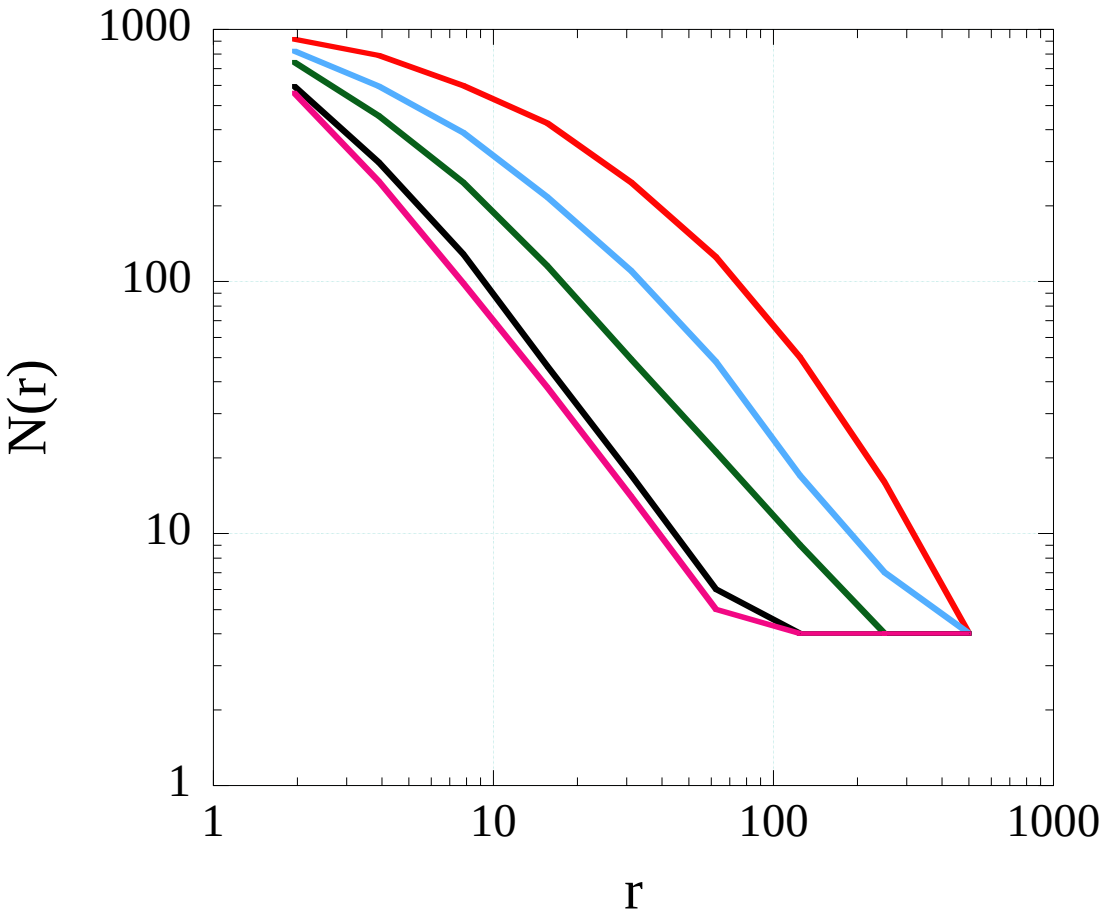

図 12: 粗視化によるフラクタル次元の解析。傾きの小さい直線 (一番上の線) から α=3.0,3.5,4.0,5.0,6.0。α の値が大きくなるほど、直線の傾きが大きい。すなわち、フラクタル次元が高いことになる。

7　まとめと問題点

本研究では、フラクタルなクラスターを作るモデルをテーマとし、べき乗法則に従うモデルとランダムウォークによるモデルから新しいモデルを提案した。その結果、新しいモデルによってフラクタルクラスターを作ることに成功した。第5.3節でも触れているが、$\alpha = 3.0$ の場合、粒子数が少ないので

$$\eta_0 = \pi(d/2)^2 \cdot M/L \tag{6}$$

M　: 粒子数
L　: 座標面積

が小さくなる上に、α の値が小さいことが原因でフラクタル次元を測るのに十分な直線は得られなかった。この点を改善するには、粒子数を増やすことが最善である。さらに、α と η_0 の関係を調べ、その関係から適切な粒子数と α を与え、フラクタル次元を測ることもできるであろう。

8　謝辞

「フラクタルを勉強したい！」という私の気持ちを羽田野直道先生に汲んでいただき、2000年4月から卒研生として、羽田野研究室に『弟子入り』させていただきました。『10時30分に登校』という規則を守ることがあまりできなかった私ですが、先生はいつでも優しく勉強をお教え下さいました。一年間、ありがとうございました。また、先生同様に指導して下さった院生の方々、アドバイスをしてくれた友人にも感謝したいと思います。

参考文献

[1] 高安　秀樹, フラクタル, 朝倉書店 (1986)

[2] 小川　泰, フラクタルとは何か, 岩波書店 (1989)

[3] Ofer Biham, Ofer Malcai, Daniel A. Lidar, David Avnir, Pattern Formation and a Clustering Transition in Power -Law Sequential Adsorption, Phys. Rev. E59 (1999) R4713-R4716

[4] 高安　秀樹, フラクタル科学, 朝倉書店 (1987)

[5] 松下　貢, パリティ『パターン形成の物理』, 丸善 (1997)

A PLSA モデルのプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

main()
{
  int i,j,k,l,hist[1024][1024],histnew[512][512],Ndiv,Nbox;
  static float x[100000],y[100000],a=0.5,b=0.5,r,r0,
                    rcenter,p,dr,d=0.002,alpha=2.5;
  FILE *datafile1;
  FILE *datafile2;

  datafile1 = fopen("plsa3.dat","w");
  datafile2 = fopen("hist1.dat","w");

  for(k=0; k<=1023; k++)
    {
      for(l=0; l<=1023; l++)
        {
          hist[k][l]=0;
}
    }


  for(i=0; i<=100000; i++)
    {
      /***** choose the coordinates at random *****/
        x[i]=(float)rand()/RAND_MAX;
        y[i]=(float)rand()/RAND_MAX;

/***** mesure the distance between particles *****/
        r0=rcenter=sqrt( (a-x[i])*(a-x[i])+(b-y[i])*(b-y[i]) );

/* printf("%d-0 %f\n",i,r0); */

for(j=0; j<i; j++)
           {
```

```
   r=sqrt( (x[i]-x[j])*(x[i]-x[j])+(y[i]-y[j])*(y[i]-y[j]) );

    /*  printf("%d-%d %f\n",i,j,r); */

           if(r<r0)
    {
              r0=r;
    }

         }


   p=(float)rand()/RAND_MAX;
            if( p>pow(d/r0,alpha) )
     {
i--;
     }

     /* printf("%d min %f\n",i,r0); */

  }
/***** measure the fractal dimension *****/
Ndiv=1024;
dr=1.0/Ndiv;

for(i=0;i<=100000;i++)
  {
    fprintf(datafile1,"%f %f \n",x[i],y[i]);
    /* printf("%f %f\n",x[i],y[i]); */

    hist[(int)(x[i]/dr)][(int)(y[i]/dr)]++;

  }

while(Ndiv>1) {

  Nbox=0;
  for(k=0; k<Ndiv; k++)
    {
```

```
for(l=0; l<Ndiv; l++)
  {
    if(hist[k][l]>0)
      {
Nbox++;
      }
  }
      }

    fprintf(datafile2,"%f %d\n",1.0/Ndiv,Nbox);
    Ndiv=Ndiv/2;

    for(k=0; k<Ndiv; k++)
      {
        for(l=0; l<Ndiv; l++)
          {
             histnew[k][l]=hist[k*2][l*2]+hist[k*2+1][l*2]
                          +hist[k*2][l*2+1]+hist[k*2+1][l*2+1];
          }
      }

    for(k=0; k<Ndiv; k++)
      {
        for(l=0; l<Ndiv; l++)
          {
             hist[k][l]=histnew[k][l];
          }
      }

  }

  fclose(datafile1);
  fclose(datafile2);
}
```

B　DLA クラスターのプログラム

```
#include <stdio.h>
#include <stdlib.h>

#define  _SIZE  1001

main()
{
  int j,x,y,x1,y1,x2,y2,x3,y3,x4,y4,particle[_SIZE][_SIZE];
  float r;
  FILE *datafile;

  for(x=0; x<=_SIZE; x++)
    {
      for(y=0; y<=_SIZE; y++)
        {
          particle[x][y]=0;
}
    }
  /***** set the origin *****/
 particle[_SIZE][_SIZE/2]=1;
 datafile = fopen("random2.dat","w");

  for(j=0; j<10000; j++)
    {
      do{
/***** choose the coordinates at random *****/
x=(int)((float)rand()/RAND_MAX*_SIZE);
y=(int)((float)rand()/RAND_MAX*_SIZE);
      }
      while(particle[x][y]==1);

      do{
x1=x;
        y1=y+1;
        if(y1>_SIZE)
  {
    y1=y1-_SIZE;
```

```
  }

        x2=x+1;
        y2=y;
        if(x2>_SIZE)
          {
    x2=x2-_SIZE;
  }

        x3=x;
        y3=y-1;
        if(y3<0)
          {
    y3=y3+_SIZE;
  }

        x4=x-1;
        y4=y;
        if(x4<0)
          {
            x4=x4+_SIZE;
  }

        if(particle[x1][y1]==1||particle[x2][y2]==1
          || particle[x3][y3]==1||particle[x4][y4]==1)
  {
    particle[x][y]==1;
  }
/***** random walk *****/
       else
          {
   r=(float)rand()/RAND_MAX;

   if(r>=0.0&&r<0.25)
     {
       x1=x;
       y1=y+1;
               if(y1>_SIZE)
                 {
```

```
                    y1=y1-_SIZE;
}
    }
          else if(r>=0.25&&r<0.5)
    {
              x1=x+1;
              y1=y;
              if(x1>_SIZE)
                {
                  x1=x1-_SIZE;
                }
    }
          else if(r>=0.5&&r<0.75)
            {
              x1=x;
              y1=y-1;
              if(y1<0)
                {
                  y1=y1+_SIZE;
}
    }
          else if(r>=0.75&&r<=1.0)
            {
              x1=x-1;
              y1=y;
              if(x1<0)
                {
                  x1=x1+_SIZE;
}
    }
  x=x1;
          y=y1;
 }

     }while(particle[x][y]==0);

     fprintf(datafile,"%d %d\n",x,y);
     printf("%d %d\n",x,y);
   }
```

```
  fclose(datafile);
}
```

C　新しいモデルのプログラム

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define  _SIZE  1001
#define _Nparticle 10000

main()
{
  int i,j,k,l,X[10000],Y[10000],particle[_SIZE][_SIZE],x,y;
  static int  hist[1024][1024],histnew[512][512],Ndiv,Nbox;
  double r,r0,p,d=1.0,alpha=3.0,R,dr;
  FILE *datafile1;
  FILE *datafile2;

  datafile1 = fopen("sotuken3.0.dat","w");
  datafile2 = fopen("sotuken-hist3.0.dat","w");

  for(x=0; x<_SIZE; x++)
    {
      for(y=0; y<_SIZE; y++)
        {
          particle[x][y]=0;
}
    }

  for(k=0; k<1023; k++)
    {
      for(l=0; l<1023; l++)
        {
          hist[k][l]=0;
}
    }
  /***** set the origin *****/
  X[0] = _SIZE/2;
  Y[0] = _SIZE/2;
  particle[X[0]][Y[0]]=1;
```

```
   for(i=1; i<_Nparticle; i++)
    {
     do
       {
 /***** choose the  cooordinates at random *****/
        X[i]=(int)((float)rand()/RAND_MAX*_SIZE);
        Y[i]=(int)((float)rand()/RAND_MAX*_SIZE);
       }
        while(particle[x][y]==1);

     do
       {
        r0=sqrt( (X[0]-X[i])*(X[0]-X[i])+(Y[0]-Y[i])*(Y[0]-Y[i]) );

  /*  printf("%d-0 %f\n",i,r0); */

for(j=1; j<i; j++)
          {
     r=sqrt( (X[i]-X[j])*(X[i]-X[j])+(Y[i]-Y[j])*(Y[i]-Y[j]) );

     /* printf("%d-%d %f\n",i,j,r); */

            if(r<r0)
      {
               r0=r;
      }
   }
        p=(float)rand()/RAND_MAX;
        if( p<=pow(d/r0,alpha) )
   {
             particle[X[i]][Y[i]]=1;
   }

/***** random walk *****/
        else
  {
           R=(float)rand()/RAND_MAX;
         if(R>=0.0&&R<0.25)
```

```
    {
      Y[i]=Y[i]+1;
              if(Y[i]>=_SIZE)
{
  Y[i]=Y[i]-_SIZE;
}
            }
          else if(R>=0.25&&R<0.5)
    {
      X[i]=X[i]+1;
              if(X[i]>=_SIZE)
{
  X[i]=X[i]-_SIZE;
}
            }
          else if(R>=0.5&&R<0.75)
            {
      Y[i]=Y[i]-1;
              if(Y[i]<0)
{
  Y[i]=Y[i]+_SIZE;
}
            }
          else if(R>=0.75&&R<=1.0)
            {
      X[i]=X[i]-1;
              if(X[i]<0)
{
  X[i]=X[i]+_SIZE;
}
            }
 }
     }while(particle[X[i]][Y[i]]==0);

   }
  /***** measure the fractal dimension *****/
  Ndiv=1024;
 dr=(double)_SIZE/(double)Ndiv;
```

```
  for(i=1; i<_Nparticle; i++)
    {
      fprintf(datafile1,"%d %d \n",X[i],Y[i]);
      /*  printf("%d %d\n",X[i],Y[i]); */

      hist[(int)(X[i]/dr)][(int)(Y[i]/dr)]++;

    }

  while(Ndiv>1)
    {

    Nbox=0;
    for(k=0; k<Ndiv; k++)
      {
for(l=0; l<Ndiv; l++)
  {
    if(hist[k][l]>0)
      {
Nbox++;
      }
  }
      }

    fprintf(datafile2,"%f %d\n",(double)_SIZE/(double)Ndiv,Nbox);
    /* printf("%f %d\n",(double)_SIZE/(double)Ndiv,Nbox); */

    Ndiv=Ndiv/2;

    for(k=0; k<Ndiv; k++)
      {
        for(l=0; l<Ndiv; l++)
          {
             histnew[k][l]=hist[k*2][l*2]+hist[k*2+1][l*2]
                          +hist[k*2][l*2+1]+hist[k*2+1][l*2+1];
          }
      }

    for(k=0; k<Ndiv; k++)
```

```
      {
        for(l=0; l<Ndiv; l++)
          {
             hist[k][l]=histnew[k][l];
          }
      }
    }

  fclose(datafile1);
  fclose(datafile2);
}
```