卒業論文

# ゲル化のモデルによる
# フラクタルクラスターの再現

益本亮

青山学院大学・理工学部・物理学科
羽田野研究室

２００２年度

# ゲル化のモデルによる
# フラクタルクラスターの再現

益本亮

羽田野研究室

2003年1月20日

**概 要**

ゲル化とは、コロイド溶液の中で分散しているコロイド粒子が、化学結合して固化する現象である。こうして生成されるゲルの構造はフラクタルである事が知られている。本研究では、ゲル化のモデルを導入してコンピューター上でシミュレーションした。その結果、二次元格子と三次元格子上でのフラクタルクラスターの再現に成功した。さらに、フラクタル次元の濃度依存性を出した。

# 目 次

# 1　はじめに

ゲル化によって生成されるゲルは、結晶とは違って不規則な原子配列でできている。こうしてできたゲルの構造は、フラクタルである事が知られている。

本研究ではゲル化のモデルを導入して、コンピューター上でシミュレーションし、フラクタルクラスターを再現した。第 2 章ではゲル化について述べる。第 3 章ではフラクタル図形と、ゲルの構造のフラクタル性について述べ、フラクタル図形の構造の複雑さを定量化したフラクタル次元の測定方法について説明する。第 4 章ではゲル化を導入した格子モデルについて説明する。第 5 章ではシミュレーションによってできたクラスターと、そのフラクタル次元の測定結果について、さらにフラクタル次元の濃度依存性について述べる。

# 2　ゲル化

ゲル化とは、ゾルがゲルに変化することである [1]。ゾルとは、直径 1〜500nm のコロイド粒子が液体の中で分散し、ブラウン運動しているものである。コロイド溶液とも呼ばれる。これに、例えば加熱のような作用をさせると、コロイド粒子が互いにつながって網目構造を形成し、力学的に固体のゲルという物質になる。この変化をゲル化という。ゲルの例としては、ゼリーや豆腐などがある。

# 3　フラクタル

ゲルの網目構造は、フラクタル図形になっているという研究がある。そこでこの節ではフラクタル図形について述べる。

## 3.1　フラクタルとは？

フラクタルとは、スケールを変えても同じ性質を持つものである [2]。つまり、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）を持つものである。フラクタル図形の例として、コッホ曲線 (図 1) を挙げる。

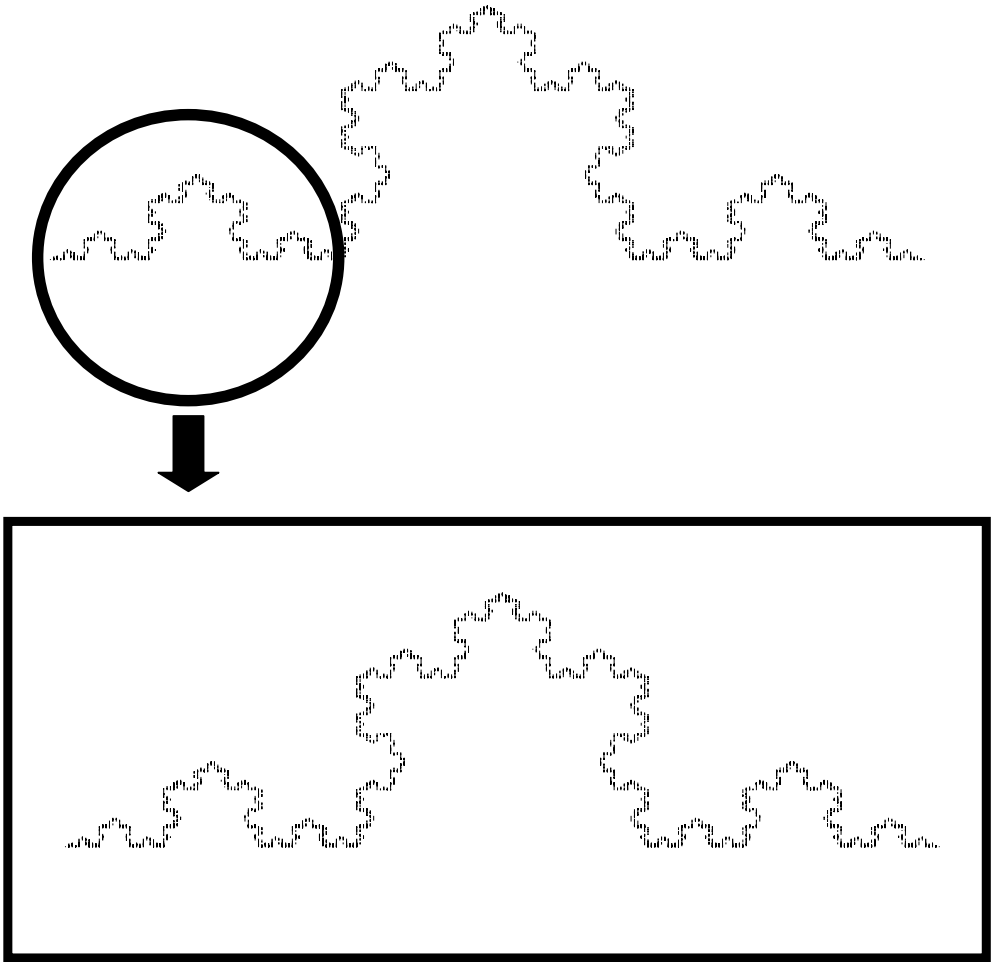

図 1: コッホ曲線。

図1のコッホ曲線の丸で囲んだ部分を拡大すると、もとの図形と同じ図形になっているのが分かる。このような性質を持つものをフラクタルという。自然界のフラクタルの例としては、川の蛇行や海岸線などがある。

## 3.2 フラクタル次元

フラクタル次元とは、フラクタル図形の構造の複雑さを定量的に表したものである。本研究では、粗視化を用いてフラクタル次元を測定した。この節では海岸線を例にして、フラクタル次元とその測定方法について説明する。

海岸線の距離を、次の方法で測定する場合を考える。海岸線を直径 $R$ の円で区切り、円の数 $N(R)$ を数える (図 2)。つまり海岸線の距離を、測定するのに必要とした円の直径の和で粗視化し、近似するのである。

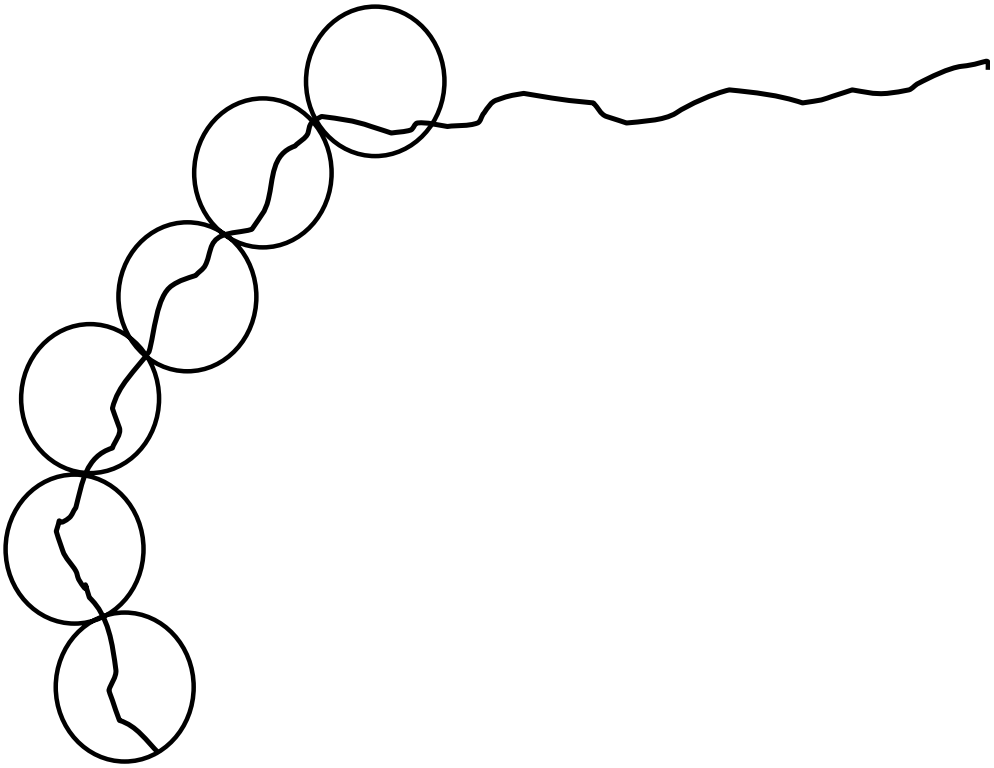

図 2: 海岸線のフラクタル次元を測定する方法。

もし、海岸線がまっすぐであれば、$RN(R)$ がその長さを表すはずだから、

$$N(R) \propto R^{-1}$$

となるはずである。しかし、実際の複雑な海岸線では、

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

の形になる事がある。ここで、$D$ は 1 より大きい定数である。$R$ を小さくすると、$R$ が大きい時に見えなかった複雑な構造が見えてくる。そのため、まっすぐな海岸線に比べ、距離を測定するのに多くの円を必要とするのである。つまり、より複雑な構造の海岸線ほど、$D$ の値は大きくなる。この時の $D$ をフラクタル次元と定義する。

一方、本研究では、シミュレーションで再現したクラスターのフラクタル次元を、次の方法で測定した。二次元格子上のクラスターについては、全体を一辺 $R$ の正方形で区切り、一つでも粒子がある正方形の数 $N(R)$ を数える (図 3)。つまり、クラスターの面積を、粒子がある正方形の面積の和で粗視化し近似する。

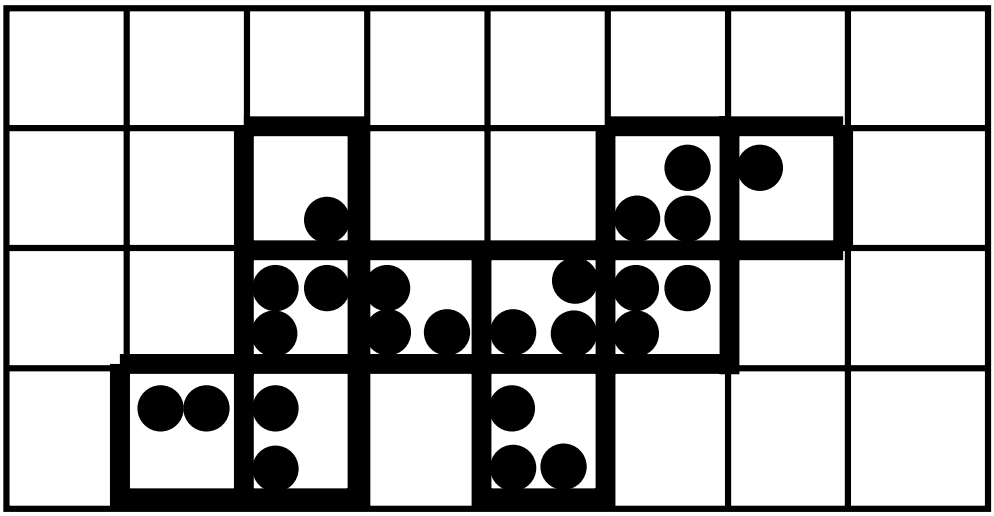

図 3: 二次元格子上のクラスターのフラクタル次元の測定方法。

通常の二次元的クラスターであれば、$R^2N(R)$ がその面積を表すはずなので、

$$N(R) \propto R^{-2}$$

となるはずである。そうではなく、$N(R)$ と $R$ の関係が、2 より小さい定数 $D$ を使って、

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

となれば、フラクタルクラスターが再現された事が確認できる。この時の $D$ をフラクタル次元と定義する。

三次元格子上のクラスターについては、全体を一辺 $R$ の立方体で区切り、一つでも粒子がある立方体の数 $N(R)$ を数える。つまり、クラスターの体積を、粒子がある立方体の体積の和で粗視化し近似する事で、フラクタル次元を求める。通常のクラスターであれば、

$$N(R) \propto R^{-3}$$

となるが、フラクタルクラスターでは指数が 3 より小さくなると期待される。

## 3.3　ゲルの構造のフラクタル性

この節では、ゲルの構造がフラクタルである事を証明する実験例について述べる。

さまざまな大きさの分子をゲルに吸着させて、ゲルの表面積を測定する実験が D.Avnir らによって行なわれた [3, 4]。分子を直径 $R$ の球と考えると、ゲルの表面に吸着した分子の量によってゲルの表面積を測定できる。つまり、ゲルの表面積を吸着した球の断面積の和に粗視化して近似する。これによって、フラクタル次元を測定したのが、彼らの実験である。

実験は、シリカゲル (二酸化ケイ素の微粒子が集まってできたゲル) で行なわれた。その結果、分子の直径 $R$ とゲルに吸着した分子の数 $N(R)$ の間には、

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

の関係が成り立ち、指数 $D$ は、

$$D \cong 2.97$$

つまり、$2 < D < 3$ となる事が確認された。

もし、なめらかな表面を持つ立体でこの実験を行なえば、$D = 2$ になるはずである。しかし、ゲルの構造は複雑なため、小さな分子を吸着させればなめらかな表面よりも分子が多く必要になる (図 4)。よって、$D > 2$ となる。この実験より、ゲルの構造はフラクタルである事が証明され、シリカゲルのフラクタル次元は 2.97 である事が分かった。

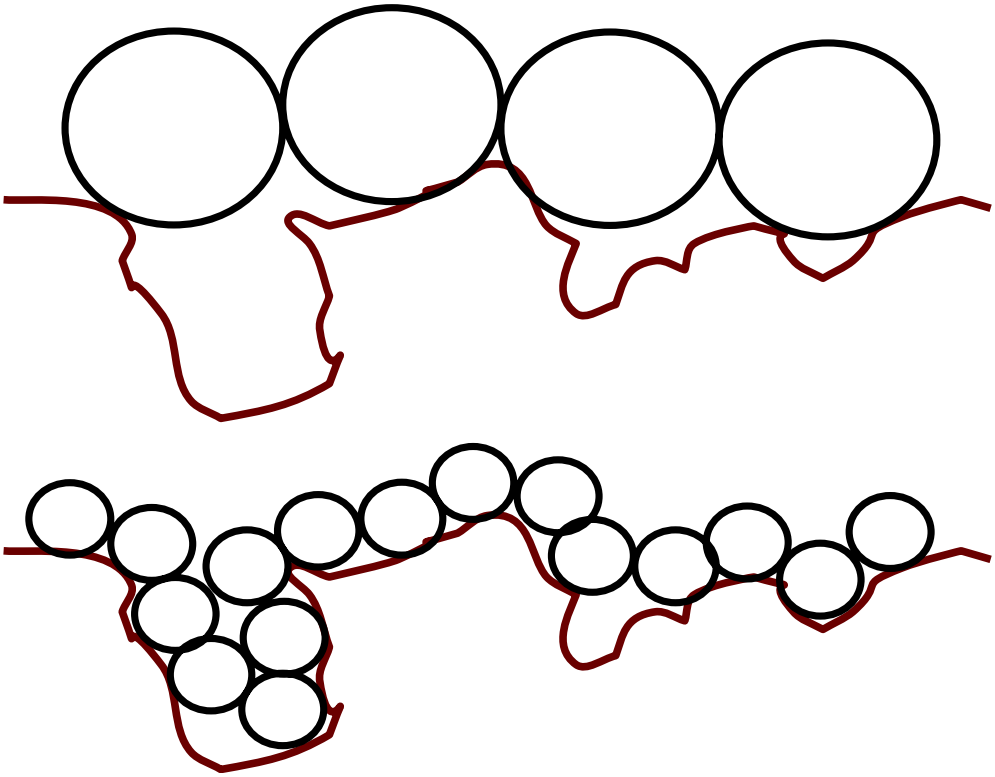

図 4: ゲルの表面に分子を吸着させた図。

# 4　ゲル化の格子モデル

ゲル化のモデルでフラクタルクラスターを再現するために、コロイド粒子がブラウン運動しながら結合する様子を、次のようなモデルで表現した。このモデルを用いて、二次元格子上と三次元格子上でシミュレーションした。

シミュレーションの手順を以下に示す。

1. 格子上に粒子を分散させる。格子には周期境界条件を課す。

2. それぞれの粒子をランダムに動かす。粒子を動かす方向は乱数で決める。図 5 のように、二次元格子では 0 から 4 までの 5 つの乱数、三次元格子では 0 から 6 までの 7 つの乱数を、同じ確率で発生させ、動かす方向を決める。

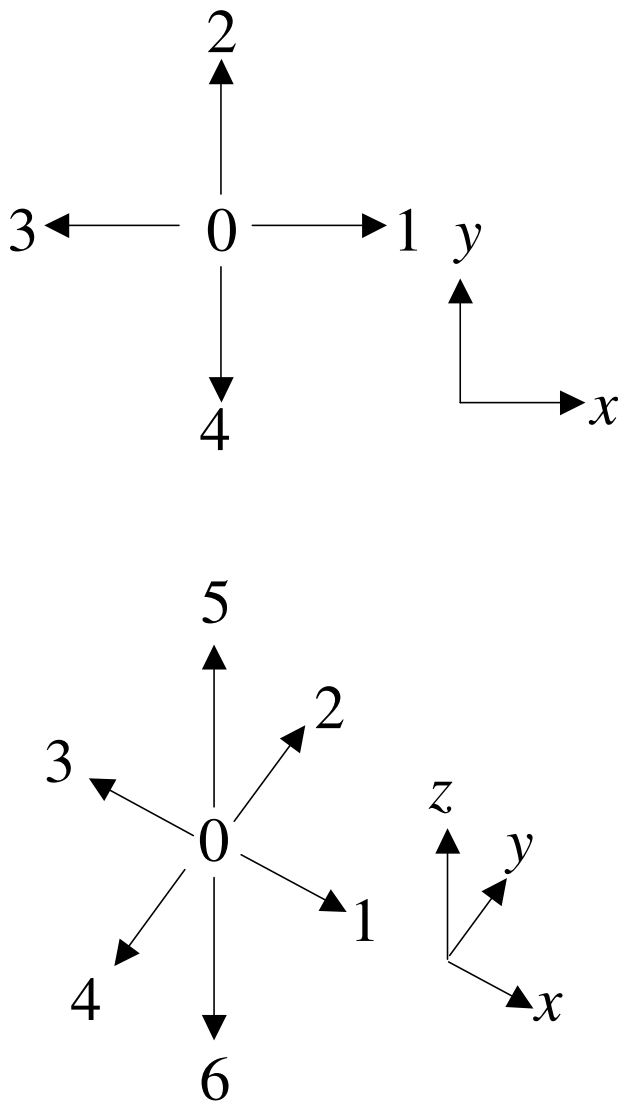


図 5: 乱数と動かす方向 (上図は二次元格子、下図は三次元格子の場合)。

3. ある粒子が別の粒子と隣り合った時、結合させて一つのクラスターにする。その後はクラスターを一体として、ステップ 2 と同様にランダムに動かす。

4. 全ての粒子が一つのクラスターにまとまったところで、シミュレーションを終了とする。

# 5　シミュレーションの結果

## 5.1　二次元格子上でのシミュレーションの結果

前節で定義したモデルを、201 × 201 の二次元格子上でシミュレーションした。10000 個の粒子を分散させてシミュレーションした結果、図 6 のようなクラスターを再現する事ができた。

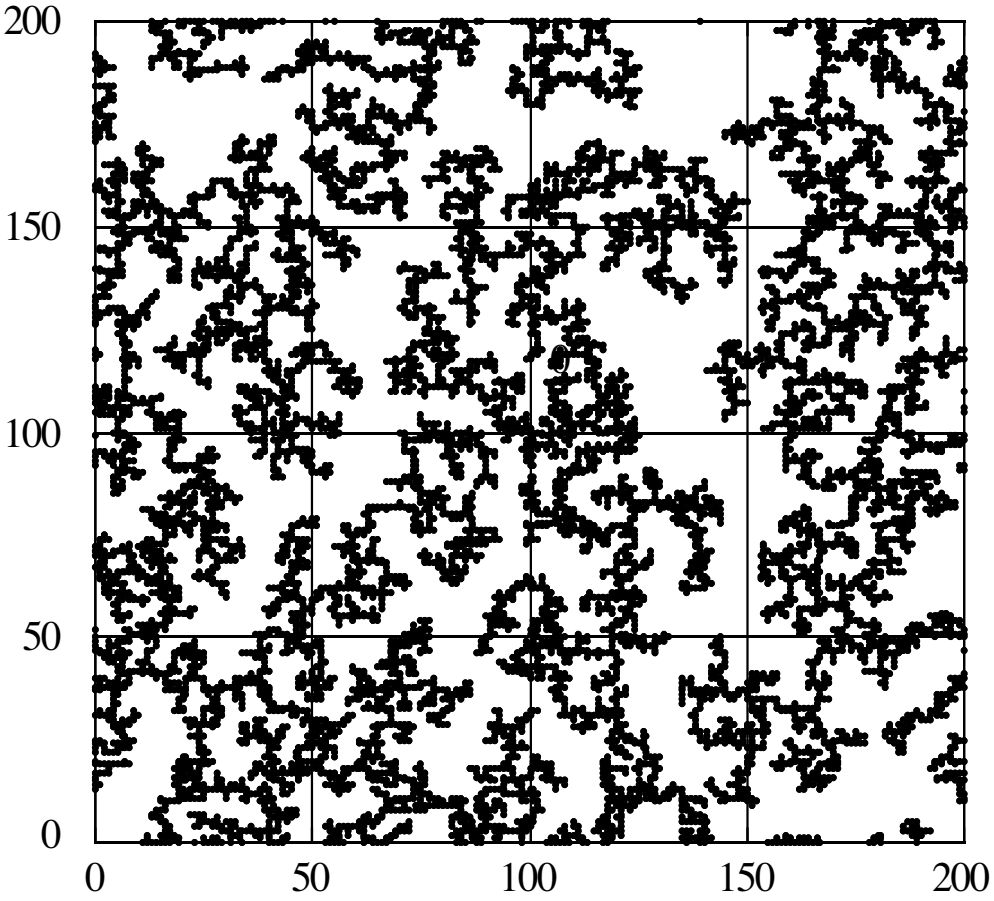


図 6: 二次元格子上でのシミュレーション結果 (粒子数 10000)。

これがフラクタルであるかどうかを、第 3.2 節の後半で述べた次の方法で確認した。全体を一辺 $R$ の正方形で区切り、一つでも粒子がある正方形の数 $N(R)$ を数えた。つまり、クラスターの面積を、粒子がある正方形の面積の和で粗視化し近似する事で、フラクタル次元を求めた。

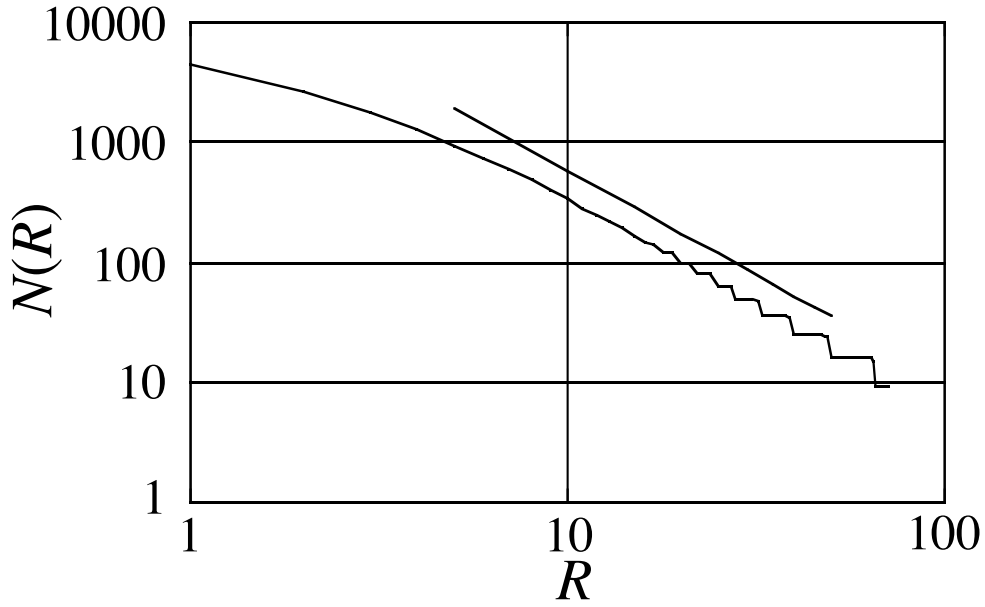


図 7: 図 6 のクラスターのフラクタル次元を測定したグラフ。直線は傾き $-1.74$ の線。

図 7 より、$N(R)$ と $R$ の間には、$5 \leq R \leq 50$ ぐらいの範囲で、

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

の関係が成り立つ。$D$ は、グラフの傾きより $D \cong 1.74$ と評価できる。つまり $1 < D < 2$ となり、フラクタルクラスターを再現できた事が確認できる。この時、$D$ の値がフラクタル次元となるので、図 6 のクラスターのフラクタル次元は 1.74 となる。

次に、フラクタル次元の濃度依存性について調べた。図 8 は、粒子数を 5000、すなわち濃度を図 6 の時の半分にして、シミュレーションした結果である。

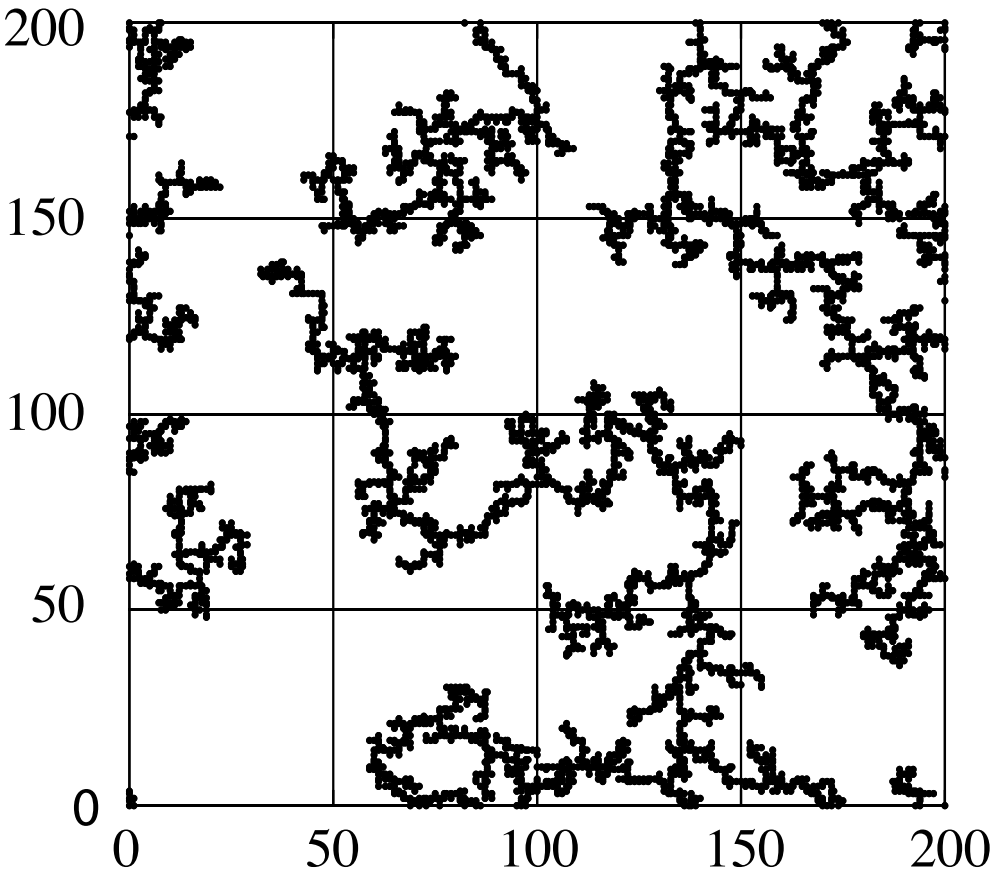


図 8: 二次元格子上でのシミュレーション結果 (粒子数 5000)。

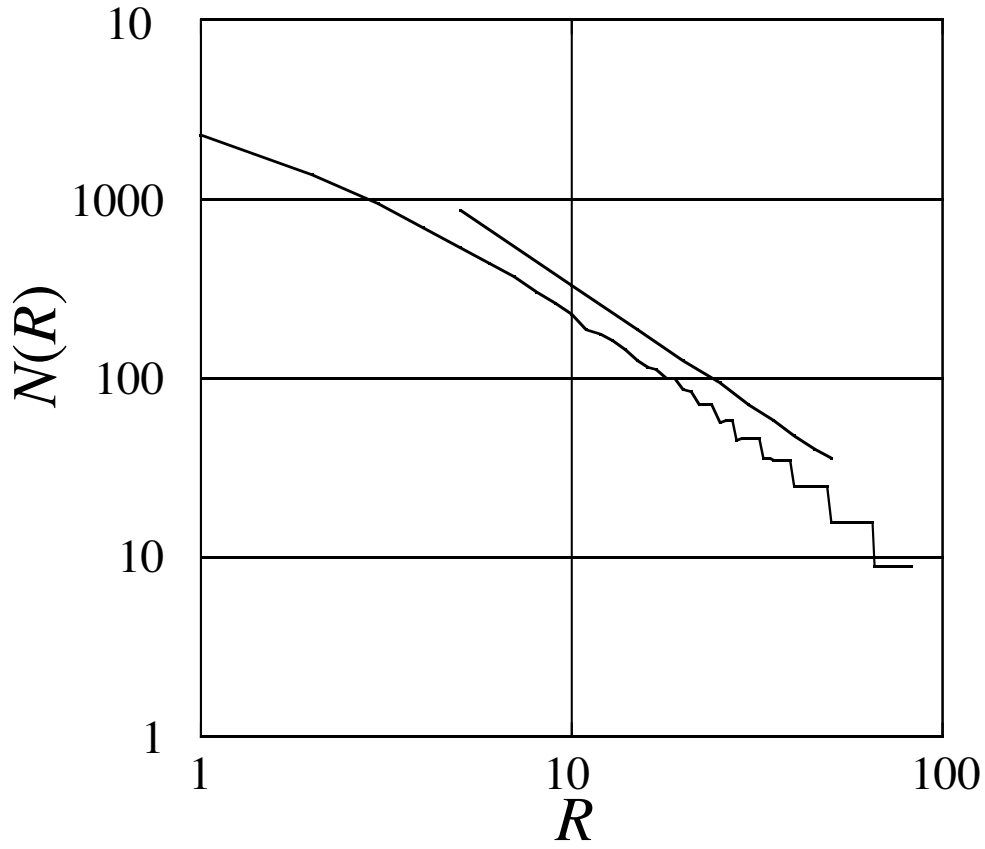


図 9: 図 8 のクラスターのフラクタル次元を測定したグラフ。直線は傾き $-1.39$ の線。

図 8 のクラスターについても、粗視化を使ってフラクタル次元を求めた。図 9 より、粒子数 5000 の時も $R$ と $N(R)$ の間には、$5 \leq R \leq 50$ ぐらいの範囲で、

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

の関係が成り立ち、フラクタルになっている事が確認できた。フラクタル次元は $D \cong 1.39$ となり、粒子数 10000 の時と比べて低くなった。

4つの濃度について、フラクタル次元を測定した結果を図10に示す。ここで、濃度は次のように定義している：

$$\text{濃度（\%）} = \frac{\text{粒子数}}{\text{全格子数}} \times 100. \qquad (1)$$

図10より、濃度が高くなるほどフラクタル次元が高くなることが分かる。

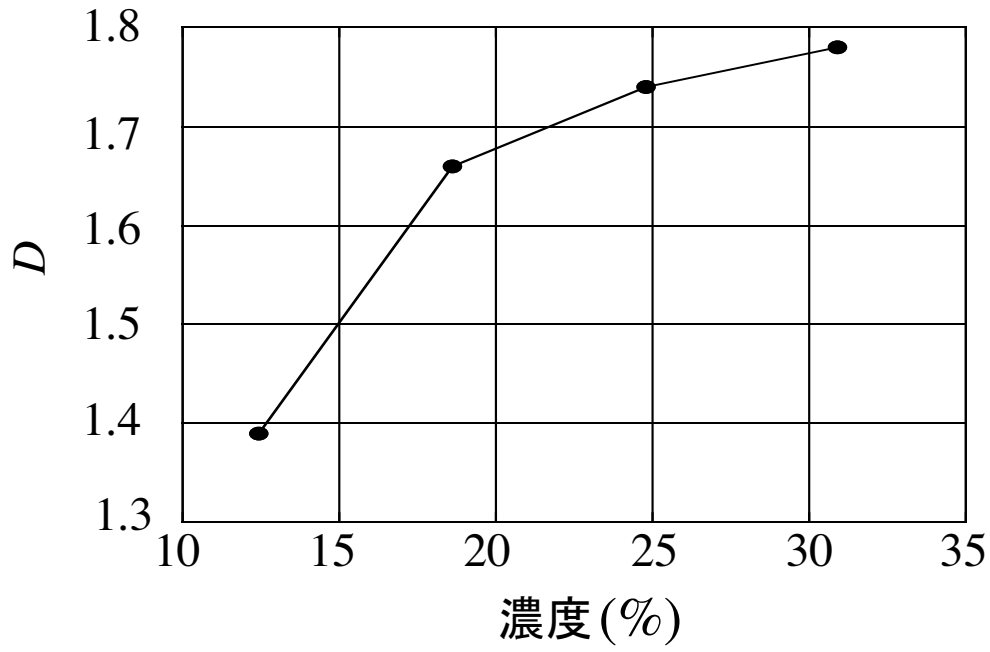


図 10: 濃度とフラクタル次元の関係のグラフ (二次元)。

## 5.2　三次元格子上でのシミュレーションの結果

同じモデルを、$61 \times 61 \times 61$ の三次元格子上でシミュレーションした。30000 個の粒子を分散させてシミュレーションした結果、図 11 のようなクラスターを再現する事ができた。

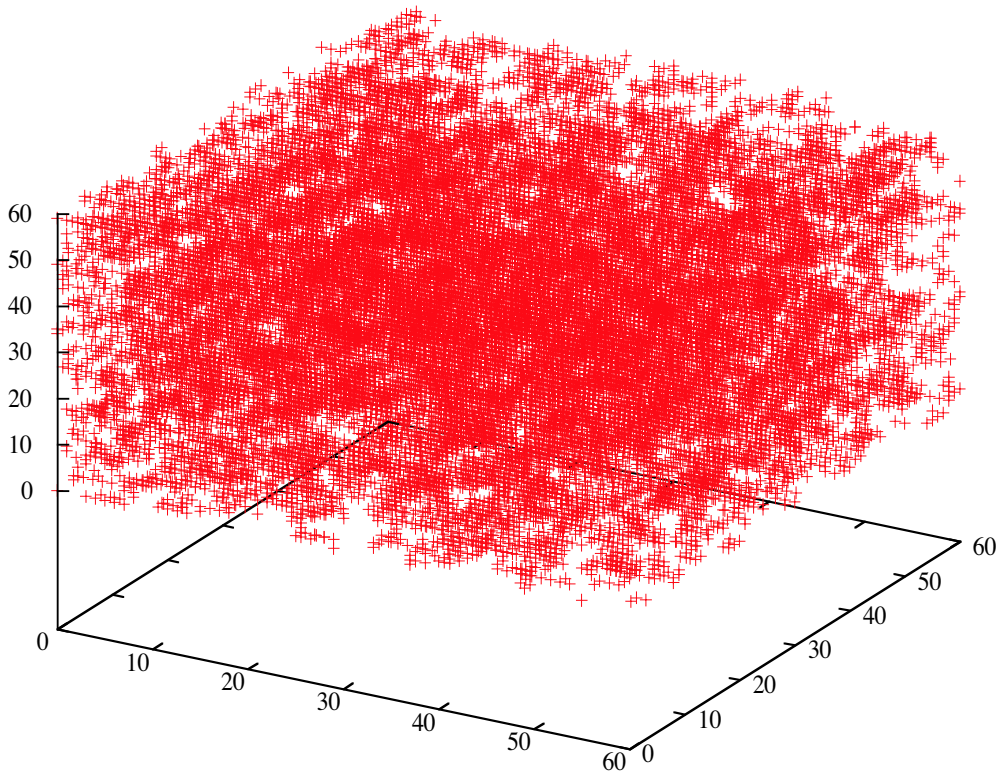


図 11: 三次元格子上でのシミュレーション結果 (粒子数 30000)。

これについてもフラクタルかどうかを確認した。三次元格子上でのクラスターについては、次の方法で確認した。全体を一辺 $R$ の立方体で区切り、一つでも粒子がある立方体の数 $N(R)$ を数えた。つまり、クラスターの体積を、粒子がある立方体の体積の和で粗視化し近似する事で、フラクタル次元を求めた。

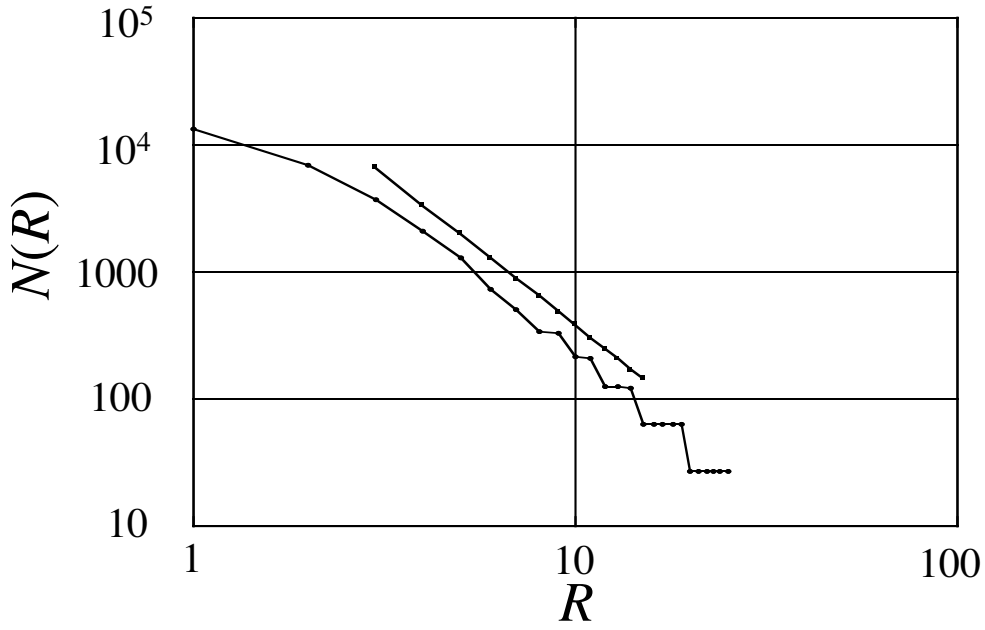


図 12: 図 11 のクラスターのフラクタル次元を測定したグラフ。直線は傾き $-2.37$ の線。

図 12 より、$N(R)$ と $R$ の間には、$3 \leq R \leq 15$ ぐらいの範囲で、

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

の関係が成り立つ。$D$ は、グラフの傾きより $D \cong 2.37$ であった。$2 < D < 3$ となり、フラクタルクラスターを再現できた事が確認できた。よって、図 11 のクラスターのフラクタル次元は 2.37 となる。

三次元格子上のクラスターについても、フラクタル次元の濃度依存性について確認した。4 つの濃度についてフラクタル次元を測定した結果を図 13 に示した。この場合も濃度は、式 (1) のように定義した。図 13 より、三次元でも、濃度が高くなるほどフラクタル次元が高くなることが確認できた。

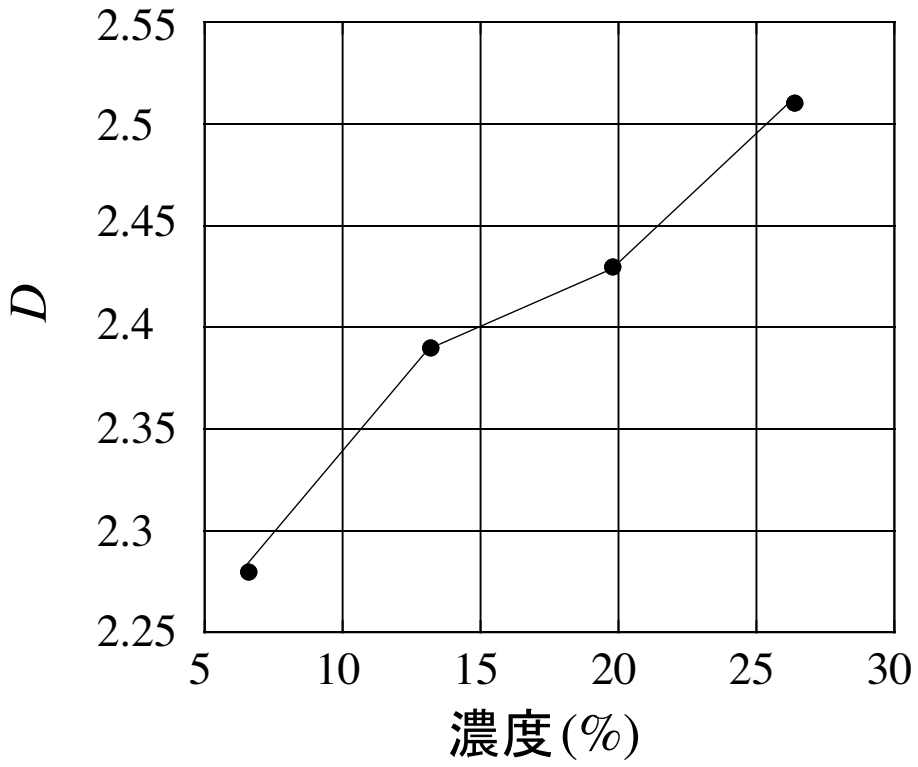


図 13: 濃度とフラクタル次元の関係のグラフ (三次元)。

# 6　まとめ

本研究では、ゲル化のモデルを導入して、フラクタルクラスターを再現することを目的とした。その結果、二次元格子と三次元格子上でのフラクタルクラスターの再現に成功した。

本研究で導入したゲル化のモデルとは、コロイド粒子がブラウン運動しながら結合する様子を、簡単に表現したものだった。よって、そのゲル化におけるコロイド粒子の反応が、ゲルの構造がフラクタルである事と、強く関係している事をコンピューター上で確認できた。

また、濃度が高いほどフラクタル次元が高くなることも発見できた。今後、実験でこれを確かめる事ができれば面白いと考える。

# 7　謝辞

本研究では、羽田野先生には色々とアドバイスを頂きました。また、パソコンの使い方が良く分からず戸惑っている時には、羽田野研究室の先輩方や同級生の皆さんにとても親切に教えて頂きました。本当にありがとうございました。

# 参考文献

[1] 「理化学辞典 第 4 版」(岩波書店、1987)

[2] 高安秀樹「フラクタル」(朝倉書店、1986)

[3] 作花済夫「ゾル-ゲル法の科学」(アグネ承風社、1988)

[4] D.Farin, A.Volpert and D.Avnir, J. Am. Chem. Soc. **107**, 3368-3370 (1985)

# A 二次元格子上でのシミュレーションのプログラム

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define _N 10000 /*Input the number of molecules.*/
#define _L 200  /*Input the length of one side of the square.*/

#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 79742103

#define p_right 0.2
#define p_up 0.2
#define p_left 0.2
#define p_down 0.2

main()
{
  FILE *OUTPUT;
  int i,j,k,l,t;
  int circle_x[_N],circle_y[_N],a[_L+1][_L+1],order[_N],b[_N],c[_N];
  int group[2][_N],walk,repeated,cluster=2,change,time_step=0;
  int max_x,min_x,max_y,min_y;
  float ransuu,p_one,p_two,p_three,p_four;

  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);

  OUTPUT=fopen("2D_N=10000.dat","w");

  /*dispersion*/
  for(i=0;i<=_L;i++){
    for(j=0;j<=_L;j++){
      a[i][j]=0;
```

```
    }
  }

  for(i=0;i<=_N-1;i++){
         circle_x[i]=(int)((_L+1)*sprng());
         circle_y[i]=(int)((_L+1)*sprng());

       if(a[circle_x[i]][circle_y[i]]==0){
          a[circle_x[i]][circle_y[i]]=1;
        }
       else{
i=i--;
       }
  }


  /*group*/

    /*group number*/
  for(i=0;i<=_N-1;i++){
    group[0][i]=i;
  }

    /*neighbor?*/
  while(cluster>=2){
    cluster=0;

    for(t=0;t<=1;t++){

  for(i=1;i<=_N-1;i++){

     for(j=0;j<=i-1;j++){

       if( (circle_x[j]==circle_x[i]+1 && circle_y[j]==circle_y[i])
 ||(circle_x[j]==circle_x[i]   && circle_y[j]==circle_y[i]+1)
```

```
||(circle_x[j]==circle_x[i]-1 && circle_y[j]==circle_y[i])
||(circle_x[j]==circle_x[i]   && circle_y[j]==circle_y[i]-1)
  /*border*/
||(circle_x[j]==0 && circle_x[i]==_L && circle_y[j]==circle_y[i])
        ||(circle_x[i]==0 && circle_x[j]==_L && circle_y[i]==circle_y[j])
||(circle_y[j]==0 && circle_y[i]==_L && circle_x[j]==circle_x[i])
        ||(circle_y[i]==0 && circle_y[j]==_L && circle_x[i]==circle_x[j]) ){


if(group[t][i]!=group[t][j]){

          if(group[t][i]<group[t][j]){
            group[t][j]=group[t][i];
  }
  else{     /* if(group[t][i]>group[t][j]){ */
    group[t][i]=group[t][j];
  }

}
      }

    }
 }
      if(t==0){
        for(i=0;i<=_N-1;i++){
          group[1][i]=group[0][i];
}
      }
      else{ /* if(t==1){ */
        for(i=0;i<=_N-1;i++){
          if(group[0][i]==group[1][i]){
            change=0;
  }
          else{ /* if(group[0][i]!=group[1][i]){ */
            change=1;
```

```
            break;
  }
}
      if(change==1){
        for(i=0;i<=_N-1;i++){
          group[0][i]=group[1][i];
}
        t=t--;
      }
      }
   }

   /*order*/
 for(i=0;i<=_N-1;i++){
   b[i]=0;
 }
 for(j=0;j<=_N-1;j++){
   order[j]=(int)(_N*sprng());
   if(b[order[j]]==0){
     b[order[j]]=1;
   }
   else{   /* if(b[order[j]]==1){ */
     j=j--;
   }
 }

 for(j=0;j<=_N-1;j++){
   /*probability(random_walk)*/
   p_one=p_right;
   p_two=p_one+p_up;
   p_three=p_two+p_left;
   p_four=p_three+p_down;


   /*random_walk*/
```

```
  ransuu=sprng();

  if(ransuu<=p_one){
    walk=1;
  }
  else if(ransuu<=p_two){
    walk=2;
  }
  else if(ransuu<=p_three){
    walk=3;
  }
  else if(ransuu<=p_four){
    walk=4;
  }
  else{
    walk=0;
  }

  if(walk==1){
    for(i=0;i<=_N-1;i++){
      if(group[0][i]==order[j]){
        a[circle_x[i]][circle_y[i]]=0;
        circle_x[i]=circle_x[i]+1;
/*border*/
if(circle_x[i]==_L+1){
          circle_x[i]=0;
}
        else if(circle_x[i]==-1){
          circle_x[i]=_L;
}
        else if(circle_y[i]==_L+1){
          circle_y[i]=0;
}
        else if(circle_y[i]==-1){
```

```
            circle_y[i]=_L;
  }
}
      }
    }
    else if(walk==2){
      for(i=0;i<=_N-1;i++){
        if(group[0][i]==order[j]){
          a[circle_x[i]][circle_y[i]]=0;
          circle_y[i]=circle_y[i]+1;
          /*border*/
  if(circle_x[i]==_L+1){
            circle_x[i]=0;
  }
          else if(circle_x[i]==-1){
            circle_x[i]=_L;
  }
          else if(circle_y[i]==_L+1){
            circle_y[i]=0;
  }
          else if(circle_y[i]==-1){
            circle_y[i]=_L;
  }
}
      }
    }
    else if(walk==3){
      for(i=0;i<=_N-1;i++){
        if(group[0][i]==order[j]){
          a[circle_x[i]][circle_y[i]]=0;
          circle_x[i]=circle_x[i]-1;
          /*border*/
  if(circle_x[i]==_L+1){
            circle_x[i]=0;
  }
```

```
          else if(circle_x[i]==-1){
            circle_x[i]=_L;
  }
          else if(circle_y[i]==_L+1){
            circle_y[i]=0;
  }
          else if(circle_y[i]==-1){
            circle_y[i]=_L;
  }
}
      }
    }
    else if(walk==4){
      for(i=0;i<=_N-1;i++){
        if(group[0][i]==order[j]){
          a[circle_x[i]][circle_y[i]]=0;
          circle_y[i]=circle_y[i]-1;
          /*border*/
  if(circle_x[i]==_L+1){
            circle_x[i]=0;
  }
          else if(circle_x[i]==-1){
            circle_x[i]=_L;
  }
          else if(circle_y[i]==_L+1){
            circle_y[i]=0;
  }
          else if(circle_y[i]==-1){
            circle_y[i]=_L;
  }
}
      }
    }

/*repeated?*/
```

```
      for(i=0;i<=_N-1;i++){
        if(group[0][i]==order[j]){
       if(a[circle_x[i]][circle_y[i]]==0){
           repeated=0;
       }
       else{      /* if(a[circle_x[i]][circle_y[i]]==1){ */
           repeated=1;
           break;
       }
}
      }

      if(repeated==1){

if(walk==1){
          for(i=0;i<=_N-1;i++){
            if(group[0][i]==order[j]){
              circle_x[i]=circle_x[i]-1;
              /*border*/
              if(circle_x[i]==_L+1){
                circle_x[i]=0;
      }
              else if(circle_x[i]==-1){
                circle_x[i]=_L;
      }
              else if(circle_y[i]==_L+1){
                circle_y[i]=0;
      }
              else if(circle_y[i]==-1){
                circle_y[i]=_L;
      }
    }
  }
}
        else if(walk==2){
```

```
            for(i=0;i<=_N-1;i++){
              if(group[0][i]==order[j]){
                circle_y[i]=circle_y[i]-1;
                /*border*/
                if(circle_x[i]==_L+1){
                  circle_x[i]=0;
      }
                else if(circle_x[i]==-1){
                  circle_x[i]=_L;
      }
                else if(circle_y[i]==_L+1){
                  circle_y[i]=0;
      }
                else if(circle_y[i]==-1){
                  circle_y[i]=_L;
      }
    }
  }
}
        else if(walk==3){
          for(i=0;i<=_N-1;i++){
            if(group[0][i]==order[j]){
              circle_x[i]=circle_x[i]+1;
              /*border*/
              if(circle_x[i]==_L+1){
                circle_x[i]=0;
      }
              else if(circle_x[i]==-1){
                circle_x[i]=_L;
      }
              else if(circle_y[i]==_L+1){
                circle_y[i]=0;
      }
              else if(circle_y[i]==-1){
                circle_y[i]=_L;
```

```
      }
    }
  }
}
        else if(walk==4){
          for(i=0;i<=_N-1;i++){
            if(group[0][i]==order[j]){
              circle_y[i]=circle_y[i]+1;
              /*border*/
              if(circle_x[i]==_L+1){
                circle_x[i]=0;
      }
              else if(circle_x[i]==-1){
                circle_x[i]=_L;
      }
              else if(circle_y[i]==_L+1){
                circle_y[i]=0;
      }
              else if(circle_y[i]==-1){
                circle_y[i]=_L;
      }
    }
  }
}
j=j--;
      }

      else{       /* if(repeated==0){ */
  for(i=0;i<=_N-1;i++){
    if(group[0][i]==order[j]){
      a[circle_x[i]][circle_y[i]]=1;
    }
  }
      }
```

```
  }

  /* the number of clusters */
  for(i=0;i<=_N-1;i++){
    c[i]=0;
  }
  for(i=0;i<=_N-1;i++){
    if(c[group[0][i]]==0){
      cluster=cluster++;
      c[group[0][i]]=1;
    }
  }

    time_step=time_step++;
    printf("time_step=%dn",time_step);
    printf("The number of clusters=%dn",cluster);


  }



  for(i=0;i<=_N-1;i++){
    printf("%d %dn",circle_x[i],circle_y[i]);
  }
  for(i=0;i<=_N-1;i++){
    fprintf(OUTPUT,"%d %dn",circle_x[i],circle_y[i]);
  }

  fclose(OUTPUT);

}
```

# B 三次元格子上でのシミュレーションのプログラム

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define _N 30000 /*Input the number of molecules.*/
#define _L 60  /*Input the length of one side of the square.*/

#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 79742103

#define p_right 1.0/7.0
#define p_up 1.0/7.0
#define p_left 1.0/7.0
#define p_down 1.0/7.0
#define p_here 1.0/7.0
#define p_there 1.0/7.0

main()
{
  FILE *OUTPUT;
  int i,j,k,t;
  int circle_x[_N],circle_y[_N],circle_z[_N],a[_L+1][_L+1][_L+1],b[_N],c[_N];
  int group[2][_N],walk,repeated,cluster=2,change,order[_N],time_step=0;
  float ransuu,p_one,p_two,p_three,p_four,p_five,p_six;

  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);

  OUTPUT=fopen("3D_N=30000.dat","w");

  /*dispersion*/
  for(i=0;i<=_L;i++){
    for(j=0;j<=_L;j++){
```

```
      for(k=0;k<=_L;k++){
      a[i][j][k]=0;
      }
    }
  }

  for(i=0;i<=_N-1;i++){
    circle_x[i]=(int)((_L+1)*sprng());
    circle_y[i]=(int)((_L+1)*sprng());
    circle_z[i]=(int)((_L+1)*sprng());

      if(a[circle_x[i]][circle_y[i]][circle_z[i]]==0){
         a[circle_x[i]][circle_y[i]][circle_z[i]]=1;
       }
      else{
i=i--;
      }
  }


  /*group*/

    /*group number*/
  for(i=0;i<=_N-1;i++){
    group[0][i]=i;
  }

    /*neighbor?*/
  while(cluster>=2){
    cluster=0;

    for(t=0;t<=1;t++){

  for(i=1;i<=_N-1;i++){
```

```
    for(j=0;j<=i-1;j++){

      if( (circle_x[j]==circle_x[i]+1 && circle_y[j]==circle_y[i]
   && circle_z[j]==circle_z[i])
||(circle_x[j]==circle_x[i]-1 && circle_y[j]==circle_y[i]
   && circle_z[j]==circle_z[i])
        ||(circle_x[j]==circle_x[i] && circle_y[j]==circle_y[i]+1
   && circle_z[j]==circle_z[i])
        ||(circle_x[j]==circle_x[i] && circle_y[j]==circle_y[i]-1
   && circle_z[j]==circle_z[i])
        ||(circle_x[j]==circle_x[i] && circle_y[j]==circle_y[i]
   && circle_z[j]==circle_z[i]+1)
        ||(circle_x[j]==circle_x[i] && circle_y[j]==circle_y[i]
   && circle_z[j]==circle_z[i]-1)

  /*border*/
||(circle_x[j]==0 && circle_x[i]==_L
           && circle_y[i]==circle_y[i] && circle_z[j]==circle_z[i])
        ||(circle_x[i]==0 && circle_x[j]==_L
           && circle_y[i]==circle_y[j] && circle_z[i]==circle_z[j])
        ||(circle_x[j]==circle_x[i]
           && circle_y[j]==0 && circle_y[i]==_L
           && circle_z[j]==circle_z[i])
        ||(circle_x[i]==circle_x[j]
           && circle_y[i]==0 && circle_y[j]==_L
           && circle_z[i]==circle_z[j])
||(circle_x[j]==circle_x[i] && circle_y[j]==circle_y[i]
           && circle_z[j]==0 && circle_z[i]==_L)
        ||(circle_x[i]==circle_x[j] && circle_y[i]==circle_y[j]
           && circle_z[i]==0 && circle_z[j]==_L) ){

    if(group[t][i]!=group[t][j]){

          if(group[t][i]<group[t][j]){
            group[t][j]=group[t][i];
```

```
  }
  else{     /* if(group[t][i]>group[t][j]){ */
    group[t][i]=group[t][j];
  }
    }
      }

    }
 }
      if(t==0){
        for(i=0;i<=_N-1;i++){
          group[1][i]=group[0][i];
}
      }
      else{ /* if(t==1){ */
        for(i=0;i<=_N-1;i++){
          if(group[0][i]==group[1][i]){
            change=0;
  }
          else{ /* if(group[0][i]!=group[1][i]){ */
            change=1;
            break;
  }
}
      if(change==1){
        for(i=0;i<=_N-1;i++){
          group[0][i]=group[1][i];
}
        t=t--;
      }
      }
   }
```

```
/*random walk*/

for(i=0;i<=_N-1;i++){
  b[i]=0;
}
for(j=0;j<=_N-1;j++){
  order[j]=(int)(_N*sprng());
  if(b[order[j]]==0){
    b[order[j]]=1;
  }
  else{   /* if(b[order[j]]==1){ */
    j=j--;
  }
}

for(j=0;j<=_N-1;j++){

  /*probability(random_walk)*/
  p_one=p_right;
  p_two=p_one+p_up;
  p_three=p_two+p_left;
  p_four=p_three+p_down;
  p_five=p_four+p_here;
  p_six=p_five+p_there;


 /*random_walk*/

  ransuu=sprng();

  if(ransuu<=p_one){
    walk=1;
  }
  else if(ransuu<=p_two){
    walk=2;
```

```
  }
  else if(ransuu<=p_three){
    walk=3;
  }
  else if(ransuu<=p_four){
    walk=4;
  }
  else if(ransuu<=p_five){
    walk=5;
  }
  else if(ransuu<=p_six){
    walk=6;
  }
  else{
    walk=0;
  }

 if(walk==1){
    for(i=0;i<=_N-1;i++){
      if(group[0][i]==order[j]){
        a[circle_x[i]][circle_y[i]][circle_z[i]]=0;
        circle_x[i]=circle_x[i]+1;
/*border*/
if(circle_x[i]==_L+1){
          circle_x[i]=0;
}
        else if(circle_x[i]==-1){
          circle_x[i]=_L;
}
        else if(circle_y[i]==_L+1){
          circle_y[i]=0;
}
        else if(circle_y[i]==-1){
          circle_y[i]=_L;
}
```

```
          else if(circle_z[i]==_L+1){
            circle_z[i]=0;
  }
          else if(circle_z[i]==-1){
            circle_z[i]=_L;
  }
}
      }
    }
    else if(walk==2){
      for(i=0;i<=_N-1;i++){
        if(group[0][i]==order[j]){
          a[circle_x[i]][circle_y[i]][circle_z[i]]=0;
          circle_y[i]=circle_y[i]+1;
  /*border*/
  if(circle_x[i]==_L+1){
            circle_x[i]=0;
  }
          else if(circle_x[i]==-1){
            circle_x[i]=_L;
  }
          else if(circle_y[i]==_L+1){
            circle_y[i]=0;
  }
          else if(circle_y[i]==-1){
            circle_y[i]=_L;
  }
          else if(circle_z[i]==_L+1){
            circle_z[i]=0;
  }
          else if(circle_z[i]==-1){
            circle_z[i]=_L;
  }
}
      }
```

```
    }
    else if(walk==3){
      for(i=0;i<=_N-1;i++){
        if(group[0][i]==order[j]){
          a[circle_x[i]][circle_y[i]][circle_z[i]]=0;
          circle_x[i]=circle_x[i]-1;
  /*border*/
  if(circle_x[i]==_L+1){
            circle_x[i]=0;
  }
          else if(circle_x[i]==-1){
            circle_x[i]=_L;
  }
          else if(circle_y[i]==_L+1){
            circle_y[i]=0;
  }
          else if(circle_y[i]==-1){
            circle_y[i]=_L;
  }
          else if(circle_z[i]==_L+1){
            circle_z[i]=0;
  }
          else if(circle_z[i]==-1){
            circle_z[i]=_L;
  }
}
      }
    }
    else if(walk==4){
      for(i=0;i<=_N-1;i++){
        if(group[0][i]==order[j]){
          a[circle_x[i]][circle_y[i]][circle_z[i]]=0;
          circle_y[i]=circle_y[i]-1;
  /*border*/
  if(circle_x[i]==_L+1){
```

```
            circle_x[i]=0;
  }
          else if(circle_x[i]==-1){
            circle_x[i]=_L;
  }
          else if(circle_y[i]==_L+1){
            circle_y[i]=0;
  }
          else if(circle_y[i]==-1){
            circle_y[i]=_L;
  }
          else if(circle_z[i]==_L+1){
            circle_z[i]=0;
  }
          else if(circle_z[i]==-1){
            circle_z[i]=_L;
  }
}
      }
    }
    else if(walk==5){
      for(i=0;i<=_N-1;i++){
        if(group[0][i]==order[j]){
          a[circle_x[i]][circle_y[i]][circle_z[i]]=0;
          circle_z[i]=circle_z[i]+1;
  /*border*/
  if(circle_x[i]==_L+1){
            circle_x[i]=0;
  }
          else if(circle_x[i]==-1){
            circle_x[i]=_L;
  }
          else if(circle_y[i]==_L+1){
            circle_y[i]=0;
  }
```

```
          else if(circle_y[i]==-1){
            circle_y[i]=_L;
  }
          else if(circle_z[i]==_L+1){
            circle_z[i]=0;
  }
          else if(circle_z[i]==-1){
            circle_z[i]=_L;
  }
}
      }
    }
    else if(walk==6){
      for(i=0;i<=_N-1;i++){
        if(group[0][i]==order[j]){
          a[circle_x[i]][circle_y[i]][circle_z[i]]=0;
          circle_z[i]=circle_z[i]-1;
  /*border*/
  if(circle_x[i]==_L+1){
            circle_x[i]=0;
  }
          else if(circle_x[i]==-1){
            circle_x[i]=_L;
  }
          else if(circle_y[i]==_L+1){
            circle_y[i]=0;
  }
          else if(circle_y[i]==-1){
            circle_y[i]=_L;
  }
          else if(circle_z[i]==_L+1){
            circle_z[i]=0;
  }
          else if(circle_z[i]==-1){
            circle_z[i]=_L;
```

```
  }
}
      }
    }

/*repeated?*/
       for(i=0;i<=_N-1;i++){
         if(group[0][i]==order[j]){
        if(a[circle_x[i]][circle_y[i]][circle_z[i]]==0){
            repeated=0;
        }
        else{      /* if(a[circle_x[i]][circle_y[i]][circle_z[i]]==1){ */
            repeated=1;
            break;
        }
 }
       }

       if(repeated==1){

 if(walk==1){
           for(i=0;i<=_N-1;i++){
             if(group[0][i]==order[j]){
               circle_x[i]=circle_x[i]-1;
               /*border*/
               if(circle_x[i]==_L+1){
                 circle_x[i]=0;
       }
               else if(circle_x[i]==-1){
                 circle_x[i]=_L;
       }
               else if(circle_y[i]==_L+1){
                 circle_y[i]=0;
       }
               else if(circle_y[i]==-1){
```

```
                circle_y[i]=_L;
      }
              else if(circle_z[i]==_L+1){
                circle_z[i]=-1;
      }
              else if(circle_z[i]==-1){
                circle_z[i]=_L;
      }
    }
  }
}
if(walk==2){
          for(i=0;i<=_N-1;i++){
            if(group[0][i]==order[j]){
              circle_y[i]=circle_y[i]-1;
              /*border*/
              if(circle_x[i]==_L+1){
                circle_x[i]=0;
      }
              else if(circle_x[i]==-1){
                circle_x[i]=_L;
      }
              else if(circle_y[i]==_L+1){
                circle_y[i]=0;
      }
              else if(circle_y[i]==-1){
                circle_y[i]=_L;
      }
              else if(circle_z[i]==_L+1){
                circle_z[i]=-1;
      }
              else if(circle_z[i]==-1){
                circle_z[i]=_L;
      }
    }
```

```
  }
}
if(walk==3){
          for(i=0;i<=_N-1;i++){
            if(group[0][i]==order[j]){
              circle_x[i]=circle_x[i]+1;
              /*border*/
              if(circle_x[i]==_L+1){
                circle_x[i]=0;
      }
              else if(circle_x[i]==-1){
                circle_x[i]=_L;
      }
              else if(circle_y[i]==_L+1){
                circle_y[i]=0;
      }
              else if(circle_y[i]==-1){
                circle_y[i]=_L;
      }
              else if(circle_z[i]==_L+1){
                circle_z[i]=-1;
      }
              else if(circle_z[i]==-1){
                circle_z[i]=_L;
      }
    }
  }
}
if(walk==4){
          for(i=0;i<=_N-1;i++){
            if(group[0][i]==order[j]){
              circle_y[i]=circle_y[i]+1;
              /*border*/
              if(circle_x[i]==_L+1){
                circle_x[i]=0;
```

```
      }
              else if(circle_x[i]==-1){
                circle_x[i]=_L;
      }
              else if(circle_y[i]==_L+1){
                circle_y[i]=0;
      }
              else if(circle_y[i]==-1){
                circle_y[i]=_L;
      }
              else if(circle_z[i]==_L+1){
                circle_z[i]=-1;
      }
              else if(circle_z[i]==-1){
                circle_z[i]=_L;
      }
    }
  }
}
if(walk==5){
          for(i=0;i<=_N-1;i++){
            if(group[0][i]==order[j]){
              circle_z[i]=circle_z[i]-1;
              /*border*/
              if(circle_x[i]==_L+1){
                circle_x[i]=0;
      }
              else if(circle_x[i]==-1){
                circle_x[i]=_L;
      }
              else if(circle_y[i]==_L+1){
                circle_y[i]=0;
      }
              else if(circle_y[i]==-1){
                circle_y[i]=_L;
```

```
      }
              else if(circle_z[i]==_L+1){
                circle_z[i]=-1;
      }
              else if(circle_z[i]==-1){
                circle_z[i]=_L;
      }
    }
  }
}
if(walk==6){
          for(i=0;i<=_N-1;i++){
            if(group[0][i]==order[j]){
              circle_x[i]=circle_x[i]+1;
              /*border*/
              if(circle_x[i]==_L+1){
                circle_x[i]=0;
      }
              else if(circle_x[i]==-1){
                circle_x[i]=_L;
      }
              else if(circle_y[i]==_L+1){
                circle_y[i]=0;
      }
              else if(circle_y[i]==-1){
                circle_y[i]=_L;
      }
              else if(circle_z[i]==_L+1){
                circle_z[i]=-1;
      }
              else if(circle_z[i]==-1){
                circle_z[i]=_L;
      }
    }
  }
```

```
}
      }

      else{       /* if(repeated==0){ */
  for(i=0;i<=_N-1;i++){
    if(group[0][i]==order[j]){
      a[circle_x[i]][circle_y[i]][circle_z[i]]=1;
    }
  }
      }



 }

 /* the number of clusters */
 for(i=0;i<=_N-1;i++){
   c[i]=0;
 }
 for(i=0;i<=_N-1;i++){
   if(c[group[0][i]]==0){
     cluster=cluster++;
     c[group[0][i]]=1;
   }
 }

   time_step=time_step++;
   printf("time_step=%dn",time_step);
   printf("The number of clusters=%dn",cluster);


 }
```

```
  for(i=0;i<=_N-1;i++){
    printf("%d %d %dn",circle_x[i],circle_y[i],circle_z[i]);
  }
  for(i=0;i<=_N-1;i++){
    fprintf(OUTPUT,"%d %d %dn",circle_x[i],circle_y[i],circle_z[i]);
  }

  fclose(OUTPUT);

}
```