カオス・フラクタル 講義ノート #11

担当：井上 純一 (情報科学研究科棟 8-13)
URL：http://chaosweb.complex.eng.hokudai.ac.jp/˜j _inoue/index.html

平成 21 年 7 月 14 日

目 次

まずは復習を兼ねて前回 (6/30) のレポート課題についてみていく.

我々がここで考える写像は

$$x_{n+1} = \begin{cases} 3x_n & (x \leq 0.5) \\ -3x_n + 3 & (x > 0.5) \end{cases} \tag{197}$$

である. この写像の形を見て直ぐにわかるのは, $x = 0$ は固定点であり, x はずっと $x = x_0 = 0$ に居座り続けることである. また, $x_0 = 1/3$ も 1 回目の反復で $x_1 = 1$ となり, 2 回目の反復で $x_2 = -3 \cdot 1 - 3 = 0$ となり, 以降ずっと $x = 0$ である. しかし, 例えば $x_0 = 1/2$ に選ぶと, 反復を繰り返すごとに x の値はマイナス方向に大きくなっていき, $-\infty$ に至ることになる. 従って, この写像の初期値の選び方によっては x の行き先が負の無限大に至るものと, ゼロに至るものの 2 通りが存在するのではないかという示唆を与える. 実際に計算機を使っていくつかの初期値に対し, 写像のはじめの数ステップをプロットしたものを図 44(左) に載せよう. この図からわかるように, $x_0 = 1/3, 1/9$ の場合, 数ステップの後に x の値はゼロとなる. 一方, $x_0 = 1/2, 1/5$ の場合にはステップ数の増加とともに, x の値は負の無限大へ行くように見える. そこで, x_0 の値を 0 から細かく刻んで行って, x の絶対値がゼロに収束するもののみをプロットすると図 44(右) のようになる. これは前回の講義で学んだカントール集合に他ならない.

この課題でみたように, 非線形写像の振る舞いを決める初期値条件の集合が自己相似な図形を与える場合がある. 今回はそのような写像を複素数に拡張した場合, 同様な自己相似図形が得られるかをみていく. 講義での解説を聞いているだけではなかなか感じがつかめないと思うので, 各自が計算機を用いて実験されることを強く勧める.

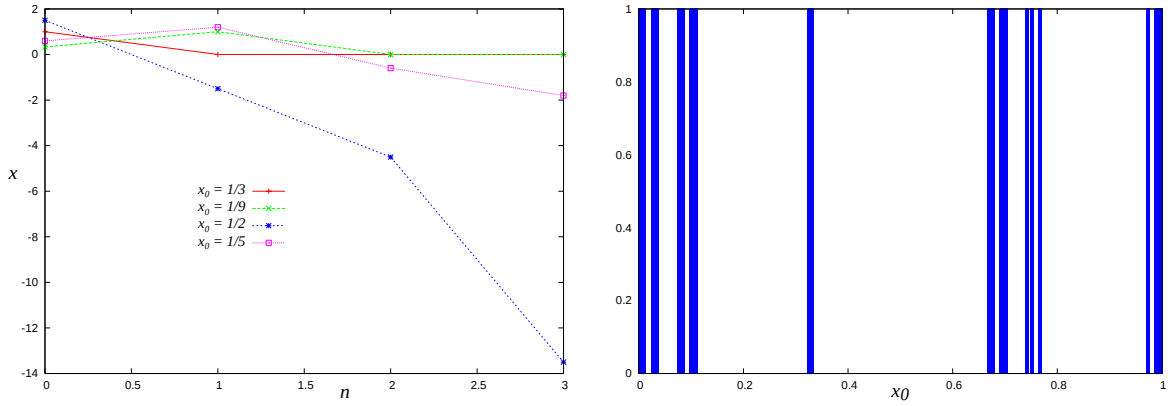


図 44: 写像力学系のゼロへ収束する初期値の集合がカントール集合を与える例. 左図はいくつかの初期値に対する写像 (197) の数ステップの様子. 右図はゼロに収束する初期値の集合 (図では見やすいように「バーコード」で描いている). これはカントール集合となる.

今回の講義は次回 (7/21) にこの講義室で行ってもらう計算機演習の説明も兼ねている. 具体的には, 次回, この講義ノート中に現れる**課題**を時間内に行ってもらう. 90 分間の演習で効率よく作業を進めるためにも, 次回の演習を念頭に今回の講義を聞くと良いであろう.

12　複素力学系

既に [カオス編」で学んだ一次元写像を複素数に拡張した場合を考えてみよう.

12.1　ロジスティック写像の複素数への拡張

既に [カオス編] で学んだロジスティック写像を思い出してみよう.

$$x_{n+1} = ax_n(1-x_n) \tag{198}$$

写像 (漸化式)(198) に含まれるパラメータ a を変えていくと $x_0 \to x_1 \to x_2 \to \cdots \to x_n \cdots$ という時系列は「固定点」「周期軌道」「カオス」というような様々な振る舞いを見せた. また, a の変化に対してその周期軌道の分岐構造がどうであるかは図 45 のようになったことも確かめた. そこでは写像の変数 x を実数であるとしたが, これを複素数 z に拡張した場合に得られる複素数版のロジスティック写像からは何がわかるのかを調べてみよう. 複素数 z の実部を x, 虚部を y で表せば, z は $z = x + iy$ と書けるので, これをロジスティック写像の式 : $z_{n+1} = az_n(1-z_n)$ に代入すれば

$$\begin{aligned} x_{n+1} + iy_{n+1} &= a(x_n + iy_n)(1 - x_n - iy_n) \\ &= a(x_n - x_n^2 + y_n^2) + iay_n(1-2x_n) \end{aligned} \tag{199}$$

が得られる. ここでパラメータ a は $a = a_R + ia_I$ として複素数に選ぶこともできるが, まずは簡単のため実数であるとして話を進めよう.

さて, (199) 式の両辺の実部, 虚部をそれぞれ等しいと置くことにより

$$x_{n+1} = a(x_n - x_n^2 + y_n^2) \tag{200}$$

$$y_{n+1} = ay_n(1-2x_n) \tag{201}$$

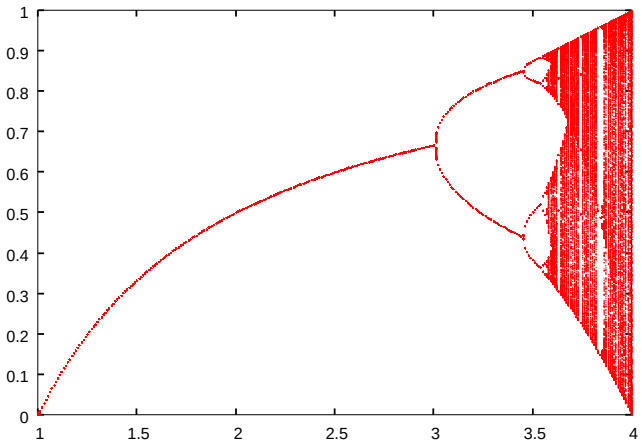


図 45: ロジスティック写像の分岐図

が得られる．この連立漸化式 (200)(201) をある初期条件 $\boldsymbol{x}_0 \equiv (x_0, y_0)$ からスタートさせると，この初期条件の選び方が良ければ連立漸化式は複素平面 x-y 内のある点へと収束するか，有限範囲内での周期軌道へと収束していく．しかし，一方で初期条件の選び方がうまくなければこの連立漸化式は収束せず，$|\boldsymbol{x}_\infty| \equiv \sqrt{x_\infty^2 + y_\infty^2} \to \infty$ のように発散してしまう．従って，全ての可能な初期条件はその初期条件から出発した軌道が [発散してしまうもの] と [有限に留まるもの] との 2 種類に分類されることがわかる (今回の講義ノートで解答例を載せた**レポート課題 10** を参照)．つまり，初期条件の作る集合：$\{\boldsymbol{x}_0\}$ は

$$\begin{aligned} \{\boldsymbol{x}_0\}_\infty &\equiv \{\boldsymbol{x}_0 |\ |\boldsymbol{x}_\infty| \to \infty\} \\ \{\boldsymbol{x}_0\}_{\rm finite} &\equiv \{\boldsymbol{x}_0 |\ |\boldsymbol{x}_\infty| < \infty\} \end{aligned}$$

でそれぞれが定義される部分集合 $\{\boldsymbol{x}_0\}_\infty$ と $\{\boldsymbol{x}_0\}_{\rm finite}$ に分類される．

連立漸化式 (200)(201) を数値的に解き，複素平面 x-y 内にこの $\{\boldsymbol{x}_0\}_{\rm finite}$ をプロットしてできる図は，その作り方が上述のように極めて単純なのにも関わらず非常に複雑な形状を持つ．それを図 46 に示そう．こ

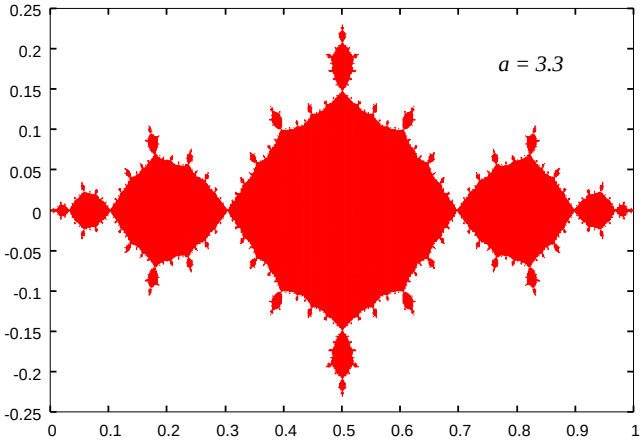


図 46: ジュリア集合．$a = 3.3$

の集合 $\{\boldsymbol{x}_0\}_{\rm finite}$ のことを**ジュリア集合**と呼ぶ．図 46 では $a = 3.3$ に選んだが，この場合に得られるジュリ

ア集合は自己相似性を有する. もちろん, a の値をいろいろ変ええれば様々なジュリア集合が得られる (各自がいろいろ試してみると楽しいと思う). 図 47 にその中から数例を載せる.

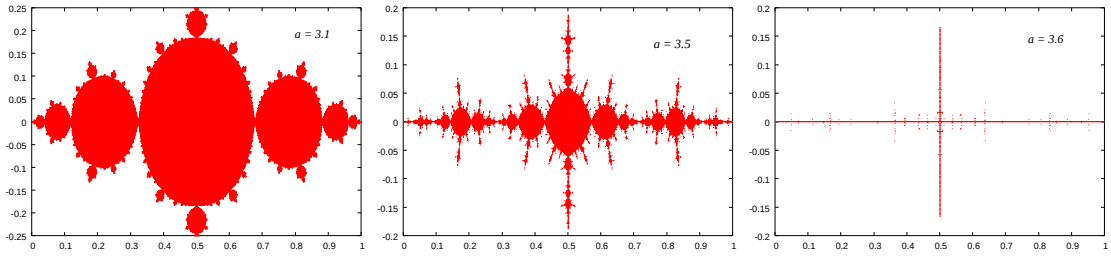


図 47: 左から $a = 3.1, 3.5$, 及び, $a = 3.6$ に対するジュリア集合.

12.2 マンデルブロ集合

前節ではロジスティック写像をコントロールするパラメータ a を実数として扱ったが, 先に述べたようにこれをも複素数に拡張することができる. $a = a_R + ia_I$ として (199) 式に代入し, 実部と虚部にわけた反復式を書き出してみると

$$x_{n+1} = a_R(x_n - x_n^2 + y_n^2) - a_I y_n(1 - 2x_n) \tag{202}$$

$$y_{n+1} = a_R y_n(1 - 2x_n) + a_I(x_n - x_n^2 + y_n^2) \tag{203}$$

が得られる. ジュリア集合を求めたときには, a の値を固定し, 反復式 (202)(203) が有限の値に収まるような初期条件の集合を複素平面内にプロットした. ここでは逆に, 初期条件 $\{x_0, y_0\}$ が一つ与えられた場合, a_R, a_I の値を様々変えたときに, 反復式 (202)(203) の解が有限の値を持つような集合 $\{a_R, a_I\}$ を x-y 平面にプロットしたならばどのような図形が得られるのかを考えてみたい. 図 48 に $x_0 = 0.5, y = 0$ を初期条

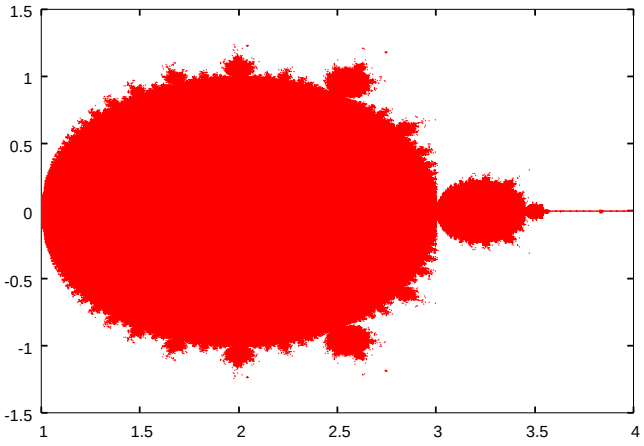


図 48: マンデルブロ集合. $x_0 = 0.5, y_0 = 0$ と初期条件を選んである.

件に選んだ場合の結果を載せる. このような図形のことを**マンデルブロ集合**と呼ぶ.

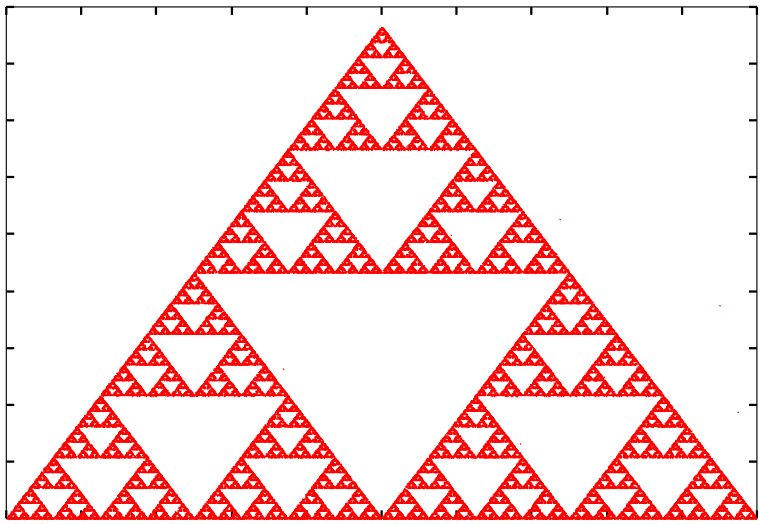


図 49: シェルピンスキー・ガスケット.

13　確率的フラクタル

ここからは確率モデルを用いたフラクタルについて見ていく.

13.1　シェルピンスキー・ガスケット

図 49 は前回学んだシェルピンスキー・ガスケットであり, これは典型的なフラクタル図形の一つである[1]. この図形は次に示すような非常に簡単なルールを用いて生成させることができる.

(1) 図 50 のように頂点を $(X_1, Y_1) = (0, 0)$ (点 O), $(X_2, Y_2) = (2l, 0)$ (点 A), $(X_3, Y_3) = (l, \sqrt{3}l)$ (点 B) とする正三角形を考える.
(2) この正三角形内部の任意の点 $\mathrm{P}^{(0)} = (x_0, y_0)$ を選ぶ.
(3) 数 0,1,2 を**ランダム**に選び
- $0 \Rightarrow$ O と $\mathrm{P}^{(0)}$ を結ぶ線分の中点に点 $\mathrm{P}^{(1)} = (x_1, y_1)$ を置く.
- $1 \Rightarrow$ A と $\mathrm{P}^{(0)}$ を結ぶ線分の中点に点 $\mathrm{P}^{(1)} = (x_1, y_1)$ を置く.
- $2 \Rightarrow$ B と $\mathrm{P}^{(0)}$ を結ぶ線分の中点に点 $\mathrm{P}^{(1)} = (x_1, y_1)$ を置く.

(4) プロセス (3) を十分多数回繰り返す.

上記のアルゴリズムにより, 点列

$$\mathrm{P}^{(0)}(x_0, y_0) \to \mathrm{P}^{(1)}(x_1, y_1) \to \cdots \mathrm{P}^{(n)}(x_n, y_n) \to \cdots$$

を 2 次元平面内にプロットする. 具体的には

x_0 y_0
x_1 y_1

[1] ガスケット (gasket) とは「詰め物」の意味である.

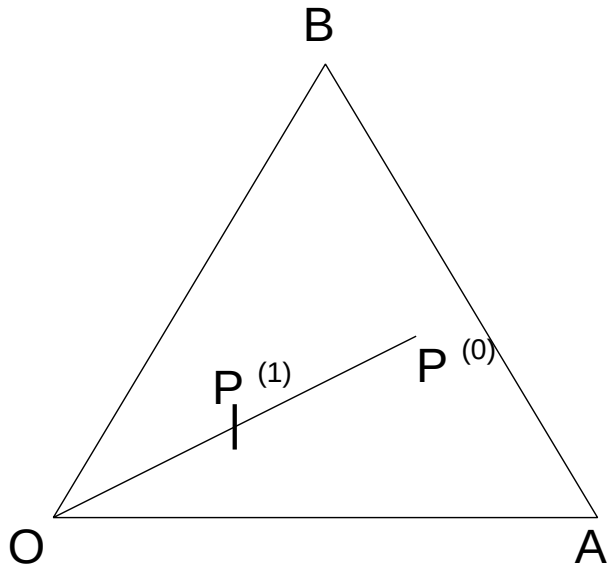


図 50: O が選ばれた場合には $\mathrm{P}^{(0)}$ と O の中点に $\mathrm{P}^{(1)}$ をおく.

x_2 y_2
x_3 y_3
.....

という形式でデータファイル (例えば gasket.dat) に格納する.

課題 1
上記 (1) から (4) のプロセスをコーディングし, シェルピンスキー・ガスケットを描け.

(注) : 実際の作図には gnuplot を用いよ.

13.2 菌糸成長の計算機シミュレーション

前節で見たシェルピンスキーガスケットは確率的なメカニズムがその生成過程に含まれているとはいえ, コンピュータ上に人工的に作られたフラクタル図形であった. しかし, この他にも自然界に目を向ければ, 木, 海岸線など, 数多くのフラクタル構造を持つものが実在する. また, はじめに我々が見たジュリア集合のフラクタル図形はその作り方から明らかなように, 点列 : $P^{(0)}(x_0, y_0) \to P^{(1)}(x_1, y_1) \to \cdots \to P^{(n)}(x_n, y_n) \to \cdots$ は決定論的な規則[2] に従ってグラフ上に点が打ち込まれることにより, 全体がフラクタルとなった.

一方で, 自然界に見られる多くのフラクタル図形の形成メカニズムには確率的要素が重要になってくるものも少なくない. ここでは, その中でも菌糸の成長過程, つまり, 核の周りをある確率に従って動き回る粒子が次々と核に吸着していくことにより複雑な図形が出来上がるプロセスをコンピュータ上にシミュレートし, 出来上がる図形がフラクタルであることを確認する.

図 51 は中央に置かれた菌糸の「種」に周りから**ランダムウォーク (酔歩)** しながらやってくる菌が次々と付着してゆくことによって, この菌糸が成長していく様子をあらわしている. この過程の作図を始める前

[2] これを作図プログラムには乱数の入る余地がない. この意味で確率が介在しない決定論的規則である.

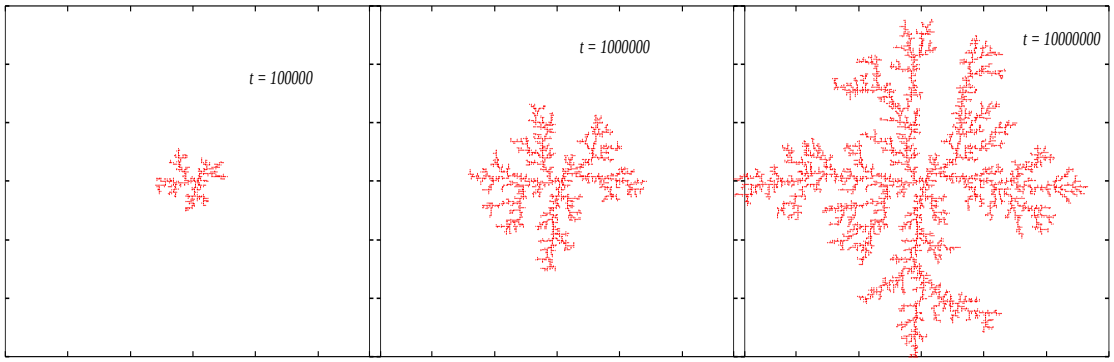


図 51: 菌糸の成長過程. 時間は左から右に流れている.

に, まずは 2 次元格子上のランダムウォークを考えてみよう.

13.2.1 準備 : 2 次元ランダムウォーク

我々の最終目標である「菌糸の成長」をシミュレートするためには 2 次元格子上のランダムウォークのプログラムを用意する必要がある. そこで, ここでは準備としてそれを見てゆくことにする. ランダムウォー

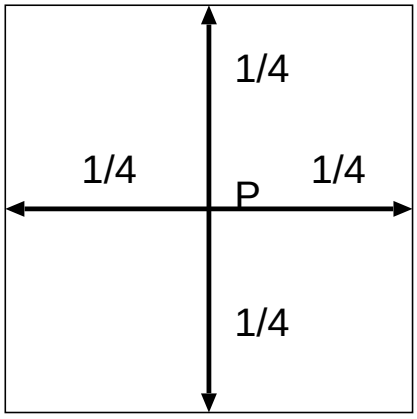


図 52: 点 P から菌は隣接する 4 つの格子点に等確率でジャンプする.

クは任意の点 (例えば原点 $(0,0)$) から出発して, 2 次元正方格子上 (格子の形は何でもよいが, ここでは正方格子とする) で, 等確率で 4 つの方向の中から 1 つをランダムに選択し (図 52 参照), 移動してゆくものである.

課題 2
2 次元正方格子上ランダムウォークの軌跡を描け. 例を図 53 載せる.

13.2.2 菌糸成長のアルゴリズム

以上をふまえて具体的な菌糸成長のアルゴリズムを以下で説明する.

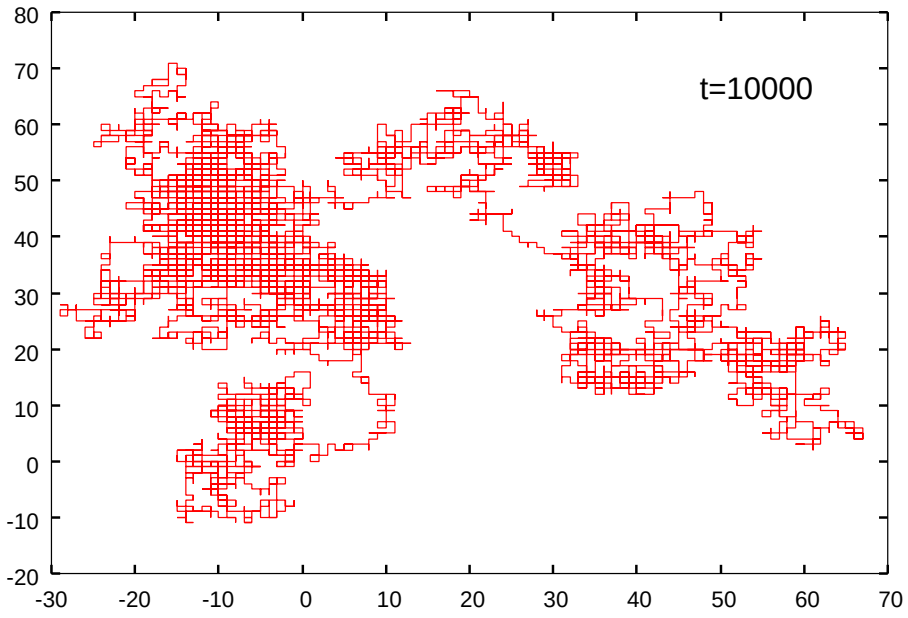


図 53: 2 次元正方格子上のランダムウォークの例.

(1) 菌 (粒子) の可動範囲は縦 × 横 = $l_{\max} \times l_{\max}$ の正方格子上とする. ($l_{\max}$ は偶数). この中央 $S(l_{\max}/2, l_{\max}/2)$ に「種」を置き, 菌糸を取り囲む最小半径 $r_{\max}$ を $r_{\max} = 1$ にセットする (図 54 参照).

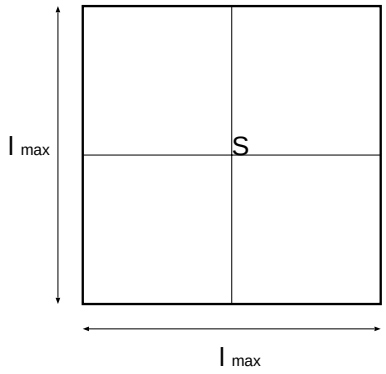


図 54: 菌の可動範囲. 中央に「種」を置く.

(2) S を中心とする半径 $r_s(= r_{\max} + 2.0)$ の円周上の任意の一点 $P(rx, ry)$ をスタート地点に選ぶ. ただし

$$\begin{aligned} rx &= r_s \cos\phi \\ ry &= r_s \sin\phi \end{aligned}$$

とし, ϕ は $\phi \in [0, 2\pi]$ の乱数として選ぶ (図 55 参照).

(3) この点 P をスタート地点に選び, **課題 2** で行った 2 次元正方格子上のランダムウォークをさせる.

(4) $r \equiv \sqrt{(rx)^2 + (ry)^2}$ の大小関係により, 次のように処理を分岐させる.

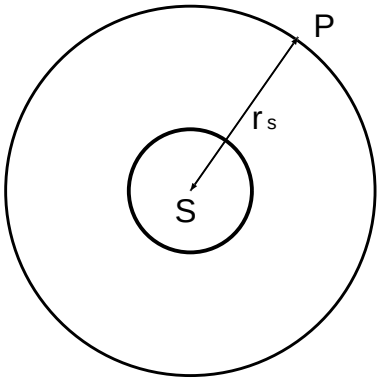


図 55: ランダムウォークの初期地点を半径が r_s の円周上に選ぶ.

(k) r が r_k (この値は自分で適当に決める.) を超えたら, P を置き直す. そして (2) へ戻る.
(c) もし, $r_k > r_d > r_s$ なる $r_d(= r_{\max} + 5.0)$ に対し, $r \geq r_d$ ならば次のようなジャンプをさせる. 図 56

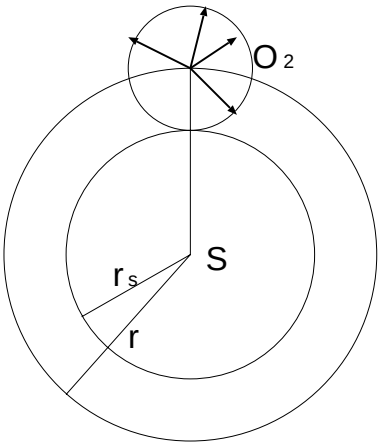


図 56: 半径 $(r - r_s)$ の円 O_2 上にジャンプする.

のように, 中心が半径 r の円周上, 半径が $(r - r_s)$ の円を考え, この円 O_2 上の任意の点にランダムにジャンプさせる. つまり,

$$\begin{aligned} rx &= rx + (r - r_s)\cos\theta \\ ry &= ry + (r - r_s)\sin\theta \end{aligned}$$

とし, θ を $\theta \in [0, 2\pi]$ の乱数として選ぶ.

(a) もし, 粒子の現在地 $(rx + l_{\max}/2, ry + l_{\max}/2)$ に隣接する 4 つの格子点 $(rx + l_{\max}/2 + 1, ry + l_{\max}/2)$, $(rx + l_{\max}/2, ry + l_{\max}/2 + 1)$, $(rx + l_{\max}/2 - 1, ry + l_{\max}/2)$, $(rx + l_{\max}/2, ry + l_{\max}/2 - 1)$ に粒子が存在すれば $(rx + l_{\max}/2, ry + l_{\max}/2)$ に粒子を置き, $r_{\max}$ を

$$r_{\max} = \max\left[r_{\max}, \sqrt{(rx)^2 + (ry)^2}\right]$$

の規則で更新させる. つまり, 1 ステップ前の $r_{\max}$ と比べて大きい方に取り替える. そして (2) へ戻

る.

(j) (k),(c),(a) のいずれでもなければ (3) へ.

課題 3

上のアルゴリズムを具体的にプログラミングして菌糸成長を描いてみよ.

[**プログラム作成上のヒント**]

以下に main 関数のサンプルを載せる. この中にある, check(), int(), occupy(), jump(), aggregate(), longjump(); 等の関数を各自考え, プログラムを完成させるとよい. (もちろん, これを使わないでもよい)

```
/*************************************************************/
/*      菌糸成長のシミュレーション  (main 関数のみ)          */
/*************************************************************/
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#define lmax 620
#define rs (rmax+2.0)
#define rd (rmax+5.0)
#define rkill (5*rmax)
#define PI 3.14159265
/***********************************************/
/*             乱数の種                        */
/***********************************************/
#define SEED1 131
#define SEED2 233
#define SEED3 151

/*  (rx,ry) に粒子がいるかどうかを決める配列. いれば xf[rx][ry]=1, いなけりゃxf[rx][ry]=0. */
int xf[lmax][lmax];

int rx, ry;
double rmax;

main()
{
  FILE *ptr;
  long i,k;
  rmax=10;
```

```
  init();      /* 中央に種を置く関数 */
  occupy();    /* スタート地点 P を選ぶ関数 */
  jump();     /* 任意の点で 4 つの隣接格子点のいずれかにジャンプさせる関数  */

   for(i = 0; i <= 1000000; i++){
    switch(check()) {  /* check() は (4) での分岐条件の k,a,j,c のいずれかを返す関数 */
    case 'k':
      occupy();
      jump();
      break;
    case 'a':
      aggregate();          /* r_max を算出する関数  */
      if((ptr=fopen("test.dat","at")) != NULL){
           fprintf(ptr, "%d %d \n", rx, ry);  /* 粒子を位置 (rx,ry) に置く */
      }
   fclose(ptr);

      occupy();
      jump();
      break;
    case 'j':
      jump();
      break;
    case 'c':
      longjump();  /* 半径 (r-r_s) の円周上へジャンプ. (c) の作業に相当する関数  */
      break;

   }
 }
}
```

13.3 フラクタル次元とその計算方法

前節まででいくつかのフラクタル図形に関して学び, 実際にそれらを作図してきた. ところで, フラクタル図形をスケールを変えてみても同じ図形に見えるものとして定義したが, もう少し定量的にフラクタル図形とその他の図形を区別するための指標はないであろうか ?

ところで「次元」というものに関し, 我々は経験上, 直線は 1 次元, 正方形は 2 次元, 立方体は 3 次元であると認識している. ここでは各図形に関する次元の決め方を我々の経験則とも一致するような形で定義することにより, フラクタル図形の次元 — **フラクタル次元** — を考えてみることにする.

一辺が l の図形の中に含まれる格子点の数を $m(l)$ とする. もし, 格子点が全て同じ重さを持っていたとすればこれは一辺が l の長さの図形の質量を表している (図 57 参). さて, l を 1 から徐々に増加させてみる

と, 正方形の場合, $m(1) = 4, m(2) = 9, \cdots, m(l) = (l+1)^2$ となる. 従って, l が十分大きいときには

$$m(l) \quad = \quad A\, l^2$$

となることはわかるであろう. ここで, A は比例係数である. 同様にして考えると立方体の場合は

$$m(l) \quad = \quad A\, l^3$$

となっていることがわかるであろう.

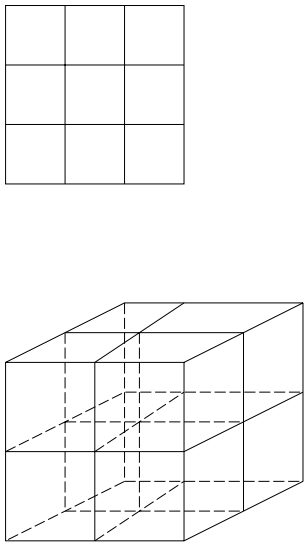


図 57: 正方形と立方体 (書くまでもないが, 念のため).

従って, これを拡張し, 対象とする図形を基本ユニットに分け, その中に含まれる点の数を対象図形 1 辺の長さ l の関数 $m(l)$ として表したとき, l と m の間に

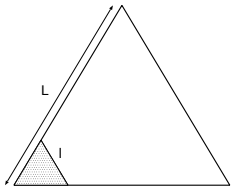
$$m(l) \quad = \quad A\, l^{d_f}$$

なる関係があったとき, この d_f を次元と定義することにしょう. ここで, 既に見たように, このような次元の定義は, 我々の経験則, つまり, 正方形は 2 次元, 立方体は 3 次元, ... に反しないことに注意されたい.

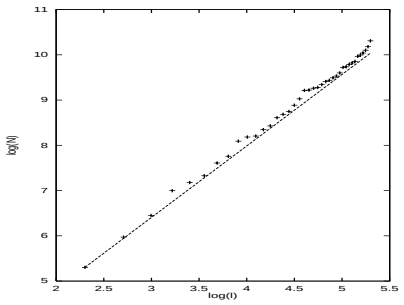
さて, 我々は常識的に次元というものは整数であると思い込んでいる. しかし, 我々が既に作図した 2 つの図形, シェルピンスキーガスケット, 菌糸の成長図は**非整数次元**をもつ. このような d_f を**フラクタル次元 (ボックスカウント次元)** と呼んでいる. フラクタル次元はフラクタル図形を特徴付ける最も基本的な量である. 逆にフラクタル次元が非整数値を持つような図形をフラクタル図形と呼ぶ.
以下の課題で, それぞれの図形のフラクタル次元を具体的に求めてみよう.

課題 4

課題 1 で作図したシェルピンスキー・ガスケットにおいて



長さ l の正三角形に含まれる点の数を N とし, l を 1 から $L = l_{\max}$ までいくつか変化させ (上図), l と $N(l)$ を求める. 次いで，縦軸に $\log N$，横軸に $\log l$ をプロットし, その傾きからフラクタル次元 d_f を求めよ (例えば下図のように).



※ 次回 (7/21) は講義室にて演習を行います．ラップトップ PC を持っている者は持参ください．この講義ノートも持ってくること．