社会システム分析のための統合化プログラム１７

― フラクタルビューア ―

尾崎誠，石丸敬二，福井正康

福山平成大学経営学部経営学科
*福山平成大学大学院経営学研究科

概要

我々は教育分野での利用を目的に社会システム分析に用いられる様々な手法を統合化したプログラム College Analysis を作成してきた。最近はグラフィックにも興味を持ち、３Ｄ表示を利用したいくつかのプログラムを開発してきた。この度はこれらのプログラムの中で、フラクタルビューアというプログラムを紹介する。これは２Ｄと３Ｄのフラクタルを描くツールであるが、既存のマンデルブロ集合などを描く機能も持っている。

キーワード

College Analysis，社会システム分析，統計，OR，意思決定，フラクタル

URL:　http://www.heisei-u.ac.jp/ba/fukui/

１．はじめに

我々はこれまで、統計、経営科学、意思決定に関するプログラムを統合したソフトウェア College Analysis を作成してきたが、これに少し娯楽性を加えて、一般の人にも科学の楽しさを伝えられるプログラムにしたいと思うようになった。この試みの最初の課題は３Ｄであった。そのため、我々は３Ｄビューアという名前の３Ｄグラフィック表示用のユーティリティを作成し [1]、最初に２変量関数グラフを表示するプログラムで利用した [2]。

今回はこれを利用してフラクタル図形が描ける、フラクタルビューア３Ｄというプログラムを作成したので紹介する。また、３Ｄビューアは利用しないが、２次元のフラクタルを表示するフラクタルビューア２Ｄについても紹介する。

フラクタルビューア３Ｄは反復関数を利用する方法や回帰的方法を用いて描かれる３Ｄ画像を作成するツールである。フラクタルは、反復関数法では「点」の集合で表され、回帰的方法では「線」の集合で表される。これに対してフラクタルビューア２Ｄの回帰的方法では、図形は線と面の集合として表される。また、マンデルブロ集合や充填ジュリア集合などのビューアもついており、美しいフラクタル画像の観賞用となっている。これらに共通する再帰的方法では、容易にフラクタル画像が描けるように、簡単なフラクタル描画言語を作成した。この報告ではその利用法についても説明する。

ここで紹介した３次元フラクタルは、３Ｄビューアを用いているため、この中に含まれる種々の機能はすべて利用できる。参考文献 1)を参照してもらいたい。

２．フラクタルビューア２Ｄ

フラクタルビューア２Ｄの実行画面を図 2.1 に示す。

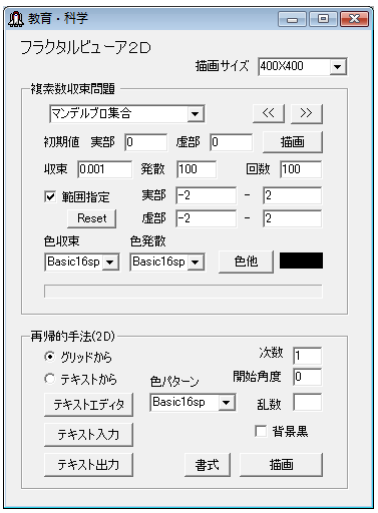


図 2.1 フラクタルビューア２Ｄ

このプログラムは複素数列の収束問題によるフラクタルと再帰的な手法によるフラクタルの２種類を扱う。

2.1 複素数収束によるフラクタル

複素数の収束を利用したフラクタルでは、マンデルブロ集合と充填ジュリア集合が特に有名である。ここではこれらを表示するプログラムについて説明する。

マンデルブロ集合と充填ジュリア集合は、$z_i \in \boldsymbol{C},\ (i = 1, 2, \cdots),\ c \in C$ を用いて以下のように表される。

マンデルブロ集合：　　$\{c|\ lim_{n\to\infty} z_n < \infty, z_n = z_{n-1}^2 + c, z_0 = 0\}$

充填ジュリア集合：　　$\{z_0|\ lim_{n\to\infty} z_n < \infty, z_n = z_{n-1}^2 + c, c = const\}$

これらを図に描く際、美しさを強調するために、例えば $|z_n - z_{n-1}| < 0.001, |z_n| > 100$ のように収束と発散の値を定め、これに到達したnの値を用いて色分けする。繰り返し回数のnの最大値は、「回数」テキストボックスに記入する。これを超える場合は収束も発散もせず、振動する場合と判定するが、これにも色を指定する。色の指定は、収束と発散はコンボボックスを使って既定の色の組み合わせから選び、振動の場合は、「色他」ボタンをクリックしてカラーボックスから選択する。これらの選択の仕方によって全く印象の異なる図となる。

図 2.1 では、まず集合名を選択する。ここではまだ、上の２つしか入っていないが、必要に応じて増やす予定である。次に初期値（定数値）の値を示す。マンデルブロ集合では z_0（デフォルトで$z = 0$）の値であり、充填ジュリア集合では c（デフォルトで$c = 0.3 + 0.3i$）の値である。

描画の範囲は、「範囲指定」チェックボックスにチェックを入れ（デフォルト）、「描画」ボタンをクリックすると、実部と虚部で指定された範囲で描画する。画面上の一部を拡大したい場合は、画面上でマウスをドラッグすると、描画範囲が赤い線で選択され、そのまま「描画」ボタンをクリックすると選択範囲が拡大表示される。「範囲指定」チェックボックスにチェックがない場合、領域を選択ていないときにはデフォルトの範囲になるが、「範囲指定」はチェックしておく方が分り易い。実部と虚部の範囲をデフォルトに直す時には「Reset」ボタンをクリックする。

何度も拡大するといくつか前の段階の画像を見たくなることがある。その時には「<<」「>>」ボタンで前後の画像を表示することができる。その際には、その画像の範囲や次の画像へ移った際の選択範囲などが表示される。

例として一部分を拡大していったマンデルブロ集合の図を図 2.2a、図 2.2b、図 2.2c、図 2.2d に示す。

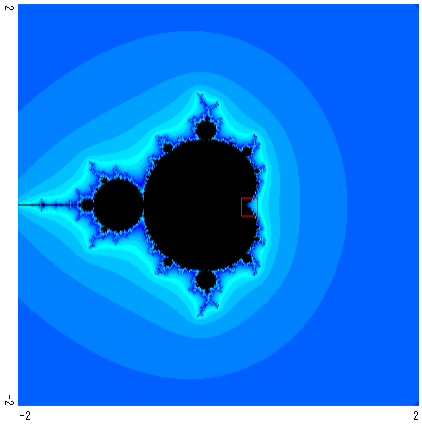


図 2.1.1a マンデルブロ集合

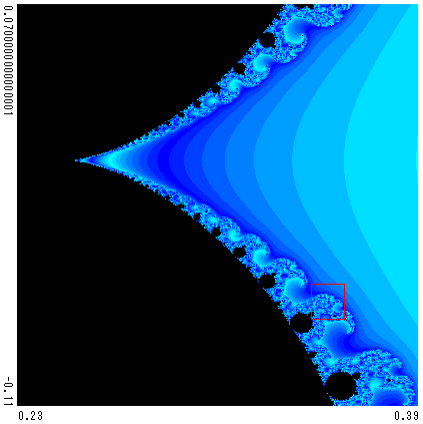


図 2.1.1b 拡大 1

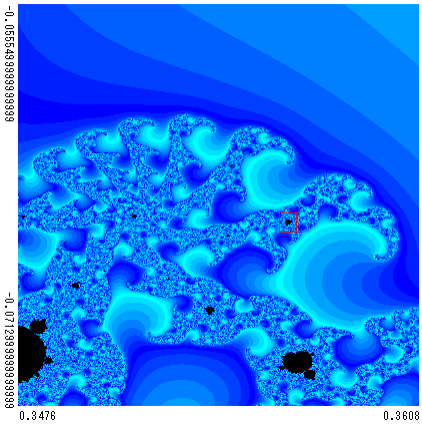


図 2.1.1c 拡大 2

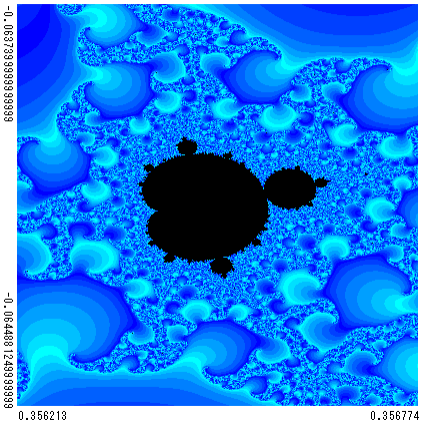


図 2.1.1d 拡大 3

これらの集合以外で、我々のプログラムに含まれる集合とそのサンプルを以下に示す。

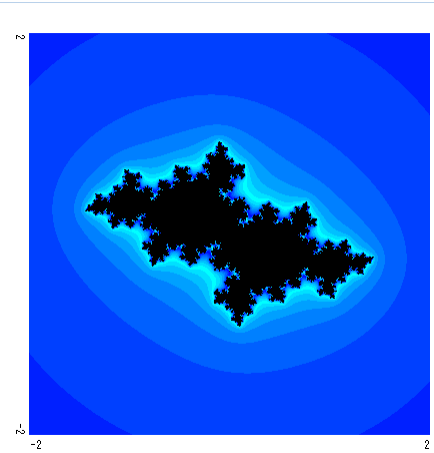


図 2.1.2a 充填ジュリア集合（C≒-0.59+0.43i ）

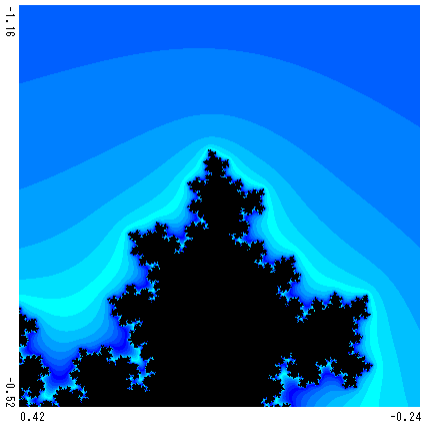


図 2.1.2b 拡大

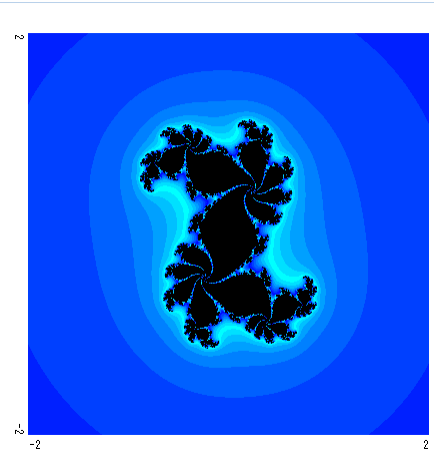


図 2.1.3a 充填ジュリア集合（C≒0.38+0.22i ）

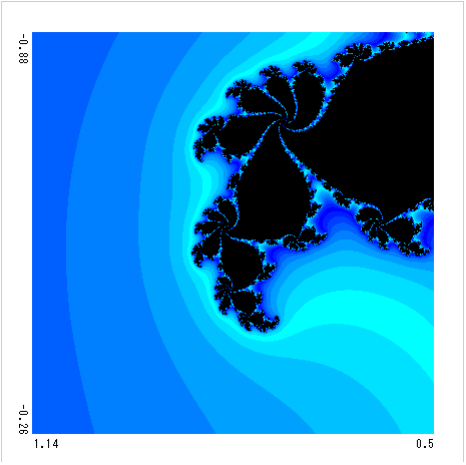


図 2.1.3b 拡大

2.2 再帰的方法によるフラクタル

再帰によるフラクタル画像の描画はプログラミングの基礎として多くのプログラマが経験する。これは簡単な課題であるが、最近はプログラム処理系のグラフィックの扱いに多少の基礎知識を必要とするので、多くの人が手軽に、というわけには行かない。そこで我々はこれらの知識に煩わされることなく、再帰処理だけを頭に入れてフラクタル画像が作れる簡単なマクロ言語を開発した。これを用いることで、何の準備もなく２Ｄのフラクタル画像を作成することができる。また、この言語はかなりの部分で３Ｄのフラクタルに応用できるので、次章の内容と合わせて読んでもらいたい。

まず、簡単な例を示す。図 2.1 のメニューで「テキスト」からラジオボタンをチェックし、「テキストエディタ」ボタンをクリックする。テキストエディタが開かれるが、この中にプログラムを書き込み、良いものができたら「テキスト入力」ボタンをクリックしてグリッドへコピーし保存するのがよい。グリッドからテキストへプログラムを書き出すのは、「テキスト出力」ボタンを利用する。

プログラムは基本的に LOGO のタートルグラフィックスのようになっており、左端 (0,0) から右端 (1,1) へ向けてスタートする。命令はコマンドとパラメータからなっており、パラメータがない場合もある。パラメータには数式も使える。

プログラムのサンプルを図 2.2.1 に示す。

```
turn 45
frac 1/2^0.5
turn -90
frac 1/2^0.5
```

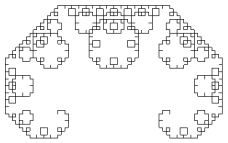


図 2.2.1a サンプル 1　　図 2.2.1b 次数 1　　図 2.2.1c 次数 12

turn 45 は反時計回りで角度 45°、frac 1/2^0.5 はスケール$1/\sqrt{2}$ の相似図形を左の角度で貼り付けることを意味する。メニューに「開始角度」テキストボックスがあるが、これを 90° にすると、上向きに描画が始まり、図 2.2.1b と図 2.2.1c が反時計回りに 90° 回転した図形となる。

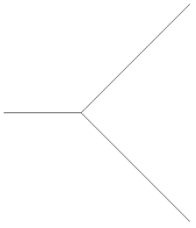
同様の例は有名なコッホ曲線である。図 2.2.2 にプログラムと描画サンプルを示す。

```
frac 1/3
turn 60
frac 1/3
turn -120
frac 1/3
turn 60
frac 1/3
```

図 2.2.2a サンプル 2　　図 2.2.2b 次数 1　　図 2.2.2c 次数 7

これは説明の必要がないであろう。同じような例が続いたので、次は分岐がある場合の例である。

```
go 0.3
separate
turn 45
frac 0.6
return
turn -45
frac 0.6
```



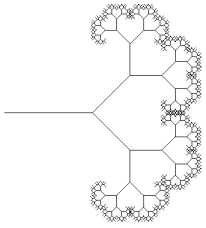


図 2.2.3a サンプル 3　　図 2.2.3b 次数 1　　図 2.2.3c 次数 9

go 0.3 は 0.3 のスケールで線を引く命令である。次の separate はこの時のタートルの位置と向きを

記憶する命令である。分岐する場合によく使われるので separate にした。5 行目にある return は、前の separate で記憶した状態に戻す命令である。残念ながら現在は separate ～ return のネスト構造には対応していない。

次は、4 角形を用いたサンプルである。

```
cfix
polygon4 1
turn 20
fracp 0.7
```

図 2.2.4a サンプル 4

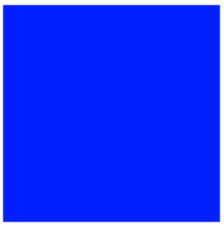


図 2.2.4b 次数 1

図 2.2.4c 次数 10

cfix は描画の際に図形の中心を基準にすることを意味する。これがなければ左下が基準である。次の polygon4 1 はスケール 1（1 辺の長さが 1）の四角形を描画する命令である。描画が終わった段階でタートルは図形の中心から、その時のタートルの向きに図形のスケールだけ進んでいる。最後の fracp 0.7 は通常の frac 0.7 とは異なる。frac の場合は次数 1 の場合にも 2 つの四角形が描かれてしまう。fracp はこれを止めるためのフラクタルの予定地のようなものである。フラクタルは置くが、次の次数から表示する命令である。ちなみに色パターンは「Aqua」を利用している。

次の例は 3 角形で構成されるよく知られたシェルピンスキーのギャスケットである。

```
separate
fract3 0.5
fract3 0.5
return
turn 60
jump 0.5
turn -60
fract3 0.5
```

図 2.2.5a サンプル 5

図 2.2.5b 次数 1

図 2.2.5c 次数 6

ここでは fract3 0.5 があるが、これは 3 角形も書いてフラクタルも貼り付けるという意味である。このような書式を利用すると、次数 1 で 3 角形が 3 つになり、それに合わせて各次数で 3 角形が増える。また jump 0.5 は線を引かずにタートルを飛ばす命令である。

次数 1 で 3 角形を 1 つにしたかったら、少し長くなるが、図 2.2.6 のようなプログラムにする。

```
separate
polygon3 1
return
separate
fracp 0.5
fracp 0.5
return
turn 60
jump 0.5
turn -60
fracp 0.5
```

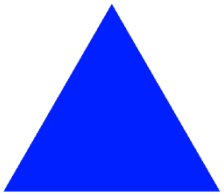
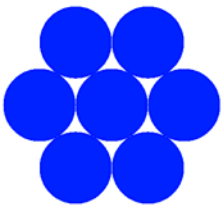


図 2.2.6a サンプル 6　　図 2.2.6b 次数 1　　図 2.2.6c 次数 6

図 2.2.7 は円を使ったサンプルである。cfract 1/3 は比率 1/3 の円を書いて、フラクタルを貼り付ける命令である。プログラムは途中から省略している。

```
cfix
separate
cfract 1/3
return
jump 1/3
cfract 1/3
return
turn 60
jump 1/3
cfract 1/3
return
…
```



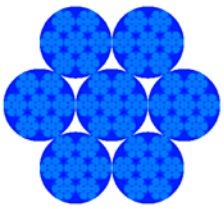
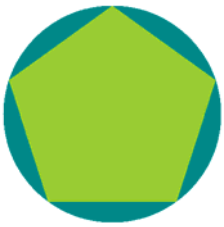


図 2.2.7a サンプル 7　　図 2.2.7b 次数 1　　図 2.2.7c 次数 4

図 2.2.8 は色に乱数を使った例である。

```
drep
cfix
separate
color int(16*rnd)
circle 1
return
color 14
pentagon 1
return
fracp 0.81
```



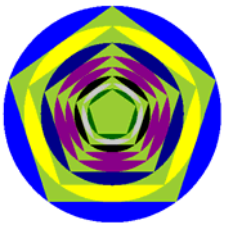


図 2.2.8a サンプル 8　　図 2.2.8b 次数 1　　図 2.2.8c 次数 8

drep は正多角形の比率を外接円の直径にする宣言である。図形の色は色パターンと描画の次数によって自動的に決まるが、自分で指定することもできる。color 命令は直後の図形描画の色を指定する。ここではパラメータの中に乱数 rnd を使っている。

以上いくつかサンプルを示したが、マクロの簡単なまとめを表 2.1 に示す。

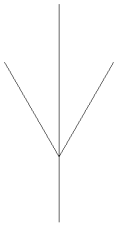
表 2.1 フラクタルビューア２Ｄの再帰処理書式

CFIX		正多角形の起点を中心に設定（お勧め）【宣言】
DREP		多角形の比率を外接円の直径表示に設定【宣言】
CONNECT	［色番号］	フラクタル同士を色番号の線でつなぐ　【宣言】
FRACP	比率	描画をしない再帰処理
FRACT3	比率	常に正三角形を描画する再帰処理
POLYGON3	比率	正三角形の描画
TRIANGLE	比率	正三角形の描画
FRACT4	比率	常に正方形を描画する再帰処理
POLYGON4	比率	正方形の描画
SQUARE	比率	正方形の描画
FRACT5	比率	常に正五角形を描画する再帰処理
POLYGON5	比率	正五角形の描画
PENTAGON	比率	正五角形の描画
FRACT6	比率	常に正六角形を描画する再帰処理
POLYGON6	比率	正六角形の描画
HEXAGON	比率	正六角形の描画
CFRACT	比率	常に円を描画する再帰処理
CIRCLE	比率	円の描画
FRAC	比率	最後の次数だけ直線描画の再帰処理
FRACT	比率	常に直線を描画する再帰処理
FRACC		強制的な終点へ連結する再帰処理
GO	比率	常に直線描画（標準/GOT）
GOF	比率	最後の次数だけの直線描画
GOC		強制的な終点へ連結する直線描画
TURN	角度	進行方向の角度変化[度]）
REVERSE		裏返し
SEPARATE		分岐の開始点・状態の保存（ネスト構造はまだ未対応）
RETURN		分岐への状態の戻り
RESET		進行方向・回転角の初期値再設定

注）多角形の場合、比率は１辺の長さで表す。
　　(DREP があるときは外接円の直径)
注）円の場合、比率は直径で表す。

最後にいくつかの２次元フラクタルの例をあげておこう。

```
go 0.3
separate
frac 0.7
retrun
turn 30
frac 0.3
return
turn -30
frac 0.3
return
```



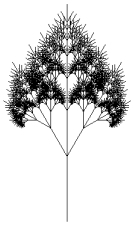


図 2.2.9a サンプル９　　図 2.2.9b 次数 1　　図 2.2.9c 次数 10

```
turn -45
jump 1/2^0.5
turn 180
frac 1/2^0.5
turn -180
jump 1/2^0.5
turn 90
frac 1/2^0.5
```

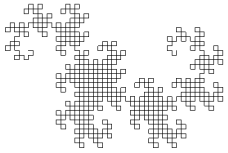


図 2.2.10a サンプル９　　図 2.2.10b 次数 1　　図 2.2.10c 次数 10

```
turn 60
jump 1/2
separate
turn -60
frac 1/2
return
separate
turn 180
frac 1/2
return
turn -60
jump 1/2
turn -60
jump 1/2
turn 180
frac 1/2
```

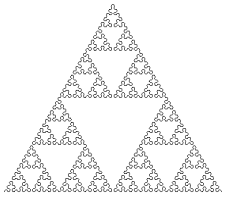


図 2.2.11a サンプル１０　　図 2.2.11b 次数 1　　図 2.2.11c 次数 7

３．フラクタルビューア３Ｄ

３次元空間へのフラクタルの描画は、反復関数による点の描画と再帰的方法による直線の描画で行われる。図 3.1 にフラクタルビューア３Ｄのメニュー画面を示す。

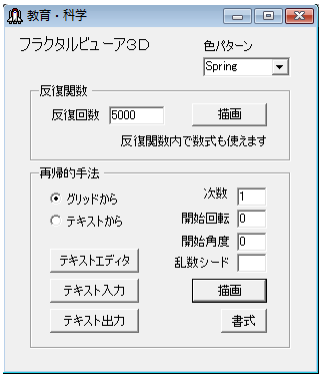


図 3.1 フラクタルビューア３Ｄメニュー

ここでは反復関数による方法と再帰的方法を順番に説明する。

3.1 反復関数による方法

２次元の場合r種類の反復関数は以下の行列計算で表される。

$$\begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} a_{11}^{\alpha} & a_{12}^{\alpha} \\ a_{21}^{\alpha} & a_{22}^{\alpha} \end{pmatrix} \begin{pmatrix} x_{n-1} \\ y_{n-1} \end{pmatrix} + \begin{pmatrix} b_1^{\alpha} \\ b_2^{\alpha} \end{pmatrix} \quad (\alpha = 1,2,\cdots,r)$$

これは複素平面上における以下の演算と同等であり、古くからその性質が調べられてきた。

$$z_n = a^{\alpha} z_{n-1} + b^{\alpha} \bar{z}_{n-1} + c^{\alpha} \quad (\alpha = 1,2,\cdots,r)$$

ここに

$$z_n = x_n + iy_n, \quad z_{n-1} = x_{n-1} + iy_{n-1}, \quad \bar{z}_{n-1} = x_{n-1} + iy_{n-1},$$

$$a^{\alpha} = (a_{11}^{\alpha} + a_{22}^{\alpha})/2 + i\,(a_{21}^{\alpha} - a_{12}^{\alpha})/2, \quad b^{\alpha} = (a_{11}^{\alpha} - a_{22}^{\alpha})/2 + i\,(a_{12}^{\alpha} + a_{21}^{\alpha})/2,$$

$$c^{\alpha} = b_1^{\alpha} + ib_2^{\alpha}.$$

これを３次元に拡張すると、反復関数は以下のように表される。

$$\begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} = \begin{pmatrix} a_{11}^{\alpha} & a_{12}^{\alpha} & a_{13}^{\alpha} \\ a_{21}^{\alpha} & a_{22}^{\alpha} & a_{23}^{\alpha} \\ a_{31}^{\alpha} & a_{32}^{\alpha} & a_{33}^{\alpha} \end{pmatrix} \begin{pmatrix} x_{n-1} \\ y_{n-1} \\ z_{n-1} \end{pmatrix} + \begin{pmatrix} b_1^{\alpha} \\ b_2^{\alpha} \\ b_3^{\alpha} \end{pmatrix} \quad (\alpha = 1,2,\cdots,r)$$

この反復関数から、確率的に１つ選んで計算を実行し、それを繰り返して点を打って行く。

以下具体的に例を示しながら結果を見て行こう。図 3.1.1 に非常に有名な C 曲線のデータと実行例を示す。実行例は 10000 点を打ったものである。描画結果は紙面（スクリーン面）上方向が z 軸正の方向、紙面右方向が x 軸正の方向である。

	確率	a1	a2	a3	a0
▶ 初期値		1	0	1	
反復関数1	0.5	0.5	0	-0.5	0
		0	0	0	0
		0.5	0	0.5	0
反復関数2	0.5	0.5	0	0.5	0.5
		0	0	0	0
		-0.5	0	0.5	0.5

図 3.1.1a C 曲線データ

図 3.1.1b 実行結果

データでは、1 行目に初期値として z_0 の値、1 列目に反復関数を選択する確率、2 行 2 列目以降が反復関数の係数行列の値である。同様にして、コッホ曲線の例を図 3.1.2 に示す。

	確率	a1	a2	a3	a0
▶ 初期値		1	0	0	
反復関数1	0.5	1/2	0	3^0.5/6	0
		0	0	0	0
		3^0.5/6	0	-1/2	0
反復関数2	0.5	1/2	0	-3^0.5/6	1/2
		0	0	0	0
		-3^0.5/6	0	-1/2	3^0.5/6

図 3.1.2a コッホ曲線データ

図 3.1.2b 実行結果

図 3.1.2 ではデータに数式を用いている。以上 2 つは 2 次元の例であったが、図 3.1.3 に 3 次元の例を示す。

	確率	a1	a2	a3	a0
▶ 初期値		1	1	1	
反復関数1	0.25	0.5	0	0	0
		0	0.5	0	0
		0	0	0.5	0
反復関数2	0.25	0.5	0	0	60
		0	0.5	0	0
		0	0	0.5	0
反復関数3	0.25	0.5	0	0	30
		0	0.5	0	60
		0	0	0.5	0
反復関数4	0.25	0.5	0	0	30
		0	0.5	0	30
		0	0	0.5	60

図 3.1.3a 3 次元ギャスケットデータ

図 3.1.3b 実行結果

これらの他にもいくつかの例をあげておこう。

	確率	a1	a2	a3	a0
初期値		1	1	1	
反復関数1	0.5	0.5	0.5	0	0.125
		-0.5	0.5	0	0.625
		0	0	0	0
反復関数2	0.5	0.5	0.5	0	-0.125
▶		-0.5	0.5	0	0.375
		0	0	0	0

図 3.1.4a ドラゴン曲線データ

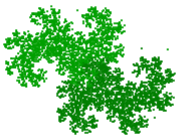


図 3.1.4b 実行結果

	確率	a1	a2	a3	a0
▶ 初期値		0.5	0.5	0	
反復関数1	0.5	0.8	0	0	0.1
		0	0.8	0	0.04
		0	0	0	0
反復関数2	0.168	0.5	0	0	0.25
		0	0.5	0	0.4
		0	0	0	0
反復関数3	0.166	0.355	-0.355	0	0.266
		0.355	0.355	0	0.078
		0	0	0	0
反復関数4	0.166	0.355	0.355	0	0.378
		-0.355	0.355	0	0.434
		0	0	0	0

図 3.1.5a 楓データ

図 3.1.5b 実行結果

	確率	a1	a2	a3	a0
▶ 初期値		0	0	0	
反復関数1	0.85	0.85	0.04	0	0
		-0.04	0.85	0	1.6
		0	0	0	0
反復関数2	0.07	-0.15	0.28	0	0
		0.26	0.24	0	0.44
		0	0	0	0
反復関数3	0.07	0.2	-0.26	0	0
		0.23	0.22	0	1.6
		0	0	0	0
反復関数4	0.01	0	0	0	0
		0	0.16	0	0
		0	0	0	0

図 3.1.6a 羊歯データ

図 3.1.6b 実行結果

	確率	a1	a2	a3	a0
▶ 初期値		1	1	1	
反復関数1	0.05	0.333333333	0	0	0
		0	0.333333333	0	0
		0	0	0.333333333	0
反復関数2	0.05	0.333333333	0	0	1
		0	0.333333333	0	0
		0	0	0.333333333	0
反復関数3	0.05	0.333333333	0	0	2
		0	0.333333333	0	0
		0	0	0.333333333	0
反復関数4	0.05	0.333333333	0	0	0
		0	0.333333333	0	1
		0	0	0.333333333	0
反復関数5	0.05	0.333333333	0	0	0
		0	0.333333333	0	2
		0	0	0.333333333	0
反復関数6	0.05	0.333333333	0	0	1
		0	0.333333333	0	2
		0	0	0.333333333	0
反復関数7	0.05	0.333333333	0	0	2
		0	0.333333333	0	1
		0	0	0.333333333	0
反復関数8	0.05	0.333333333	0	0	2
		0	0.333333333	0	2
		0	0	0.333333333	0
反復関数9	0.05	0.333333333	0	0	0
		0	0.333333333	0	0
		0	0	0.333333333	1
反復関数10	0.05	0.333333333	0	0	2
		0	0.333333333	0	0
		0	0	0.333333333	1
反復関数11	0.05	0.333333333	0	0	0
		0	0.333333333	0	2
		0	0	0.333333333	1
反復関数12	0.05	0.333333333	0	0	2
		0	0.333333333	0	2
		0	0	0.333333333	1
反復関数13	0.05	0.333333333	0	0	0
		0	0.333333333	0	0
		0	0	0.333333333	2
反復関数14	0.05	0.333333333	0	0	1
		0	0.333333333	0	0
		0	0	0.333333333	2
反復関数15	0.05	0.333333333	0	0	2
		0	0.333333333	0	0
		0	0	0.333333333	2
反復関数16	0.05	0.333333333	0	0	0
		0	0.333333333	0	1
		0	0	0.333333333	2
反復関数17	0.05	0.333333333	0	0	0
		0	0.333333333	0	2
		0	0	0.333333333	2
反復関数18	0.05	0.333333333	0	0	1
		0	0.333333333	0	2
		0	0	0.333333333	2
反復関数19	0.05	0.333333333	0	0	2
		0	0.333333333	0	1
		0	0	0.333333333	2
反復関数20	0.05	0.333333333	0	0	2
		0	0.333333333	0	2
		0	0	0.333333333	2

図 3.1.7a 3 次元メンガーデータ

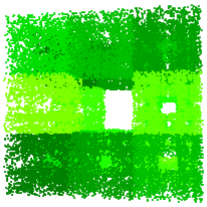


図 3.1.7b 実行結果

3.2 再帰的方法

再帰的方法は 2 章でも述べたが、2 次元でタートルは平面上を移動するので左右に向きを変えるとき turn 命令を用いるが、3 次元ではタートルの背中方向（初期値は y 軸負の方向）に法線ベクトルを考え、この向きを rotate 命令で変更する。ここで反復関数のときと同様に紙面（スクリーン面）上方向が z 軸正の方向、紙面右方向が x 軸正の方向である。しかし考えにくい場合は、タートルを 90° ひねり、背中を z 軸方向に向け、法線ベクトルの方向をタートルの右脇方向と考え、利用者はタートルに乗った状態をイメージすればよい。これで、rotate はタートルの回転（右ねじ方向）、turn は上下方向への向き変えとなる。以後この考え方に基づいて解説する。

この方法を用いた 3 次元フラクタルの例を図 3.2.1 に示す。

```
go 0.4
separate
turn -30
frac 0.6
return
rotate 60
turn 30
frac 0.6
return
rotate -60
turn 30
frac 0.6
return
```

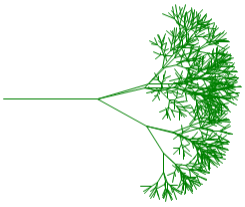


図 3.2.1a サンプル 1　　図 3.2.1b 次数 1　　図 3.2.1c 次数 6

初期状態で進行方向は右横（x 軸方向 0 から 1 へ）で、タートルの右脇に当たる法線ベクトルは、y 軸方向－である。紙面は x-z 平面で、タートルを右横から見ている状態になる。go 0.4 は 0.4 だけ進んで線を描く命令である。separate はその状態を記憶する。turn -30 は 30° 下を向きで、frac 0.6 で、0.6 倍に縮小したフラクタルを貼り付ける。return で記憶した位置に戻り、rotate 60 で 60° 右ねじの方向に回転、turn 30 で 30° 上を向いて、0.6 倍に縮小したフラクタルを貼り付ける、等々である。

次に図 3.2.2 にフラクタルを貼り付ける際に直線を表示しない fracp を用いた例を示す。

```
separate
rotate 90
go 1
turn 90
go 1
turn 90
go 1
turn 90
go 1
return
jump 0.5
turn -90
jump 0.25
turn 90
rotate 90
turn 45
rotate -90
fracp 0.7071
```

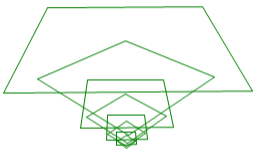


図 3.2.2a サンプル 1　　図 3.2.2b 次数 1　　図 3.2.2c 次数 8

今回のサンプルは真横からだと見にくいので、少し傾けて表示してある。最初に rotate 90 することによって、描画の方向が x-y 平面上になり、正方形を描いている。始点まで戻ったら、辺の中央にジャンプし、90° 傾けて 0.25 ジャンプする。元に戻して 45° 傾けて 0.7071 倍のフラクタルを表示せずに貼り付ける。

以上いくつかサンプルを示したが、マクロの簡単なまとめを表 3.1 に示す。

表 3.1 フラクタルビューア３Ｄの再帰処理書式

CONNECT		フラクタル同士を線でつなぐ　【宣言】　例：ヒルベルト曲線等
FRAC	比率	最後の次数だけ直線描画の再帰処理（標準/FRACF）
FRACP	比率	直線描画をしない再帰処理
FRACT	比率	常に直線描画する再帰処理
GO	比率	常に直線描画（標準/GOT）
GOF	比率	最後の次数だけの直線描画
TURN	角度	進行方向の角度変化[度]）
ROTATE	角度	進行方向の回転[度](/ROUND)
FRACC		強制的な終点へ連結する再帰処理
GOC		強制的な終点へ連結する直線描画
SEPARATE		分岐の開始点・状態の保存（ネスト構造はまだ未対応）
RETURN		分岐への状態の戻り
RESET		進行方向・回転角の初期値再設定

４．考察

我々のフラクタルビューアでは、２次元は通常のビットマップ画像として、３次元は３Ｄビューアを用いた空間データとして出力した。当然２次元も平面データとして出力することができるが、画面を動かしてもあまり効果的でないこと、ドットの数を増やしてより細かい絵を描きたかったことからビットマップへの出力を選択した。３次元では動きを重視するために、描画要素数 10000 程度までが望ましい。これでは画面を覆うような図は描ききれない。

我々は、２次元のフラクタル描画手法として、複素数の収束による方法と再帰的方法、３次元の描画手法として反復関数による方法と再帰的方法を提供してきたが、反復関数による方法は２次元でも利用可能であるので、これを加えることも考えたい。ただその際には、３次元ではあまり考えていなかった色を重視した追加にすべきであろう。またその他にもフラクタルの表示方法はいくつかあり、これらについても検討したい。しかし、我々のプログラムの目的はあくまでフラクタル描画原理の学習支援であるので、機能強化を重ねプログラムを複雑化することには疑問がある。適度なバランスが重要であると考える。

フラクタルビューア２Ｄのマクロに比べ、フラクタルビューア３Ｄのマクロは単純である。描画も直線を集めたもので、面の概念もない。そのため四角形１つにも 10 行近くのプログラムが必要である。我々は、単純な多角形を描画要素に加えたいと考えるが、面にはその向きのデータが必要となり、拡張の仕様が固まっていない。現在はフラクタルビューア２Ｄに準拠して、法線ベクトルに垂直な面での描画を考えているが、全体的にもう少し簡単にしなければ使いづらい。これは今後の課題である。

我々のプログラムはフラクタルアートを作成するようなものではなく、基本的に原理を学ぶためのツールである。それゆえ作る画像は、完全な自己相似形であり、それらを組み合わせて表示することや視覚的な効果を加えることは殆ど考えていない。今後の課題として残したいが、プログラムの趣旨とは外れてくる。この先 College Analysis 本体から分離独立させることを考えてもよい。

参考文献

１）社会システム分析のための統合化プログラム 16 －３Ｄビューアとその応用－, 福井正康, 尾崎誠, 石丸敬二, 福山平成大学経営学部経営研究, 7 号, (2011) 掲載予定.

２）College Analysis における 2 変量関数表示プログラム, 石丸敬二, 福井正康, 福山大学経済学部経済学論集, 35 巻, 2 号, (2010) 掲載予定.

Multi-purpose Program for Social System Analysis 17

- Fractal Viewer -

Makoto OZAKI, Keiji ISHIMARU* and Masayasu FUKUI

Department of Business Administration, Faculty of Business Administration,
Fukuyama Heisei University
* Department of Economics, Faculty of Economics,
Fukuyama University

Abstract

We have been constructing a unified program on the social system analysis for the purpose of education. We developed some programs using 3D display, because we had been interesting in CG. This time, we introduce the program of fractal viewer. This program can display the fractal of 2D and 3D. In addition, it has the function to display the Mandelbrot set.

Keywords

College Analysis, social system analysis, statistics, management science, decision making, fractal, 3D

URL: http://www.heisei-u.ac.jp/ba/fukui/