Chapter 14

力学系（補講１）

「力学系」とは「時間発展する世界」のごく単純なモデルである．本章では以下の 3 項目について解説する．

- グラフ解析と呼ばれる方法で実 1 次元力学系を可視化．
- 方程式の近似解法であるニュートン法を可視化．
- フラクタル図形として有名な複素 1 次元力学系のジュリア集合・マンデルブロー集合の描画．

注意． 本章は「補講」，すなわちオマケです．13 章までの内容は既知とさせていただきます．説明も親切さに欠ける部分があるかもしれませんがご了承ください．

14.1 力学系とは？

まずは**力学系** (dynamical system) とは何か，具体例をみながら説明してみよう．次のような漸化式を考える：

$$a_{n+1} = 2a_n;\ a_0 = 1.$$

この漸化式を解くのは簡単だが（$a_n = 2^n$），ちょっと遠回りをして次のように解釈してみる．いま $f(x) = 2x$ とおくと，

$$a_{n+1} = f(a_n)$$

と表されるから，$a_1 = f(a_0)$, $a_2 = f(f(a_0))$，$a_3 = f(f(f(a_0)))$，という具合に関数 f を繰り返し合成（反復合成）することで数列が生成されていることがわかる．初項 a_0 はたまたま 1 としているが，この値を変化させても数列を生成する仕組み自体は変化しないのである．

ところで，ニュートン力学のような素朴な世界観では，世界（宇宙）を構成する要素として「空間」と「時間」，そして物体の運動を記述する「運動法則」（物理法則）の 3 つがある．[*1]空間内に質量をもった複数の物体が配置されて，時間とともに全体の「系」が変化するのである．

同様に，上の数列 $\{a_n\}$ が棲む「世界」は次のように記述される：

- 空間：数直線 $\mathbb{R}$
- 時間：$0, \pm 1, \pm 2, \cdots$（秒）
- 運動法則：x の位置にある動点は 1 秒後に $f(x) = 2x$ の位置に移動する

この「世界」を「$\mathbb{R}$ 上の f による**力学系** (dynamical system)」と呼ぶ．[*2]また，点 $x_0 \in \mathbb{R}$ にたいし

$$x_0 \overset{f}{\longmapsto} f(x_0) \overset{f}{\longmapsto} f(f(x_0)) \overset{f}{\longmapsto} \cdots$$

（すなわち $x_0 \overset{f}{\longmapsto} 2x_0 \overset{f}{\longmapsto} 2^2 x_0 \overset{f}{\longmapsto} \cdots$ ）で定まる数列を初期値 x_0 の**軌道** (orbit) と呼ぶ．たとえば数列 $\{a_n\}$ は初期値 $a_0 = 1$ の軌道だと考えられる．このようにして得られる軌道の振る舞いを調べるのが，力学系理論の目標である．

また，$f \circ f \circ f \circ \cdots \circ f$ と f を n 回合成したものを f^n と書くことにすれば，上の軌道は $\{f^n(x_0)\}$ と表されるから，[*3]，力学系理論では本質的に「関数 f の反復合成によって得られる数列の振る舞いを記述すること」が目標となる．しかし，一般にはこれが至って難しい．たとえば $f(x)$ が 2 次関数の場合，$f^{10}(x)$ は x の 1024 次関数である．手計算で力学系の時間発展を追いかけ続けるのは至難の業，実験もままならない—— というのはあくまで昔の話．いまは便利なパソコンと，*Mathematica* があるではないか．

14.2 力学系のグラフ解析

与えられた実数 p と関数 $y = f(x)$ に対し，軌道

$$p \ \mapsto \ f(p) \ \mapsto \ f(f(p)) \ \mapsto \ f(f(f(p))) \ \mapsto \ \cdots$$

を視覚化する非常によい方法がある．$y = f(x)$ のグラフを用いて，xy 平面上の点 (p, p) から $(f(p), f(p))$ を作図する方法である：

[*1] 相対性理論によれば，これらの 3 要素を明確に区別することはできない．

[*2] より正確には離散力学系 (discrete dynamical system) と呼ばれるものである．時間が「ぱらぱら漫画」のように離散的だからである．歴史的にはポアンカレが微分方程式系で記述される（連続時間をもつ）力学系の理論を創始した．そこでは離散力学系が重要な「道具」として用いられる．

[*3] $f^n(x)$ は $f \circ f \circ f \circ \cdots \circ f(x)$ のことで $f(x)$ の n 乗 $\{f(x)\}^n$ ではない．

Step 0 $y = f(x)$ と $y = x$ のグラフを描く．
Step 1 点 (p, p) からタテ方向に，$y = f(x)$ のグラフまで線分を引く．その交点は $(p, f(p))$ である．
Step 2 $(p, f(p))$ からヨコ方向に，$y = x$ のグラフとまで線分を引く．その交点が $(f(p), f(p))$ である．

この Step 1 と 2 を続けていき，直線 $y = x$ を数直線だと思えば点 p の軌道が視覚化される（図 14.1）．この手続きを**グラフ解析** (graphical analysis) と呼ぶ．また，グラフ解析で得られる図を**ウェブ・ダイアグラム**（web diagram）と呼ぶ．

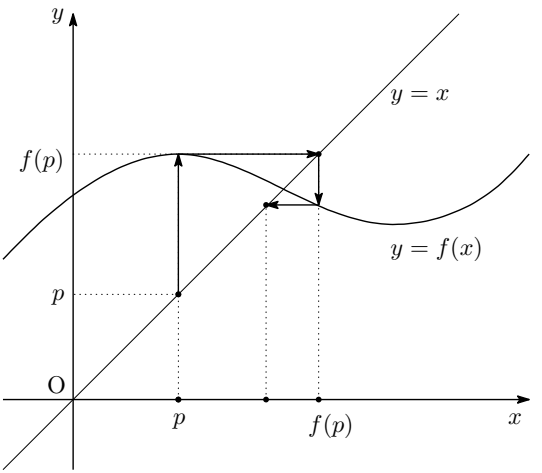


図 14.1　$f(x)$ のグラフ解析．(p, p) から $(f^2(p).f^2(p))$ までを作図したところ．

たとえば $f(x) = 2x$ で初期値 $x_0 = \pm 1/4$ とした場合は図 14.2 のようになる．

これを *Mathematica* に作図させる方法を紹介しよう．基本的にはウェブ・ダイアグラムの折れ線を `ListLinePlot` で描画する．これに `Show` で関数のグラフを重ね合わせるのである．

[1] Step 0 のように，関数 $y = f(x) = 2x$ と $y = x$ のグラフを描く：

```
In[1]:= f[x_] := 2 x;
        gr = Plot[{f[x], x}, {x, -2, 8}, AspectRatio -> Automatic]
```

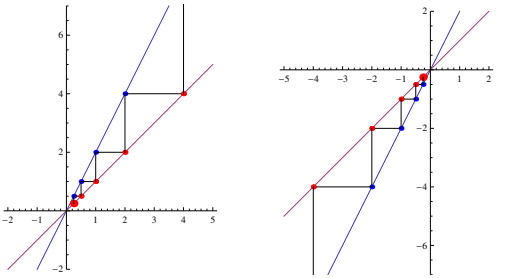
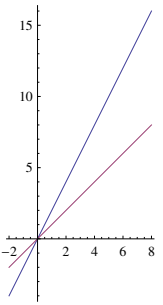


図 14.2 $f(x) = 2x$ のグラフ解析．左は $x_0 = 1/4$ のとき，右は $x_0 = -1/4$ のとき．

Out[1]=



[2] Step 1 と Step 2 に対応する部分．点 (p, p) からタテに移動して $(p, f(p))$ へ，さらにヨコに移動して $(f(p), f(p))$ へ，という「1 操作分」の折れ線を追加するための関数を定義：

In[2]:=
```
tateyoko[p_] := {{p, f[p]}, {f[p], f[p]}};
```

[3] 初期値 p に対して，上記の「1 操作分」を n 回繰り返したリストを作る関数を定義：

In[3]:=
```
weblist[p_, n_] := (w = {{p, p}}; x = p;
    Do[(w = Join[w, tateyoko[x]]; x = f[x]), {i, 1, n}];
    w);
```

[4] たとえば…：

In[4]:=
```
weblist[1/2, 3]
```

Out[4]=
$$\left\{\left\{\frac{1}{2}, \frac{1}{2}\right\}, \left\{\frac{1}{2}, 1\right\}, \{1, 1\}, \{1, 2\}, \{2, 2\}, \{2, 4\}, \{4, 4\}\right\}$$

[5] ウェブ・ダイアグラムを作る関数：

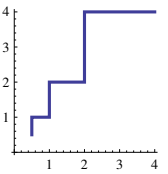
```
In[5]:= webdiag[p_, n_] := ListLinePlot[weblist[p, n],
                    PlotStyle -> Thick, AspectRatio -> Automatic,
                    PlotRange -> All];
```

最後の PlotRange -> All はおまじないのようなもので，ないとうまく描画されないことがある．

[6] たとえば…：

```
In[6]:= webdiag[1/2, 3]
```

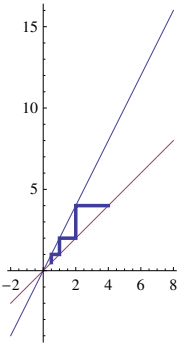
Out[6]=



[7] Show を用いて最初のグラフと重ねて表示：

```
In[7]:= Show[gr, webdiag[1/2, 3]]
```

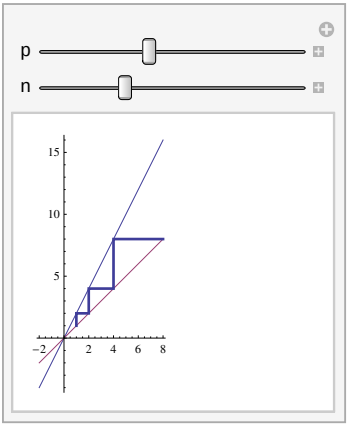
Out[7]=



[8] Manipulate で p と n を変化させる：

```
In[8]:= Manipulate[Show[gr, webdiag[p, n]],
                    {{p, 1}, -1, 4}, {{n, 3}, 0, 10, 1}]
```

Out[8]=



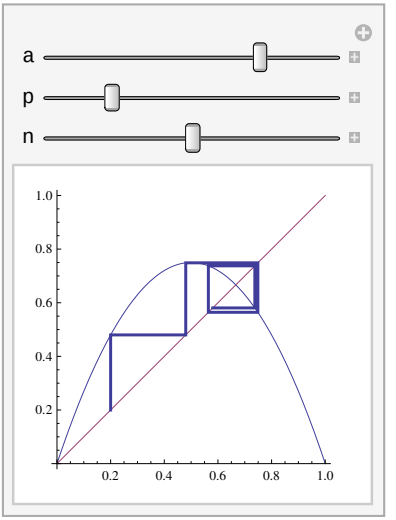
n は整数のみを動くように指定していることに注意.

問題 14.1 (ロジスティック写像) 2 次関数 $g_a(x) = ax(1-x)$ は $0 \leq a \leq 4$ のとき区間 $[0,1]$ から 区間 $[0,1]$ への関数となる．そのウェブ・ダイアグラムを Manipulate を用いて可視化せよ．ただし，a もパラメーターとして変化できるようにすること.

【解答】 基本的に [1] – [8] と同じである．パラメーター a が新たに加わる部分だけ注意すればよい.

In[ ]:=
```
g[a_, x_] := a x (1 - x);
gr2[a_] := Plot[{g[a, x], x}, {x, 0, 1},
                  AspectRatio -> Automatic]
tateyoko2[a_, p_] := {{p, g[a, p]}, {g[a, p], g[a, p]}};
weblist2[a_, p_, n_] := (w = {{p, p}}; x = p;
        Do[(w = Join[w, tateyoko2[a, x]]; x = g[a, x]),
            {i, 1, n}]; w);
webgr2[a_, p_, n_] := ListLinePlot[weblist2[a, p, n],
                  PlotStyle -> Thick, PlotRange -> All]
Manipulate[Show[gr2[a], webgr2[a, p, n]],
 {{a, 3}, 0, 4}, {{p, 0.2}, 0, 1}, {{n, 5}, 0, 10, 1}]
```

以上のように入力すると，次のような結果が得られる：



■

問題 14.2 (より見やすく) 問題 1 のウェブ・ダイアグラムを次のように変えて「見やすさ」を追求せよ：

(1) ダイアグラムの線分は黒に．
(2) ダイアグラムで，$y = x$ 上の点は赤く，グラフ上の点は青く．

【解答】 ListLinePlot のオプションではうまくいかないので，最初から折れ線を Graphics で作ってしまおう．Graphics の使い方の詳細はドキュメントセンターを参照されたい．

まずはタテヨコの黒い線分と赤の点，青の点を作成するための関数を定義：

```
In[ ]:= tateyoko3[p_, q_] := (pp = {p, p}; pq = {p, q}; qq = {q, q};
          {Line[{pp, pq, qq}],
           {Red, PointSize[Medium], Point[{pp, qq}]},
           {Blue, PointSize[Medium], Point[pq]}
           });
```

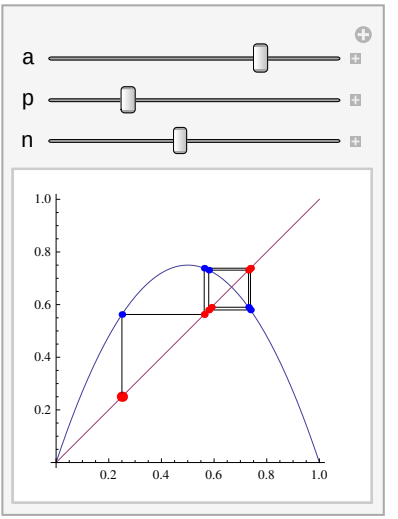
基本的には q に g[a, p] を入れるのだが，汎用性と式の見やすさを考慮してこのようにした．次に関数 tateyoko3 を n 回施す関数を定義：

```
In[ ]:= weblist3[a_, p_, n_] := (
          w = {{Red, PointSize[Large], Point[{p, p}]}}; x = p;
           Do[(w = Join[w, tateyoko3[x, g[a, x] ]]; x = g[a, x]),
              {i, 1, n}]; w);
```

Manipulate で描画：

```
In[ ]:= Manipulate[ Show[gr2[a], Graphics[weblist3[a, p, n]]],
        {{a, 3}, 0, 4}, {{p, 0.25}, 0, 1}, {{n, 5}, 1, 10, 1}]
```

以上のように入力すると，次のような結果が得られる：



■

14.3 ニュートン法

与えられた関数 $f(x)$ に対し方程式 $f(x) = 0$ の解を数値的に求める方法として，**ニュートン法** (Newton's method) というものがある．幾何学的（直感的）には，次のような手続きによって解の近似値を得るのである：

(1) 関数 $y = f(x)$ のだいたいのグラフを描き，解の場所に見当をつける．
(2) 解の近くにあると思われる値 x_0 を選び，グラフ上の点 $(x_0, f(x_0))$ における接線を引く．
(3) 接線と x 軸の交わる点を $(x_1, 0)$ とする．

もし x_0 が方程式の解 $f(x)=0$ の解 α に十分に近ければ，x_1 は解 α の「さらに良い近似値」を与えることが証明できる．[*4]さらに $(x_1, f(x_1))$ からグラフの接線を引いて同様の操作を繰り返せば，近似はもっと良くなるであろう．簡単な計算により $x_1 = x_0 - f(x_0)/f'(x_0)$ であることがわかるから，

$$N_f(x) := x - \frac{f(x)}{f'(x)}$$

とおいて得られる（関数 N_f によって得られる力学系の）軌道

$$x_0 \overset{N_f}{\longmapsto} x_1 = N_f(x_0) \overset{N_f}{\longmapsto} x_2 = N_f^2(x_0) \overset{f}{\longmapsto} \cdots$$

は解 α への収束列を与えると期待される．この関数 N_f を f の**ニュートン写像**という．この節では，ニュートン法で解が近似される様子を数値的に観察し，それをグラフ上で可視化する方法を紹介しよう．

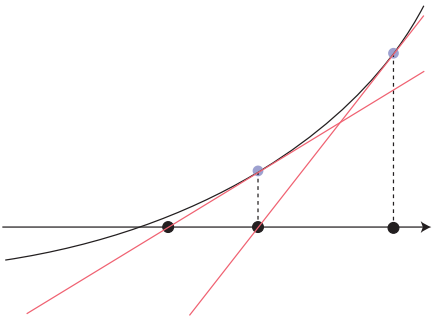


図 14.3　ニュートン法

[9] $f(x) = x^2 - 2$ とすれば，ニュートン法で得られる列は $\sqrt{2}$ の近似列を与えるはずである．その様子を見てみよう．まずは関数 f を上書きし，さらにニュートン写像を定義：

```
In[9]:=  f[x_]  = x^2 - 2;
         df[x_] = D[f[x], x];
         newton[x_] = x - f[x]/df[x]
```

[*4] ここで，関数 f は最低でも接線を引ける程度に滑らかでなくてはならない．たとえば C^2 程度を仮定すると十分である．

```
Out[9]=   x - (-2 + x^2)/(2 x)
```

すなわち $N_f(x) = \dfrac{x^2+2}{2x}$ であることがわかる．

[10] NestList を用いて，たとえば $x_0 = 1$ から始めたときの $x_0, x_1, \ldots, x_5$ をリストにする．：

```
In[10]:= app = NestList[newton, 1, 5]
Out[10]= {1, 3/2, 17/12, 577/408, 665857/470832, 886731088897/627013566048}
```

[11] これでは近似の様子が見えてこないので，実用上はあまり意味がない．有効数字 20 桁で数値化する：

```
In[11]:= N[app, 20] //TableForm
```

```
Out[11]//TableForm=
1.0000000000000000000
1.5000000000000000000
1.4166666666666666667
1.4142156862745098039
1.4142135623746899106
1.4142135623730950488
```

TableForm を用いて縦方向に並べた．

[12] 真の値 Sqrt[2] との誤差を見てみよう[*5]：

```
In[12]:= N[{app, app - Sqrt[2]}, 20] //Transpose//TableForm
```

```
Out[12]//TableForm=
1.0000000000000000000   -0.41421356237309504880
1.5000000000000000000   0.085786437626904951198
1.4166666666666666667   0.0024531042935716178650
1.4142156862745098039   2.1239014147551198799 × 10^-6
1.4142135623746899106   1.5948618246068546804 × 10^-12
1.4142135623730950488   8.9929283216504531005 × 10^-25
```

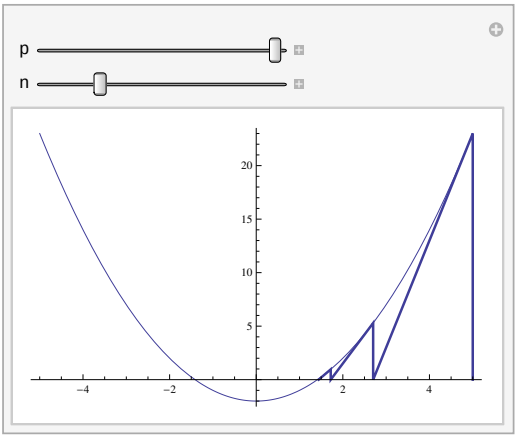
右側に真の値との誤差が並んでいる．誤差は「急速に」小さくなっていることが分かるだろう．大雑把に言って，１ステップごとに真の値と合致する小数点以下の桁数が倍近くになる．方程式の近似解法の中でも，ニュートン法はたいへん効率がよい．

[*5] この式はちょっと技巧的かもしれない．これは
Transpose[N[{app, app - Sqrt[2]}, 20]] //TableForm と同じ意味であり，一旦転置を取ってから表にすることで誤差が右側に並ぶようにしたのである．次のように書いても同様の表示が得られる：
TableForm[N[{app, app - Sqrt[2]}, 20], TableDirections -> Row]

[13] 次に図示してみよう．基本的にはウェブ・ダイアグラムを描く要領でやればよい：

In[13]:=
```
seg[p_] := {{p, f[p]}, {newton[p], 0}};
seglist[p_, n_] := (w = {{p, 0}}; x = p;
   Do[ (w = Join[w, seg[x]]; x = newton[x]), {i, 1, n}]; w);
seggr[p_, n_] := ListLinePlot[seglist[p, n],
                    PlotRange -> All, PlotStyle -> Thick]
gr = Plot[f[x], {x, -5, 5}];
Manipulate[ Show[gr, seggr[p, n]],
 {{p, 5}, -5, 5}, {{n, 3}, 1, 10, 1}]
```

Out[13]=

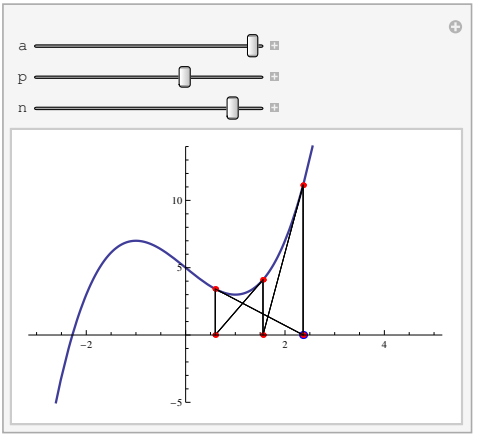


p が負のときは $-\sqrt{2}$ に収束し，p が正のときは $\sqrt{2}$ に収束することがわかる．[*6]

解答は省略するが，問題 1 を参考にぜひ次の問題 3 も考えてみていただきたい．

問題 14.3 (解に収束しない例) 3 次関数 $g_a(x) = x^3 - 3x + a$ $(a > 0)$ に対して，ニュートン法の軌道を Manipulate を用いて可視化せよ．ただし，a もパラメーターとして変化できるようにすること．また a の値によっては下の図のような軌道が現れて，必ずしも方程式 $g_a(x) = 0$ の解に収束するとは限らない（と思われる）ことを確認せよ．

[*6] ちなみに，p がちょうど 0 の場合は f の微分がゼロになるので数列が定義できなくなる．実際は軌道が「無限遠点」という固定点に乗ってしまう．



14.4 複素力学系とジュリア集合

今度は「複素平面 $\mathbb{C}$ 上の，関数 $f(z)$ による力学系」を考える．いわゆる（1 次元）**複素力学系** (complex dynamics) と呼ばれるものである．ここでは関数 $f(z)$ を $f_c(z) = z^2 + c$ $(c \in \mathbb{C})$ の形の 2 次多項式に制限して，その力学系における軌道のふるまい理解したいとしよう．ここで f_c の定数項 $c \in \mathbb{C}$ は複素数のパラメーターであり，力学系における物理定数のようなものだと考えたらよいかもしれない．このパラメーターを変化させることで，「世界」の様相は意外なほど変化するのである．

無限の鉢と充填ジュリア集合． f_c の力学系に共通の性質として，

『$|z|$ がある程度大きいとき，$|f_c^n(z)| \to \infty$ $(n \to \infty)$ が成り立つ』

ということが証明できる．[*7] したがって「軌道（の絶対値）が無限大に発散する初期値」の集合

$$B_c := \{z \in \mathbb{C} \mid |f_c^n(z)| \to \infty \ \ (n \to \infty)\}$$

の形をみれば，パラメーター c に依存する力学系の変化を捉えることができるかもしれない．この集合を f_c の**無限の鉢** (basin at infinity) と呼ぶ．また，B_c の補集合

$$K_c := \mathbb{C} - B_c$$

[*7] たとえば，$|z| \geq 2 + |c|$ のとき $|f_c(z)| \geq 2|z|$ が成り立つ．実際，$|f(z)| = |z^2 + c| \geq |z| \cdot |z| - |c| \geq (2 + |c|)|z| - |c| = 2|z| + |c|(|z| - 1) \geq 2|z| + |c|(1 + |c|) \geq 2|z|$．よって $|f^n(z)| \geq 2^n|z| \to \infty$ $(n \to \infty)$．もう少し精密に評価すると，$|z| \geq \max\{2, |c|\}$ のとき $|f^n(z)| \to \infty$ $(n \to \infty)$ であることが証明できる．

を**充填ジュリア集合** (filled Julia set) と呼ぶ．この集合は「軌道が有限の範囲にとどまるような初期値の集合」であることも知られている．

ちなみに B_c と K_c の境界部分は「発散する軌道」と「有界にとどまる軌道」がせめぎあう部分であり，わずかな誤差で前者にも後者にも変化しうる．すなわち，この力学系における「カオス部分」である．これを f_c の**ジュリア集合** (Julia set) と呼び，J_c で表す．面白いことに，一般に J_c は自己相似性をもつフラクタル集合となるのである．

では，*Mathematica* でこれらの集合を描く方法を考えてみよう．いろんな描き方があるが，ここでは計算効率は度外視して数学的な単純さ・分かりやすさを優先した次の方法を紹介する．[*8]

まず話を簡単にするために，$|c| \leq 2$ と仮定しておこう．このとき，与えられた $z \in \mathbb{C}$ に対し $|f_c^k(z)| \geq 2$ となる k が存在すれば，$|f_c^{k+n}(z)|$ は n に関して単調増加かつ発散することが証明できる．したがって，

$$B_c(k) := \left\{ z \in \mathbb{C} \mid |f_c^k(z)| \geq 2 \right\}$$

とすれば，$B_c(1) \subset B_c(2) \subset \cdots$ かつ $B_c = \bigcup_{k \geq 1} B_c(k)$ である．よって k が十分大きいとき $B_c(k)$ は B_c を近似していると考えてよいだろう．以下では $k = 50$ としている．

[14] まずは $c = -0.122 + 0.745i$ に対して B_c と K_c を描いてみよう．f の定義を上書きする．汎用性を考えて c の定義と別にしておく：

```
In[14]:= c = -0.122 + 0.745 I; f[z_] := z^2 + c;
```

[15] さて無限の鉢 B_c に色をつける関数を定義：

```
In[15]:= col[z_] := (p = z; k = 0;
                     While[(Abs[p] < 2.0) && (k < 50),
                           (p = f[p]; k = k + 1)];
                     k)
```

`While` の中の意味は「$|f_c^k(z)| < 2$ かつ $k < 50$ である限り，カッコ `(...)` 内のふた

[*8] 実際のところ，描画速度を優先するならば *Mathematica* を使うべきではない．C や Java のようなプログラミング言語を用いるべきである．複素力学系の計算に *Mathematica* を用いる利点は，
- 数値がデフォルトで複素数であり，計算が極めて容易
- グラフィクスやユーザー・インターフェイスが組込み関数で準備されている

という部分であって，C 言語などに比べて「プログラムを書きあげるまでの時間」が大幅に短縮できるのである．

つの命令 `p = f[p]; k = k + 1` を繰り返せ，という意味である．最終的には `k` が値として定まるが，

- $|f_c^k(z)| \geq 2$ となる $k \leq 49$ が見つかった．
- $k = 50$ になるまで $|f_c^k(z)| < 2$ であり続けた．

のいずれかである．すなわち，`col` は 0 以上 50 以下の整数値をとる関数であって，前者の場合（49 以下）は $z \in B_c$ とみなし，後者の場合（ちょうど 50）は $z \in K_c$ とみなすことにする．

[16] 複素平面の領域

$$\{x + yi \mid -2 \leq x \leq 2,\ -2 \leq y \leq 2\}$$

上を刻み幅 `d = 0.01` で分割し，`Table` を用いて関数 `col` の値をリスト（のリスト，すなわち行列と同じ形のデータ）にする：

```
In[16]:= d = 0.01;

         tab = Table[col[x + y I], {x, -2, 2, d}, {y, -2, 2, d}];
```

ここの計算が一番時間がかかる．数十秒は我慢しよう．[*9]

[17] `tab` の値を `ArrayPlot` で表現するために，少し修正した `complexAP` (complex Array Plot の略) を導入する（この意味は図 14.4 参照．）：

```
In[17]:= complexAP[t_] := ArrayPlot[Reverse[Transpose[t]]];
```

[*9] $400 \times 400 = 160000$ 個の複素数についてそれぞれ `col` を計算させることになる．パソコンに長時間向かって肩もこっていることだろうから，ストレッチでもしながらのんびり待つとよい．ちなみに `col` 関数の定義内の `50` を減らすか，刻み幅 `d` を増やせば計算は速くなる．そのかわり，得られる画像は粗くなる．

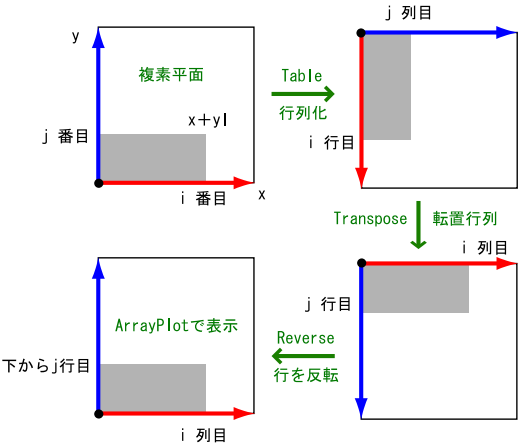
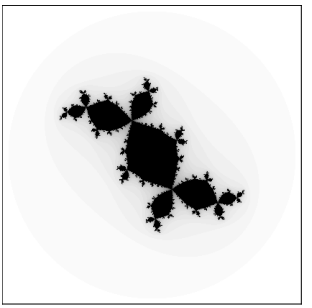


図 14.4　complexAP の意味．左上の複素平面を刻み幅 d でグリッドに分割して col 関数を適用するが，それを tab のように行列型のデータにすると右上のような配置になる．これに Transpose（行列の転置），Reverse（行の前後入れ替え）を施すことで，もとの複素平面のグリッドとデータの配置が一致する．これを ArrayPlot で表示するのである．

[18] tab を complexAP で図示する：

```
In[18]:= complexAP[tab]
```

Out[18]=



col の値が小さいほど白くなる．したがって白からグレーの部分が B_c であり，黒い部分が K_c である．このように意外と複雑なフラクタル集合となる．

問題 14.4 (より美しく？) 上の B_c を白黒でなく，適当に色づけせよ．

【解答】 ArrayPlot のオプションには ColorFunction というのがあるので，それで指定できる．（詳しくはドキュメントセンターを参照せよ．）たとえば，complexAP を修正して

```
In[ ]:=  complexAP2[t_] := ArrayPlot[Reverse[Transpose[t]],
                                     ColorFunction -> "LightTemperatureMap"];
```

としてみよう．これに [16] で作ったデータ tab を放り込んで complexAP2[tab] とすればよい．結果は図 14.5 の左上のようになる． ■

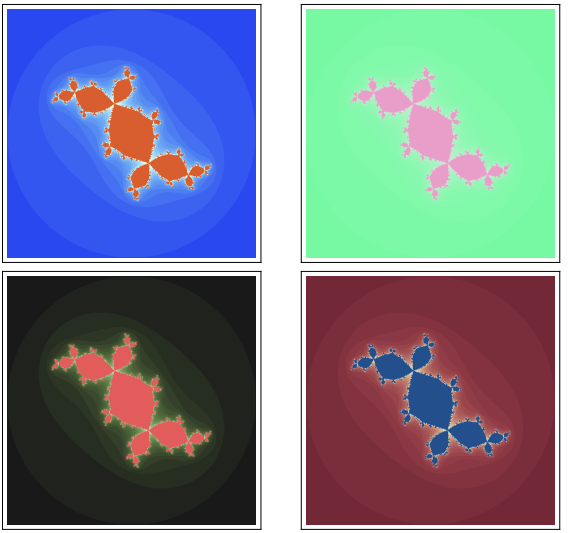


図 14.5 ColorFunction のオプションはドキュメントセンターの『カラースキーム』の項に詳しく述べられている．左上："LightTemperatureMap"，右上："MintColors"，左下："WatermelonColors"，右下："RedBlueTones"．

c の値をいろいろと変えても面白い．たとえば図 14.6 のようになる．

マンデルブロー集合． f_c の力学系において，$z=0$ は $f_c'(z)=0$ となる唯一の点である．そのような特殊性から，$z=0$ の軌道によって力学系の性質がある程度分類できることが知られている．たとえば「f_c の力学系において $z=0$ の軌道（の絶対値）が発散する（すなわち $0 \in B_c$ ）」という性質を持ったパラメーター c の集合

$$H_\infty := \{c \in \mathbb{C} \mid |f_c^n(0)| \to \infty \ \ (n \to \infty)\}$$

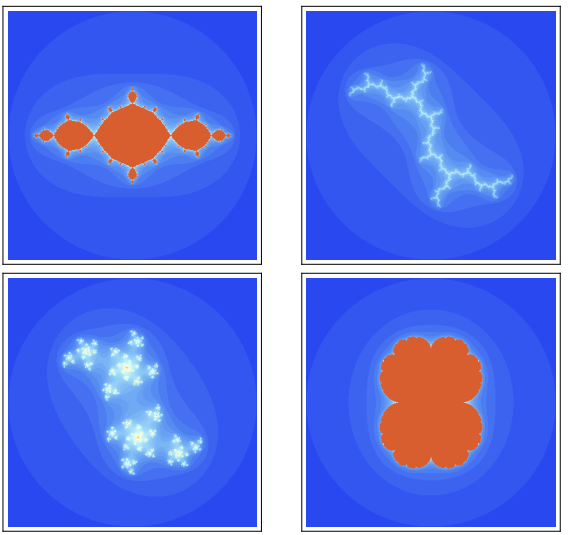


図 14.6　左上：$c=-1$，右上：$c=i$, 左下：$0.22+0.65i$，右下：$c=0.25$.

を考え，その補集合

$$\mathbb{M} := \mathbb{C} - H_\infty$$

を**マンデルブロー集合** (the Mandelbrot set) と呼ぶ.[*10]

問題 14.5（マンデルブロー集合） complexAP2 を用いて集合 $\mathbb{M}$ を描け.

【解答】　無限の鉢を描く方法を修正して，たとえば

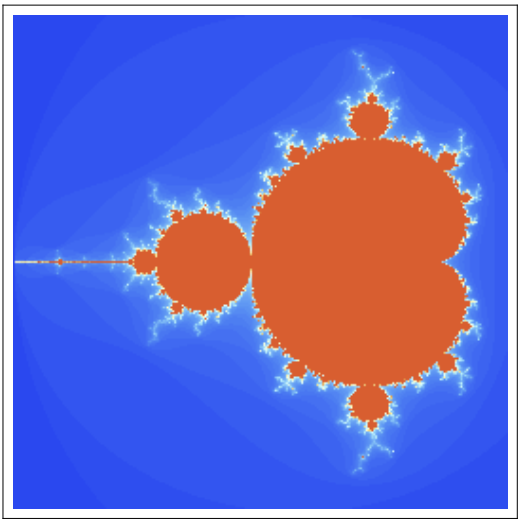
```
In[ ]:=  colM[c_] := (p = 0.0; k = 0;
                      While[(Abs[p] < 2.0) && (k < 100),
                      (p = p^2 + c; k = k + 1)];
                       k)
```

としてみよう．d = 0.01 はそのまま流用して，

```
In[ ]:=  tabM = Table[colM[a + b I],
                       {a, -2, 0.6, d}, {b, -1.3, 1.3, d}];
```

とおく．（これも実行に数十秒かかるだろう．）あとは得られたデータを上の問題で作った complexAP2[tabM] のようにして表示すればよい．結果として次のような図が得られる：

[*10] この集合は「K_c が連結（ひとつながり）になっているような c」と一致することが知られている.



■

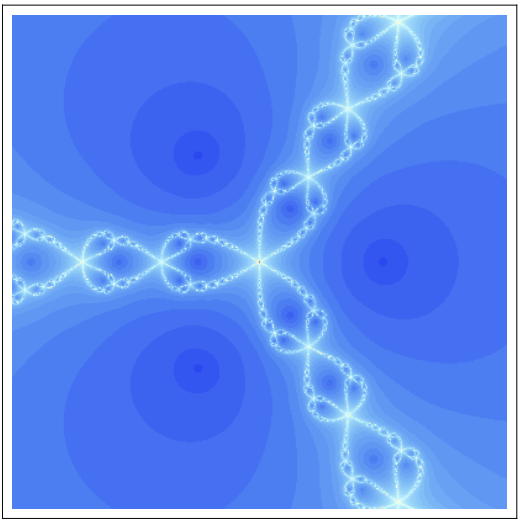
参考：ニュートン法再訪. 方程式 $f(z) = z^3 - 1 = 0$ について，複素数解もふくめて考えよう．じつはニュートン法は複素数でもそのまま応用できて，初期値 z_0 が方程式の解に十分近いとき，そのニュートン写像 $N_f(z)$ による軌道は解に近づくことが知られている．

上で 2 次多項式の「無限の鉢」を描かせたときと同様の考え方で，「軌道が 1 の 3 乗根に近づくかどうか」を判定するプログラムを描けば，有理関数 $N_f(z)$ の「ジュリア集合」（カオス部分）を描くことができる．

たとえばこの f の場合（f, df, newton など使用済みの文字は適宜クリアしておこう），

```
d = 0.01; f[x_] = x^3 - 1;
df[x_] = D[f[x], x];
newton[x_] := x - f[x]/df[x];
colN[z_] := (p = z; k = 0;
              While[(Abs[ f[p] ]> 0.1) && (k < 50),
              (p = newton[p]; k = k + 1)];
               k);
tabN = Table[colN[x + y I], {x, -2, 2, d}, {y, -2, 2, d}];
complexAP2[tabN]
```

としてみよう．次のような，意外と複雑なフラクタル図形が得られるはずである．



14.5　参考文献

離散力学系理論と複素力学系理論の入門書として次を挙げておく．

(1) R. Devaney.「カオス力学系入門（第 2 版）」．共立出版．
(2) 川平友規.「マンデルブロー集合 — 2 次多項式の複素力学系入門」．
http://www.math.titech.ac.jp/~kawahira/courses/mandel.pdf