第 5 章　分形与混沌图形

司守奎
烟台市，海军航空大学
Email：sishoukui@163.com
微信：sishoukui

分形（Fractal）这个术语是美籍法国数学家 Mandelbrot 于 1975 年创造的。Fractal 出自拉丁语 fractus（碎片，支离破碎）、英文 fractured（断裂）和 fractional（碎片，分数），说明分形是用来描述和处理粗糙、不规则对象的。Mandelbrot 是想用此词来描述自然界中传统欧几里得几何学所不能描述的一大类复杂无规则的几何对象，如蜿蜒曲折的海岸线、起伏不定的山脉、令人眼花缭乱的漫天繁星等。它们的共同特点是极不规则或极不光滑，但是却有一个重要的性质——自相似性，举例来说，海岸线的任意小部分都包含有与整体相似的细节。要定量地分析这样的图形，要借助分形维数这一概念。经典维数都是整数，而分形维数可以取分数。简单的来讲，具有分数维数的几何图形称为分形。

1975 年，Mandelbrot 出版了他的专著《分形对象：形、机遇与维数》，标志着分形理论的正式诞生。1982 年，随着他的名著《The Fractal Geometry of Nature》出版，分形这个概念被广泛传播，成为当时全球科学家们议论最为热烈、最感兴趣的热门话题之一。

分形具有以下几个特点：

（1）具有无限精细的结构。

（2）有某种自相似的形式，可能是近似的或是统计的。

（3）一般它的分形维数大于它的拓扑维数。

（4）可以由非常简单的方法定义，并由递归、迭代等产生。

混沌（Chaos）是指混乱或无序的状态，它是确定性动力学系统因对初始值敏感而表现出的不可预测的、类似随机性的运动。混动行为存在于许多自然系统中，如气象和气候。

5.1　Weierstrass 函数

德国数学家 Karl Theodor Wilhelm Weierstrass 于 1972 年构造了一个处处连续但处处不可导的函数，即

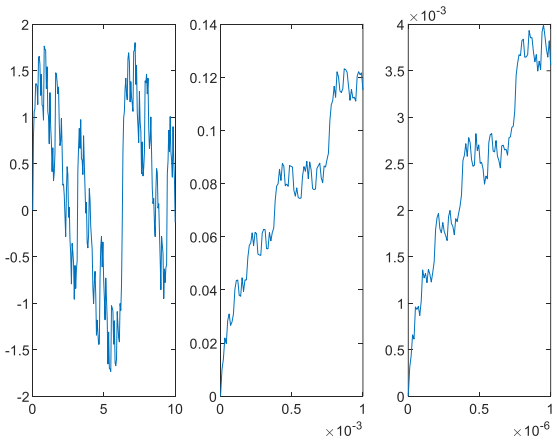
$$W(x)=\sum_{k=0}^{\infty}\lambda^{(s-2)k}\sin(\lambda^k x),\quad -\infty<x<\infty, \tag{5.1}$$

其中，$\lambda>1$且$1<s<2$。Weierstrass 函数是一种特殊的分形。

例 5.1　考虑函数$W(x)$的有限取值区间为$[a,b]$，且用有限项（前$n+1$项）来逼近它。绘制$W(x)$曲线的 Matlab 程序如下：

```
clc, clear
subplot(131), Weierstrass(2,1.5,100,0,10,200)
subplot(132), Weierstrass(2,1.5,100,0,1e-3,100)
subplot(133), Weierstrass(2,1.5,100,0,1e-6,100)

function Weierstrass(lambda,s,n,a,b,m);
K=[0:n]';              %级数的求和项
x=linspace(a,b,m);     %将区间[a,b]离散化
y=sum(lambda.^((s-2)*K).*sin(lambda.^K*x));
plot(x,y)
end
```



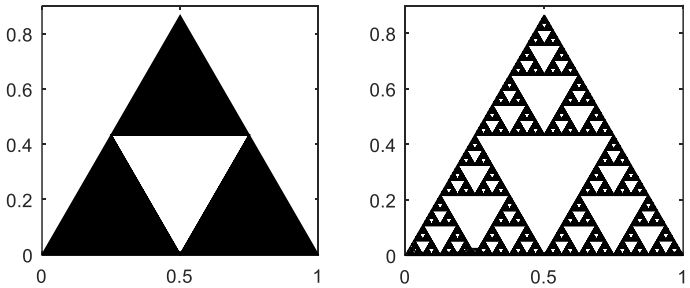
（A） [0,10]　（B） [0,10^{-3}]　（C） [0,10^{-6}]

图 5.1　不同区间上的 Weierstrass 函数逼近曲线

5.2　Sierpinski 三角形

Sierpinski 三角形是波兰数学家 Waclaw Sierpinski 1914 年构造的，其构造方法是取一个等边三角形，将其四等分，得四个较小的正三角形，然后去掉中间的那个三角形，保留周围的三个三角形（图 5.2（a）），然后，再将这三个较小的正三角形按上述方法分割与舍取，无限重复这种操作得到的几何图形便称为 Sierpinski 三角形，巴黎著名的埃菲尔铁塔正是以它为平面图，当然铁塔并没有将分形进行到无穷，但这已经体现了数学的美与精彩。

实现 Sierpinski 三角形的程序很多，但一般都是通过迭代函数系或仿射变换得到的，下面给出两种迭代算法和一种递归法来实现它，图 5.2（b）是经过 6 次迭代后的图形。



（a）一次迭代　（b）六次迭代

图 5.2　Sierpinski 三角形

例 5.2　用迭代算法生成图 5.2（b）中的 Sierpinski 三角形。

解　迭代算法的 Matlab 程序如下：

```
clc, clear, close all
sierpinski1(6)

function sierpinski1(N)
if nargin==0, N=6; end
z1=[1,0,(1+sqrt(3)*i)/2,1];
for k=1:N
    z2=z1; n=length(z2)-1;
    for m=0:n-1
        dz=(z2(m+2)-z2(m+1))/2;
        z1(6*m+1)=z2(m+1);
        z1(6*m+2)=z2(m+1)+dz;
        z1(6*m+3)=z1(6*m+2)+dz*((-1-sqrt(3)*i)/2);
        z1(6*m+4)=z1(6*m+3)+dz;
```

```
        z1(6*m+5)=z2(m+1)+dz;
        z1(6*m+6)=z2(m+1)+2*dz;
    end
    z1(6*n+1)=z2(n+1);
end
fill(real(z1),imag(z1),'k' ); axis equal, xlim([0,1]), ylim([0,0.9])
end
```

例 5. 3　用随机点迭代算法生成 Sierpinski 三角形。

解　迭代算法的 Matlab 函数如下：

```
clc, clear, close all
sierpinski2

function sierpinski2(N)
if nargin==0; N=20000; end
A=0; B=100; C=complex(50,sqrt(3)*50); %用复数表示三角形顶点A,B,C的坐标
P=complex(10,20);   %任取三角形内的一点
TP=[]; %所有生成点的初始化
gailv=randperm(N); %产生1到N的随机全排列
for k=gailv
        if k<N/3+1;
            P=(P+A)/2; %生成新点为点P和A的中点
        elseif k<2*N/3+1
            P=(P+B)/2; %生成新点为点P和B的中点
        else
            P=(P+C)/2; %生成新点为点P和C的中点
        end
        TP=[TP,P];   %TP中加入新点
end
plot(TP,'.','markersize',5)   %画所有生成的新点
end
```

画出的Sierpinski三角形见图5.3。

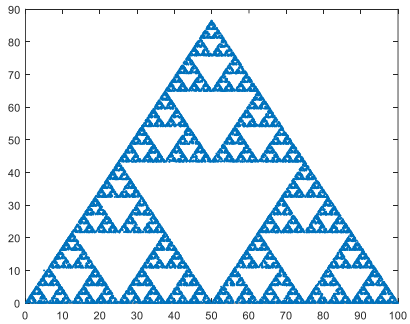
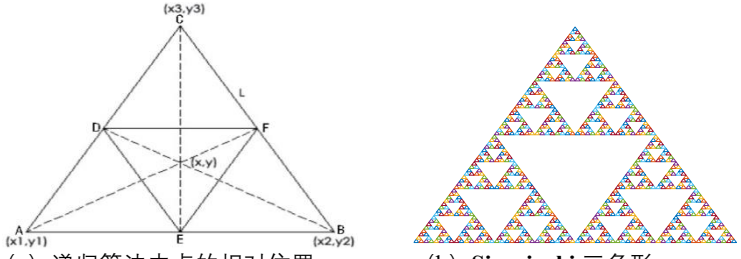


图 5.3　Sierpinski 三角形



(a) 递归算法中点的相对位置　　(b) **Sierpinski** 三角形

图 5. 4　递归算法绘制 **Sierpinski** 三角形

例 5.4 用递归算法生成 Sierpinski 三角形。

解 递归算法中点的相对位置见图 5.4(a)，使用 Matlab 画出的图形效果见图 5.4(b)。

递归算法的 Matlab 程序如下

```
function mysierpinski3(x,y,L,n)
if nargin==0; x=0; y=0; L=100; n=7; end
%x,y 为三角形中心点坐标，L 为三角形边长，n 为递归深度
axis off, hold on
if n==1
    z1=complex(x,y)-complex(L/2,L*tan(pi/6)/2);   %计算三角形顶点的复数坐标
    z2=complex(x,y)+complex(L/2,-L*tan(pi/6)/2);
    z3=complex(x,y)+complex(0,L*tan(pi/6)/2);
    plot([z1,z2,z3,z1]) %画三角形的边
else
    x01=x-L/4; y01=y-L*tan(pi/6)/4;   %计算小三角形中心的坐标
    x02=x+L/4; y02=y-L*tan(pi/6)/4;
    x03=x;   y03=y+L*tan(pi/6)/4;
    mysierpinski3(x01,y01,L/2,n-1)   %递归调用
    mysierpinski3(x02,y02,L/2,n-1)
    mysierpinski3(x03,y03,L/2,n-1)
end
```

5.3 Newton 分形

1.Newton 迭代法

Newton迭代法是解非线性方程

$$f(z)=0$$

的最著名的和最有效的数值方法之一。若初始值充分接近于根，则Newton法的收敛速度很快。

在不动点迭代中，用不同的方法构造迭代函数便得到不同的迭代方法。假设 $f'(z)\neq 0$，令

$$g(z)=z-\frac{f(z)}{f'(z)}, \tag{5.2}$$

则方程 $f(z)=0$ 和 $z=g(z)$ 是等价的。我们选取(5.2)为迭代函数，得到Newton迭代公式

$$z_{k+1}=z_k-\frac{f(z_k)}{f'(z_k)}, k=0,1,2,\cdots, \tag{5.3}$$

称 $\{z_k\}$ 为Newton序列。

取一个较简单的函数 $f(z)=z^n-1$，则 $f(z)$ 的一阶导数 $f'(z)=nz^{n-1}$，代入 Newton 迭代公式(5.3)得，

$$z_{k+1}=z_k-\frac{f(z_k)}{f'(z_k)}=z_k-\frac{z_k^n-1}{nz_k^{n-1}}, k=0,1,2,\cdots \tag{5.4}$$

(5.4)式就是下面使用的迭代计算公式。

2.Newton 分形的生成算法

在复平面上取定一个窗口，将此窗口均匀离散化为有限个点，将这些点记为初始点 z_0，按(5.4)式进行迭代。其中，大多数的点都会很快收敛到方程 $f(z)=z^n-1$ 的某一个零点，但也有一些点经过很多次迭代也不收敛。为此，可以设定一个正整数 M 和一个很小的数 δ，如果当迭代次数小于 M 时，就有两次迭代的两个点的距离小于 δ，即

$$|z_{k+1}-z_k|<\delta \tag{5.5}$$

则认为 z_0 是收敛的，即点 z_0 被吸引到方程 $f(z)=z^n-1=0$ 的某一个根上；反之，当迭代次数达到了 M，而 $|z_{k+1}-z_k|>\delta$ 时，则认为点 z_0 是发散（逃逸）的。这就是时间逃逸算法的基本思

想。

当点 z_0 比较靠近方程 $f(z)=z^n-1=0$ 的根时，迭代过程就很少；离得越远，则迭代次数越多甚至不收敛。

由此设计出函数 $f(z)=z^n-1$ 的 Newton 分形生成算法步骤如下：

（1）设定复平面窗口范围，实部范围为 $[a_1,a_2]$，虚部范围为 $[b_1,b_2]$，并设定最大迭代步数 M 和判断距离 δ。

（2）将复平面窗口均匀离散化为有限个点，取定第一个点，将其记为 z_0，然后按(5.4)式进行 M 次迭代。

每进行一次迭代，按(5.5)式判断迭代前后的距离是否小于 δ，如果小于 δ，根据当前迭代的次数 M 选择一种颜色在复平面上绘出点 z_0；如果达到了最大迭代次数 M 而迭代前后的距离仍然大于 δ，则认为 z_0 是发散的，也选择一种颜色在复平面上绘出点 z_0。

（3）在复平面窗口上取定第二个点，将其记为 z_0，按第（2）步的方法进行迭代和绘制。直到复平面上所有点迭代完毕。

例 5.5 按上面的算法绘制 Newton 分形图案。

解 编写绘制 Newton 分形图案的程序如下：

```
clc, clear, close all, c=30;        %设置颜色数
subplot(221), new(3,c), title('$f(z)=z^3-1$','Interpreter','Latex')
subplot(222), new(4,c), title('$f(z)=z^4-1$','Interpreter','Latex')
subplot(223), new(5,c), title('$f(z)=z^5-1$','Interpreter','Latex')
subplot(224), new(10,c), title('$f(z)=z^{10}-1$','Interpreter','Latex')

function new(n,c)                          %多项式的次数，c 为颜色数
if nargin==0; n=5; end
fz=@(z)z-(z.^n-1)/(n*z.^(n-1));   %定义 Newton 迭代函数
x=-1.5:0.01:1.5;
[x,y]=meshgrid(x); z=x+i*y;
for j=1:length(x);
    for k=1:length(y);
        n=0; zn1=z(j,k); zn2=fz(zn1);        %第一次 Newton 迭代
        while (abs(zn1-zn2)>0.01 & n<c-1) %n<c-1 限制颜色的种数
            zn1=zn2; zn2=fz(zn1); n=n+1;   %继续进行 Newton 迭代
        end
        f(j,k)=n;                                  %使用 c 种颜色
    end
end
pcolor(x,y,f); shading flat; axis square; colorbar
xlabel('Re $z$','Interpreter','Latex');
ylabel('Im $z$','Interpreter','Latex');
end
```

分形图案与颜色的种数选择有很大的关系，使用30种颜色的Newton分形图案见图5.5。

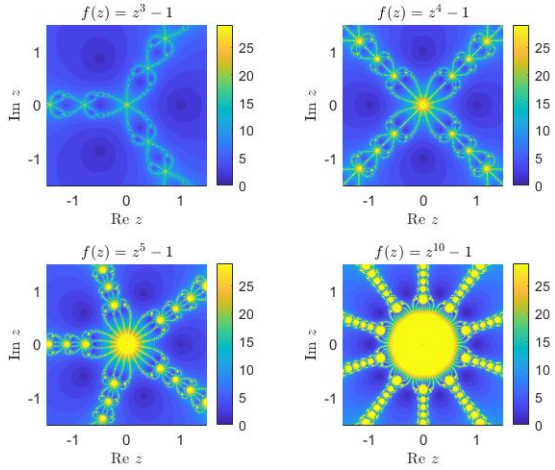


图 5.5　30 中颜色的 Newton 分形图案

注 5.1　使用其他的函数，可以生成多种多样的 Newton 分形图案。

5.4　Julia 集合

Julia集合是研究复平面上的迭代，考虑的是复平面上的一个二次映射

$$f(z)=z^2+c,\ z,c\in\mathbb{C},c=a+bi\in\mathbb{R} \tag{5.6}$$

的迭代行为，对于这个复平面上的迭代，等价于二维实平面上的迭代

$$\begin{cases} x_{n+1}=x_n^2-y_n^2+a, \\ y_{n+1}=2x_ny_n+b. \end{cases} \tag{5.7}$$

当$c=0.74543+i0.11301$时，利用(5.6)式进行迭代，给定几个初始值，计算发现迭代轨迹是收敛的。(5.6)式的迭代是否收敛与初始值有很大的关系。

1.Julia集

定义 5.1　一个平面区域上(5.6)式迭代的收敛点的集合称为填充 Julia 集。

下面是生成一个填充Julia集的算法：

（1）设定参数$c=a+ib$以及一个最大的迭代步数N。

（2）设定一个界限值R，例如实数$R\geq\max\left(2,\sqrt{a^2+b^2}\right)$。

（3）对于某矩形区域$[-a,a]\times[-b,b](a>0,b>0)$内的每一点进行迭代，如果对于所有的$n\leq N$，都有$|z_{n+1}|\leq R$，那么，在屏幕上绘制出相应的起始点，否则不绘制。

例 5.6　绘制 Julia 集图形。

解　编写的Matlab程序如下：

```
clc, clear, close all
subplot(121), julia1(-0.11+i*0.65,4,200),
title('$c=-0.11+0.65i$','Interpreter','latex')
subplot(122), julia1(-0.19+i*0.6557,4,200),
title('$c=-0.19+0.6557i$','Interpreter','latex')

function julia1(c,R,N)
if nargin==0
    c=-0.11+0.65*i; R=5; N=100;   %R为界限值，N为迭代的次数
end
x=linspace(-1.2,1.2,400);          %x方向取400个点
[x,y]=meshgrid(x); z=x+i*y;
f=zeros(size(x));                  %颜色矩阵的初始化
for  k=1:N
    f=f+(abs(z)<R); z=z.^2+c;
```

```
end
imagesc(f); shading flat; axis('square'); colorbar
xlabel('Re $z$','Interpreter','Latex');
ylabel('Im $z$','Interpreter','Latex');
end
```

$c=-0.11+0.65i$ 和 $c=-0.19+0.6557i$ 所绘制的分形图案见图5. 6。

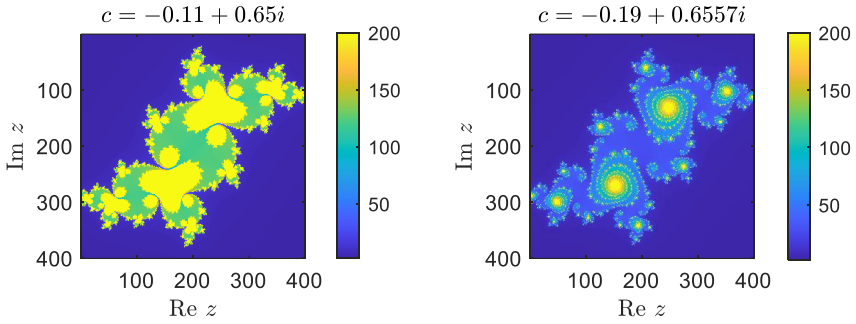


图 5. 6　$c=-0.11+0.65i$ 和 $c=-0.19+0.6557i$ 时的 Julia 集图形

2.广义Julia集

如果把(5.6)式的迭代函数换成其他的函数，生成的Julia集称为广义Julia集。

例5. 7　生成 $f(z)=c\cos(\pi z)$，$c=0.62+0.15i$ 时的广义Julia集图案。

解　画出的Julia集图案见图5. 7，画图的Matlab程序如下：

```
clc, clear, close all, julia2(0.62+0.15i,6,200)

function julia2(c,R,N)
if nargin==0
    c=0.62+0.15*i; R=5; N=100;   %R为界限值，N为迭代次数
end
x=linspace(-1.2,1.2,400); %x方向取400个点
[x,y]=meshgrid(x); z=x+i*y;
f=zeros(size(x)); %颜色矩阵的初始化
for  k=1:N
    f=f+(abs(z)<R); z=c*cos(pi*z);
end
f=mod(f,7); %只使用7种颜色
pcolor(f); shading flat; axis('square'); colorbar
xlabel('Re $z$','Interpreter','Latex');
ylabel('Im $z$','Interpreter','Latex');
end
```

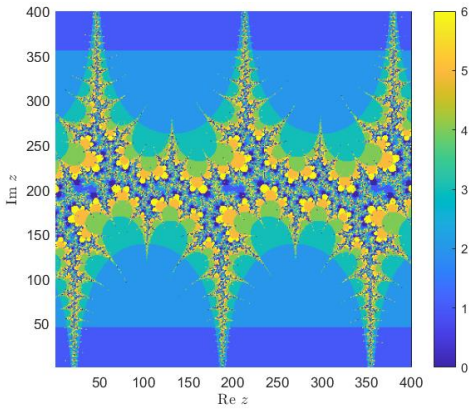


图 5. 7　$f(z)=c\cos(\pi z)$，$c=0.62+0.15i$ 时的广义 Julia 集图案

5.5 IFS分形

迭代函数系统（Iterated Function System，IFS）是一种构造分形的方法，所构造的结果往往是自相似的。以二维平面上的点为例，考虑如下形式的 k 个迭代函数系统：

$$\begin{bmatrix} x \\ y \end{bmatrix} = \boldsymbol{A}^{(i)} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1^{(i)} \\ b_2^{(i)} \end{bmatrix}, \quad i = 1, 2, \cdots, k \,, \tag{5.8}$$

其中，$\boldsymbol{A}^{(i)} = \begin{bmatrix} a_{11}^{(i)} & a_{12}^{(i)} \\ a_{21}^{(i)} & a_{22}^{(i)} \end{bmatrix}$ 为二阶方阵，$[b_1^{(i)}, b_2^{(i)}]^T$ 为常数向量。

对于给定的初始点 $[x_0, y_0]^T$，随机选取上述 k 个迭代公式中的某一个，第 i 个迭代公式被选中的概率为 p_i，显然有 $p_i > 0$，$\sum_{i=1}^{k} p_i = 1$，我们称点集 $\{(x_j, y_j)\}_{j=1}^{\infty}$ 的聚点为IFS吸引子。

采用如下方式选取迭代函数，在 $(0,1]$ 区间内按均匀分布产生一个随机数 ξ，如果 $\xi \in (0, p_1]$，则采用第1个迭代公式；如果 $\xi \in (p_1, p_1 + p_2]$，则采用第2个迭代公式，往后依次类推。

为描述问题方便，将 k 个迭代公式中的 $\boldsymbol{A}^{(i)}$ 和 $[b_1^{(i)}, b_2^{(i)}]^T$ 保存到一个 $k \times 6$ 维的矩阵 $\boldsymbol{M}$ 中，记为

$$\boldsymbol{M} = \begin{bmatrix} a_{11}^{(1)} & a_{21}^{(1)} & a_{12}^{(1)} & a_{22}^{(1)} & b_1^{(1)} & b_2^{(1)} \\ a_{11}^{(2)} & a_{21}^{(2)} & a_{12}^{(2)} & a_{22}^{(2)} & b_1^{(2)} & b_2^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{11}^{(k)} & a_{21}^{(k)} & a_{12}^{(k)} & a_{22}^{(k)} & b_1^{(k)} & b_2^{(k)} \end{bmatrix}, \quad \boldsymbol{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_k \end{bmatrix}.$$

例 5.8　绘出

$$\boldsymbol{M} = \begin{bmatrix} -0.64 & 0 & 0 & 0.5 & 0.86 & 0.25 \\ -0.04 & -0.47 & 0.07 & -0.02 & 0.49 & 0.51 \\ 0.2 & 0.33 & -0.49 & 0.43 & 0.44 & 0.25 \\ 0.46 & -0.25 & 0.41 & 0.36 & 0.25 & 0.57 \\ -0.06 & 0.45 & -0.07 & -0.11 & 0.59 & 0.1 \end{bmatrix}, \quad \boldsymbol{p} = \begin{bmatrix} 0.06 \\ 0.22 \\ 0.23 \\ 0.24 \\ 0.25 \end{bmatrix}$$

时的分形图形。

解　Matlab程序如下：

```
clc, clear, close all
M=[-0.64,0,0,0.5,0.86,0.25;-0.04,-0.47,0.07,-0.02,0.49,0.51
    0.2,0.33,-0.49,0.43,0.44,0.25;0.46,-0.25,0.41,0.36,0.25,0.57
    -0.06,0.45,-0.07,-0.11,0.59,0.1];
p=[0.06;0.22;0.23;0.24;0.25];
IFS(M,p)

function IFS(M,p)
N=10000;                          %迭代次数
a=cell(1,length(p));        %将M转换为若干个2×3维的矩阵
for i=1:length(p)
    a{i}=reshape(M(i,:),2,3);
end
xy=zeros(2,N);                  %用于存储迭代的数据点
pp=cumsum(p);                   %求概率向量的累积和
for i=1:N-1
    d=find(rand<=pp,1);   %选取第d个迭代函数
    xy(:,i+1)=a{d}(:,1:2)*xy(:,i)+a{d}(:,3);
end
plot(xy(1,:),xy(2,:),'k.','MarkerSize',3)
```

```
axis equal off
end
```
所绘制的图形如图5.8所示，该图形状像一棵树。

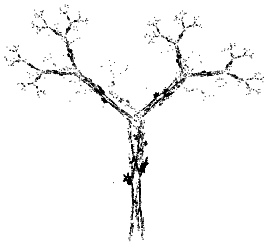


图 5.8　树状分形图

5.6　混沌图形

1.Logistic 映射

在客观实际问题中，存在一些动力学系统，其状态变量随时间的变化是离散的，这种系统称为离散动力学系统。Logistic 映射就是一种简单的离散动力学系统，其定义[4,103,104]如下：

$$x_{n+1}=\alpha x_n(1-x_n)\,, \tag{5.9}$$

其中α是系统的参数，且$0<\alpha<4$。Logistic 映射也称虫口模型，因为这个模型最初用来描绘昆虫的数量随时间的变化。由于资源的限制性，昆虫的数量不可能无限制的增加，当到达一个数量后，它们之间就会因为食物的缺少而竞争。因此，这个模型描述的就是繁殖和竞争同时存在时昆虫的数量随时间的变化情况。对于这样一个简单的映射，我们所关心的是其最终状态是什么。

例 5.9　画出 Logistic 模型状态变量x_n随参数α变化的图形。

图 5.9 给出了 Logistic 模型状态变量x_n随参数α变化的分岔图，图中横坐标是参数α，取值范围是 0~4，取样步长为 0.001；初值皆为$x_0=0.2$，纵轴为序列$X=\{x_n\}$的取值（每个α迭代 100 次，取后 30 次迭代值），范围为 0~1。图 5.9 也称为费根鲍姆图。

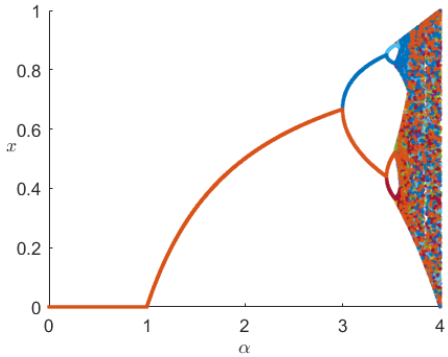


图 5.9　Logistic 模型的分岔图

画图的 Matlab 程序如下：

```
clc, clear, close all, hold on
a=[0:0.001:4]; y=zeros(30,length(a)); %数据矩阵初始化
for i=1:length(a)
    x1=0.2;
    for j=1:70
        x2=a(i)*x1*(1-x1); x1=x2;
        for j=71:100
            y(j-70,i)=x1; x2=a(i)*x1*(1-x1); x1=x2;
        end
    end
end
```

```
plot(a,y,'.')        %a 的一个点对应 y 的一列
xlabel('$\alpha$','Interpreter','Latex')
ylabel('$x$','Interpreter','Latex','Rotation',0)
```

2.洛伦茨方程

例 5.10　1963 年，美国气象学家 Edward　Lorenz 建立了大气对流的一个数学模型。该模型由常微分方程构成，即洛伦茨方程

$$\begin{cases}\dfrac{dx}{dt}=\sigma(y-x),\\ \dfrac{dy}{dt}=x(\rho-z)-y,\\ \dfrac{dz}{dt}=xy-\beta z,\end{cases}$$

其中 σ、ρ、β 为常数，取初值条件为 $x(0)=y(0)=z(0)=1$。

该常微分方程组是非线性的，下面采用欧拉法求数值解。以微分方程 $\dfrac{dx}{dt}=\sigma(y-x)$ 为例，当自变量的增量 Δt 较小时，该方程近似表示为

$$\frac{x(t+\Delta t)-x(t)}{\Delta t}\approx\sigma[y(t)-x(t)],$$

即

$$x(t+\Delta t)\approx x(t)+\sigma[y(t)-x(t)]\Delta t.$$

于是求解洛伦茨方程数值解的迭代公式为

$$\begin{cases}y(t+\Delta t)=x(t)+\sigma[y(t)-x(t)]\Delta t,\\ y(t+\Delta t)=y(t)+[x(t)(\rho-z(t))-y(t)]\Delta t,\\ z(t+\Delta t)=z(t)+[x(t)y(t)-\beta z(t)]\Delta t.\end{cases}$$

在计算中，取 $\sigma=10$，$\beta=8/3$，$\rho=28$，$\Delta t=0.01$，迭代次数为 5000（即 t 对应的取值区间为 $[0,50]$），具体求解程序如下：

```
clc, clear, close all, hold on
sigma=10; beta=8/3; rho=28; dt=0.01;
u=ones(3,1);        %初始条件
N=5000;             %最大迭代次数
plot3(u(1),u(2),u(3),'.','MarkerSize',5)
for i=1:N
    u=u+[sigma*(u(2)-u(1));u(1)*(rho-u(3))-u(2);u(1)*u(2)-beta*u(3)]*dt;
    plot3(u(1),u(2),u(3),'.','MarkerSize',5)
end
view(3)             %设置视角
```

绘出的图形如图 5.10 所示，从此图可以看出，对于所给定的参数，洛伦茨方程具有混沌解，解的图形像一只蝴蝶。

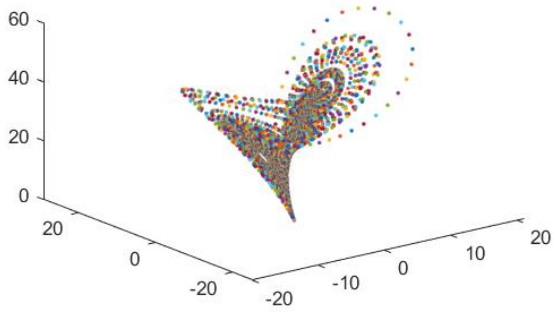


图 5.10　洛伦茨方程数值解图形

习题5

5.1　平面上3个点 $\boldsymbol{a}=(0,0)$、$\boldsymbol{b}=(2,0)$、$\boldsymbol{c}=(1,\sqrt{3})$ 构成一个三角形 $\Delta\boldsymbol{abc}$。选取该三角形的中心 $\boldsymbol{d}$，可构成三个三角形 $\Delta\boldsymbol{abd}$、$\Delta\boldsymbol{bcd}$、$\Delta\boldsymbol{cad}$。选取每个小三角形的中心，再构成三个三角形。重复上述步骤5次，绘出所有的三角形。

5.2　画出 $f(z)=\frac{1}{2}\sin(2z^2)+c$，$c=0.62+0.15i$ 时的广义Julia集图案。

5.3　考虑下面两种迭代系统：

$$\boldsymbol{M}^{(1)}=\begin{bmatrix}0.06 & 0 & 0 & 0.6 & 0 & 0\\ 0.04 & 0 & 0 & -0.5 & 0 & 1\\ 0.46 & -0.34 & 0.32 & 0.38 & 0 & 0.6\\ 0.48 & 0.17 & -0.15 & 0.42 & 0 & 1\\ 0.43 & -0.26 & 0.27 & 0.48 & 0 & 1\\ 0.42 & 0.35 & -0.36 & 0.31 & 0 & 0.8\end{bmatrix},\quad \boldsymbol{p}^{(1)}=\begin{bmatrix}0.1\\0.1\\0.1\\0.23\\0.23\\0.24\end{bmatrix};$$

$$\boldsymbol{M}^{(2)}=\begin{bmatrix}0.8 & -0.2 & 0.3 & 0.9 & -1.9 & -0.1\\ 0.1 & -0.5 & 0.5 & -0.4 & 0.8 & 7\end{bmatrix},\quad \boldsymbol{p}^{(2)}=\begin{bmatrix}0.7\\0.3\end{bmatrix},$$

分别绘出它们对应的分形图形。

5.4　已知函数 $f(x)=a-\left(x-\sqrt{a}\right)^2$，构造迭代公式 $x_{k+1}=f(x_k)$，$k=1,2,\cdots$。取 $x_1=0.1$，绘出 $a\in[0,4]$ 对应的费根鲍姆图。

5.5　Henon吸引子是混沌和分形的著名例子，迭代模型为

$$\begin{cases}x_{k+1}=1-1.4x_k^2+y_k,\\ y_{k+1}=0.3x_k,\end{cases}\quad k=1,2,\cdots,$$

取初值 $x_1=0$，$y_1=0$，进行2000次迭代。当 $k>1000$ 时，绘出点 (x_k,y_k)（不要连线），可得所谓的Henon引力线图，编写程序绘出Henon引力线图。