计算机理论基础与应用丛书

混沌的计算实验与分析

于万波 /著

科学出版社
www.sciencep.com

混沌的计算实验与分析

于万波 著

科学出版社
北京

内 容 简 介

本书从实验分析的角度介绍了混沌研究领域的基本概念以及一些最新的研究内容。书中提供了大量的程序以及一些新的研究思路。

第1章从迭代入手介绍了混沌与分岔等概念，同时介绍了经典的Lorenz系统族等内容；第2章对混沌的一些相关领域如神经网络、分形与图形设计、混沌加密、混沌智能模拟等进行了讨论；第3章对迭代的多种形式、混沌吸引子、复杂的分岔现象、混沌同步、系统混沌化等进行了研究。

本书可供想了解混沌的初学者使用，也可供想进一步研究学习混沌的学者参考，可作为混沌相关领域的教材或参考书。

图书在版编目（CIP）数据

混沌的计算实验与分析/于万波著. —北京：科学出版社，2008
（计算机理论基础与应用丛书）
ISBN 978-7-03-021659-5

Ⅰ. 混… Ⅱ. 于… Ⅲ. 混沌学-应用-计算方法 Ⅳ. O241

中国版本图书馆CIP数据核字（2008）第052719号

责任编辑：张海娜/责任校对：陈玉凤
责任印制：刘士平/封面设计：王 浩

科 学 出 版 社出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com

铭浩彩色印装有限公司印刷
科学出版社发行 各地新华书店经销
*

2008年6月第 一 版 开本：B5（720×1000）
2008年6月第一次印刷 印张：8 1/2
印数：1—5 000 字数：164 000

定价：28.00元

（如有印装质量问题，我社负责调换〈新欣〉）

前　　言

学习一门新的课程，进入一个新的研究领域，最好的办法是多思考、多动手实验，充分发挥学习主体的能动性。在这个过程中培养自己的研究兴趣，提高自己的学习研究能力。当然，知识是必要的。所以，本书在介绍混沌的基本知识、简单应用及一些相关研究进展的同时引导读者主动进入混沌这个领域。

混沌作为当前学术界研究的几个重大问题之一，已经成为许多学者的研究课题。目前，仍需要更多的混沌方面的著作、教材和读物。

本书尽量做到使用计算机程序算法以及简单的例题来说明一些重要但难于理解的概念定理，强调从直观的角度学习和理解混沌。这样做是为了遵循混沌发展的轨迹，使学生主动地投入到研究式的学习当中，当然，这样也失去了一些严密性，并且不能把一些好的理论结果归纳进来，这些内容读者可以参考其他著作。

作为一个新的研究领域，混沌的许多重要研究结果都比较简单，不需要太多的专业知识，所以把这些内容放到里面，代表着混沌研究的一些新的动态，同时也作为进一步研究的基础。在了解这些内容的同时，引导读者重新验证别人给出的结果，尝试发现一些新的问题。

本书可供想了解混沌和深入研究混沌的读者使用，也可作为高年级本科生、研究生的教材。本书要求读者具有基本的、简单的程序设计能力及基本的高等数学知识。书中的程序是使用 Matlab 语言实现的，稍加修改就可以转化成其他语言。

于硕编写了本书的 1.1 节、1.4 节与 2.1 节，黄艳峰编写了 2.3 节，于万波编写完成了其他章节及全书的统稿工作。

本书得到国家自然科学基金（批准号：70572069）以及淮海工学院科学著作基金的资助，在此表示感谢。

书中还有很多地方需要补充，也难免存在一些疏漏和不足，希望读者批评指正，非常感谢！

目　　录

第 1 章　混 沌 介 绍

1.1　迭　　代

1.1.1　迭代

【定义 1.1】　设 $f:X \rightarrow X$ 是集合 X 到自身的一个映射，记

$$\begin{aligned} f^0(x) &= x \\ f^1(x) &= f(x) \\ f^2(x) &= f(f(x)) \\ &\vdots \\ f^n(x) &= f \circ f^{n-1}(x) \end{aligned} \tag{1.1}$$

称 $f^n(x)$ 为一元函数 $f(x)$ 的 n 次迭代。其中 n 为非负整数，$f \circ f^{n-1}(x)$ 中的 $\circ$ 是函数复合运算符号。

在本书中，如果集合 X 没有特殊声明，都指实数集合。

复合运算是数学上的一种基本运算，例如：如果 $f(x) = \sin x, g(x) = x^2$，那么

$$f \circ g = f(g(x)) = \sin(x^2) = \sin x^2$$
$$g \circ f = g(f(x)) = (\sin x)^2$$

定义 1.1 可以用下面的表达式进行直观的解释。

如果 $f(x)=x+a$，那么

$$\begin{aligned} f^0(x) &= x \\ f^1(x) &= x + a \\ f^2(x) &= (x+a)+a = x+2a \\ &\vdots \\ f^n(x) &= x + na \end{aligned}$$

即 $f(x) = x+a$ 的 n 次迭代为 $f^n(x) = x + na$。

【例 1.1】　设 $f(x) = 2x+1$，当 $x = 0.6$ 时，计算 $f(f(f\cdots f(x)))$ 共 100 次复合，即 $f^{100}(x)$ 的值。

这个迭代的过程如下：

$$\begin{aligned} f^1(x) &= 2x + a \\ f^2(x) &= 2(2x+a)+a = 4x+3a \\ &\vdots \\ f^n(x) &= 2^n x + a + 2a + 4a + \cdots \end{aligned}$$

由于计算量比较大，所以使用计算机来实现。

设计下面 Matlab 程序：

```
x = 0.6;
for i = 1:100
    y = 2*x + 1;
    x = y;
end
y
```

通过运行程序，能够得到 $f^{100}(x)$ 的值是 2.0282E+30（即 2.0282×10^{30}），这是一个很大的数。

虽然这个程序是针对 $f(x)=2x+1$，当 $x=0.6$ 时设计的，但它也可以推广到其他函数表达式的迭代中。非常值得注意的是赋值语句 $x=y$ 的使用，正是由于这句话才实现了迭代。

改动语句 `x = 0.6`中的 0.6 即是改变迭代初始值；改动语句 `y = 2*x + 1`就是更改函数迭代式；改动语句 `for i = 1:100`中的 100 就是改变迭代次数。

【例 1.2】　$f(x)=\sin x$，那么当 $n\to\infty$ 时 $f^n(x)$ 会是什么结果呢？研究这个结果与 x 的初值是否有关。

修改例 1.1 中的程序如下：

```
x = 0.6;
for i = 1:100
    y = sin(x);
    x = y;
end
y
```

运行结果是 0.1652，当把 100 次改为 1000 次时，运行结果是 0.0545，能够看出随着迭代次数的增加，结果逐渐趋近于 0。当然这只是一个猜测，要得到准确的结论，必须进行大量的实验或给出理论证明。

【练习题】

1. 设 x 的初始值为 1.2，迭代表达式 $f(x)=x^2-1$，计算 $f(f(f\cdots f(x)))$ 共 10 次复合，即 $f^{10}(x)$ 的值。如果 x 的初始值为 0.9，迭代表达式不变，计算 $f^{10}(x)$ 的值。

2. 研究 $f(x)=\sin(x)+\cos(x)$ 的迭代情况，随意给出初始值进行迭代，观察其是否收敛到某个点（值），分析评价收敛速度的快慢。

1.1.2　二元二次迭代

【定义 1.2】　设 f,g 是映射 $\{X,X\}\to X$，记

$$\begin{cases} x_{n+1} = f(x_n, y_n) \\ y_{n+1} = g(x_n, y_n) \end{cases} \tag{1.2}$$

称式(1.2)为二元函数迭代。其中 n 为非负整数。

二元二次迭代是使用式(1.3)进行迭代：

$$\begin{cases} x_{n+1} = a_1 + a_2 x_n + a_3 x_n^2 + a_4 x_n y_n + a_5 y_n + a_6 y_n^2 \\ y_{n+1} = b_1 + b_2 x_n + b_3 x_n^2 + b_4 x_n y_n + b_5 y_n + b_6 y_n^2 \end{cases} \tag{1.3}$$

当 $n = 0$ 时给出(x_0, y_0)的值，然后把这个初始值代入式(1.3)的右边，即

$$\begin{cases} x_1 = a_1 + a_2 x_0 + a_3 x_0^2 + a_4 x_0 y_0 + a_5 y_0 + a_6 y_0^2 \\ y_1 = b_1 + b_2 x_0 + b_3 x_0^2 + b_4 x_0 y_0 + b_5 y_0 + b_6 y_0^2 \end{cases}$$

求出(x_1, y_1)，再把(x_1, y_1)的值代入式(1.3)右边，得到

$$\begin{cases} x_2 = a_1 + a_2 x_1 + a_3 x_1^2 + a_4 x_1 y_1 + a_5 y_1 + a_6 y_1^2 \\ y_2 = b_1 + b_2 x_1 + b_3 x_1^2 + b_4 x_1 y_1 + b_5 y_1 + b_6 y_1^2 \end{cases}$$

计算出(x_2, y_2)，以此类推。上述过程就是二元二次迭代，n 称为迭代次数，a_i，b_i($0 < i < 7$，i 是整数)称为迭代式系数。迭代出的点(数对)(x_0, y_0)，(x_1, y_1)，(x_2, y_2)，… 称为该迭代的轨迹。

【例 1.3】 一个简单的二元二次迭代式(Hénon 映射)为

$$\begin{cases} x_{n+1} = 1 - a x_n^2 + y_n \\ y_{n+1} = b x_n \end{cases} \tag{1.4}$$

绘制这个迭代得到的二元点(相空间)的轨迹图形。

所谓绘制二元点轨迹图形，就是把迭代出的(x_i, y_i)作为平面上的点，然后把这些点逐个在平面上绘制出来。

使用的程序为

```
N = 1000;
x = 0.3; y = 0.5;
a = 1.2; b = 0.3;
for i = 1:N
    x1 = 1 - a*x^2 + y;
    y1 = b*x;
    plot(x1,y1);
    hold on;
    x = x1;
    y = y1;
end
```

绘制出的图形如图 1.1 所示。

这个迭代绘制出的图形就是著名的 Hénon 吸引子。

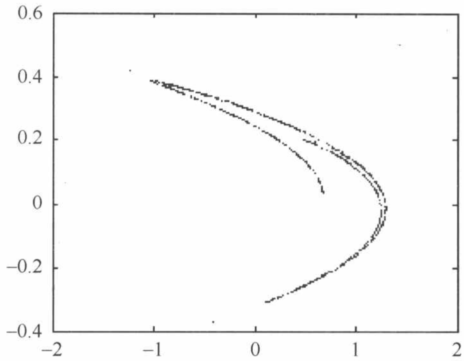


图 1.1　迭代轨迹图

程序中只出现了 x_0, y_0, x_1, y_1，在每次循环过程中 x_1, y_1 中都存储进新的轨迹点的坐标值，原来存储的点坐标值被覆盖了。不过，每一次的坐标值都被画在窗口中，用这种方式保留下来。

绘制多少个点由 N 决定，有些点距离很近，所以看上去没有 1000 个点。图形的形状与 $a=1.2, b=0.3$ 有直接关系，但与初始点 $x_0=0.3, y_0=0.5$ 几乎没有关系。

程序中，语句 plot(x1,y1)是在平面直角坐标系中(x_1, y_1)位置处绘制出一个点，hold on 是在图形窗口中保留原有(已经绘制出)的点。

【例 1.4】　一个二元二次迭代式为

$$\begin{cases} x_{n+1} = 0.0165 - x_n + x_n^2 - x_n y_n + y_n + y_n^2 \\ y_{n+1} = 0.1 - x_n + x_n y_n \end{cases} \tag{1.5}$$

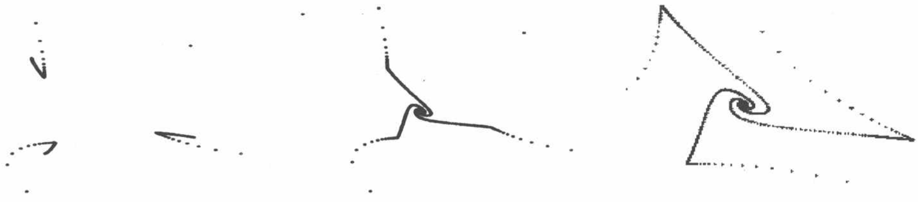
绘制这个迭代的轨迹图形。

设计下面程序：

```
x = 0.9;y = 0.6;
for i = 1:5000
    x1 = 0.0165 - x + x ^ 2 - x*y + y + y ^ 2;
    y1 = 0.1 - x + x *y;
    plot(x1,y1);
    hold on;
    x = x1;
    y = y1;
end
```

程序的运行结果如图 1.2(b)所示。

程序运行后，绘制出的点的运动轨迹是从外往里旋转，图 1.2(a)从外旋转到三个尖点后再往外生长，最后停留在三个点上。图 1.2(b)与 1.2(c)都是从外往里一直旋转，最后收敛到一个点上。



(a) (b) (c)

图 1.2 一个二次迭代的收敛轨迹

(a) 是常数项为 0.0164 时绘制出的图形；(b) 是常数项为 0.0165 时绘制出的图形；(c) 是把初始点改为 $x_0=0.215, y_0=0.215$，常数项为 0.0164 时画出的图形

这一节中介绍的迭代是一种最基本最简单的操作。但是，就是这种最基本的迭代也显示出了复杂的内涵。

【例 1.5】 用下面程序进行迭代操作，观察实验结果。

设计如下程序：

```
x = 0.2; y = 0.2;
for i = 1:10000
    x1 = -0.2 - x + x^2 - x*y + y + y^2;
    y1 = -0.1 - x + x*y;
    x = x1;y = y1;
    plot(x,y);
    hold on;
end
```

该程序使用的迭代表达式为

$$\begin{cases} x_1 = -0.2 - x + x^2 - xy + y + y^2 \\ y_1 = -0.1 - x + xy \end{cases}$$

程序的运行结果如图 1.3 所示。

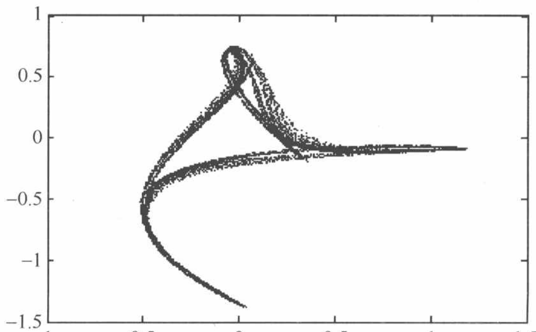


图 1.3 一个二次迭代的复杂轨迹

【练习题】

1. 调整例 1.4 中的程序进行实验,看哪些点先绘制,哪些点后绘制。

2. 在例 1.4 中不论是收敛到一点还是归结到三个点都不是混沌,改动系数与初始值进行实验,看是否可以出现混乱状态,观察绘制出的图形的变化情况。

3. 在例 1.5 中如果只想绘制 5000 个点的后 1000 个,如何修改程序?

1.1.3　线性随机 IFS 迭代

IFS 是英文 iterated function systems(迭代函数系统)的缩写。下面是一种常用的线性随机迭代算法。

【算法 1.1】 线性 IFS 随机迭代算法:

(1) 设定一个起始点(x_0, y_0)及总的迭代步数;

(2) 以概率 $P_k(k=1,2,3,\cdots,N)$ 选取仿射变换 W_k,W_k 形式为

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \tag{1.6}$$

(3) 以 W_k 作用点(x_0, y_0),得到新坐标(x_1, y_1);

(4) 令 $x_0 = x_1, y_0 = y_1$;

(5) 在屏幕上打出(x_0, y_0);

(6) 重返第(2)步,进行下一次的迭代,直到迭代次数大于总步数为止。

其中,$W_k(k=1,2,3,\cdots,N)$ 是一组收缩仿射变换;式(1.6)叫做线性 IFS 迭代表达式。

使用算法 1.1,确定表达式参数,对于一些确定的表达式参数,可以绘制出树叶与山等自然景物。

算法 1.1 的(2)中的 W_k 是用矩阵形式表示的,写成一般的表达式为

$$\begin{cases} x_1 = ax_0 + by_0 + e \\ y_1 = cx_0 + dy_0 + f \end{cases} \tag{1.7}$$

整个迭代的流程与前两节都是一样的,计算出一点然后再代回到右边的式中,不过这里有若干个迭代式,每次迭代只能选择一个迭代式进行,选择哪一个迭代式要靠概率决定。要绘制多少个点,取决于迭代总次数。

【例 1.6】 下面一共有 6 个迭代式,每个迭代式的 a,b,c,d,e,f 都已经给出(见表 1.1),每次选取各个表达式的概率 P_k 也已经给出(初始点随意给出),迭代

表 1.1　使用 IFS 绘制树叶的系数数据表

k	a	b	c	d	e	f	P_k
1	0.00	0.00	0.00	0.16	0.00	0.00	0.01
2	0.85	0.04	−0.04	0.85	0.00	1.60	0.85
3	0.20	−0.26	0.23	0.22	0.01	1.60	0.07
4	−0.15	0.28	0.26	0.24	0.00	0.44	0.07

5000次(也可以多一些或少一些)，绘制出的图形如图1.4所示。

设计程序如下：

```
a=[0  0  0  0.16  0  0; 0.85  0.04  -0.04  0.85  0  1.6;
  0.2  -0.26  0.23  0.22  0.01  1.6; -0.15  0.28  0.26  0.24  0  0.44];
x0=1; y0=1;
for i=1:5000
  r=rand;
  if r<=0.01
    x1=a(1,1)*x0+a(1,2)*y0+a(1,5);
    y1=a(1,3)*x0+a(1,4)*y0+a(1,6);
  end
  if r>0.01 & r<=0.86
    x1=a(2,1)*x0+a(2,2)*y0+a(2,5);
    y1=a(2,3)*x0+a(2,4)*y0+a(2,6);
  end
  if r>0.86 & r<=0.93
    x1=a(3,1)*x0+a(3,2)*y0+a(3,5);
    y1=a(3,3)*x0+a(3,4)*y0+a(3,6);
  end
  if r>0.93 & r<=1
    x1=a(4,1)*x0+a(4,2)*y0+a(4,5);
    y1=a(4,3)*x0+a(4,4)*y0+a(4,6);
  end
  x0=x1; y0=y1;
  plot(x1,y1); hold on;
end
```

上面的Matlab程序是根据算法1.1编制的，运行后绘制出树叶来，如图1.4(a)所示。

【例1.7】 例1.6中的程序不改动，只要改变表1.1中的数据就可以绘制出山来，绘制出山的数据如表1.2所示。

表1.2　使用IFS绘制山的系数以及概率等数据表

	a	b	c	d	e	f	P_k
1	0.0800	−0.0310	0.0840	0.0306	5.1700	7.9700	0.0300
2	0.0801	0.0212	−0.0800	0.0212	6.6200	9.4000	0.0250
3	0.7500	0.0000	0.0000	0.5300	−0.3570	1.1060	0.2200
4	0.9430	0.0000	0.0000	0.4740	−1.9800	−0.6500	0.2450
5	−0.4020	0.0000	0.0000	0.4020	15.5130	4.5880	0.2100
6	0.2170	−0.0520	0.0750	0.1500	3.0000	5.7400	0.0700
7	0.2620	−0.1050	0.1140	0.2410	−0.4730	3.0450	0.1000
8	0.2200	0.0000	0.0000	0.4300	14.6000	4.2860	0.1000

绘制出的图形如图 1.4(b)所示。

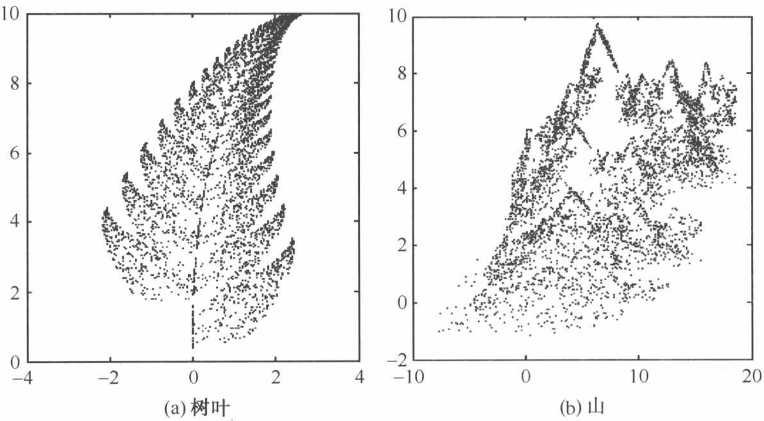


图 1.4　两个 IFS 迭代轨迹

线性随机 IFS 迭代也是迭代的一种,由于其是线性的,所以目前的研究成果相对比较多。一般把(绘制出的)图 1.4 所示的点集也叫做(近似的混沌)吸引子。

【练习题】

1. 使用例 1.7 中给出的数据,编写(改写)程序绘制出山来。

2. 修改算法 1.1,使其每个表达式都是三元一次方程,最后绘制出空间点集。

3. 如果例 1.6 中的各个表达式被选择的概率 P_k 分别为 0.1,0.5,0.07,0.33,如何修改程序?

1.1.4　三元二次迭代

1.1.2 节和 1.1.3 节研究了二元二次迭代及线性随机迭代,本节研究一般的三元二次迭代。三元二次迭代一共由三个式子构成一个迭代表达式,但至少有一个式子是三元二次多项式。式(1.8)就是一个三元二次的迭代表达式。

$$\begin{cases} x_n = -0.3 - x_{n-1} + x_{n-1}^2 - x_{n-1}y_{n-1} + y_{n-1} + y_{n-1}^2 \\ y_n = 0.2 - x_{n-1} + x_{n-1}y_{n-1} \\ z_n = -x_{n-1}y_{n-1} - y_{n-1}z_{n-1} - x_{n-1}z_{n-1} \end{cases} \tag{1.8}$$

使用三元二次迭代能够绘制出漂亮的三维空间混沌吸引子。

【例 1.8】　使用下面程序绘制一个三元二次迭代的吸引子。

```
y = 0.2; x = 0.2; z = 0.2; q = 1;
for i = 1:8000
    x1 = ( - 0.3 - x + x ^ 2 - x * y + y + y ^ 2)*q;
    y1 = (0.2 - x + x * y)*q;
    z1 = ( - x*y - y*z - z*x)*q;
```

```
        plot3(x1,y1,z1);
        hold on;
        x = x1; y = y1; z = z1;
    end
```

改变 q 的值，绘制出的图形可以从混沌吸引子演化为极限环，如图 1.5 所示。

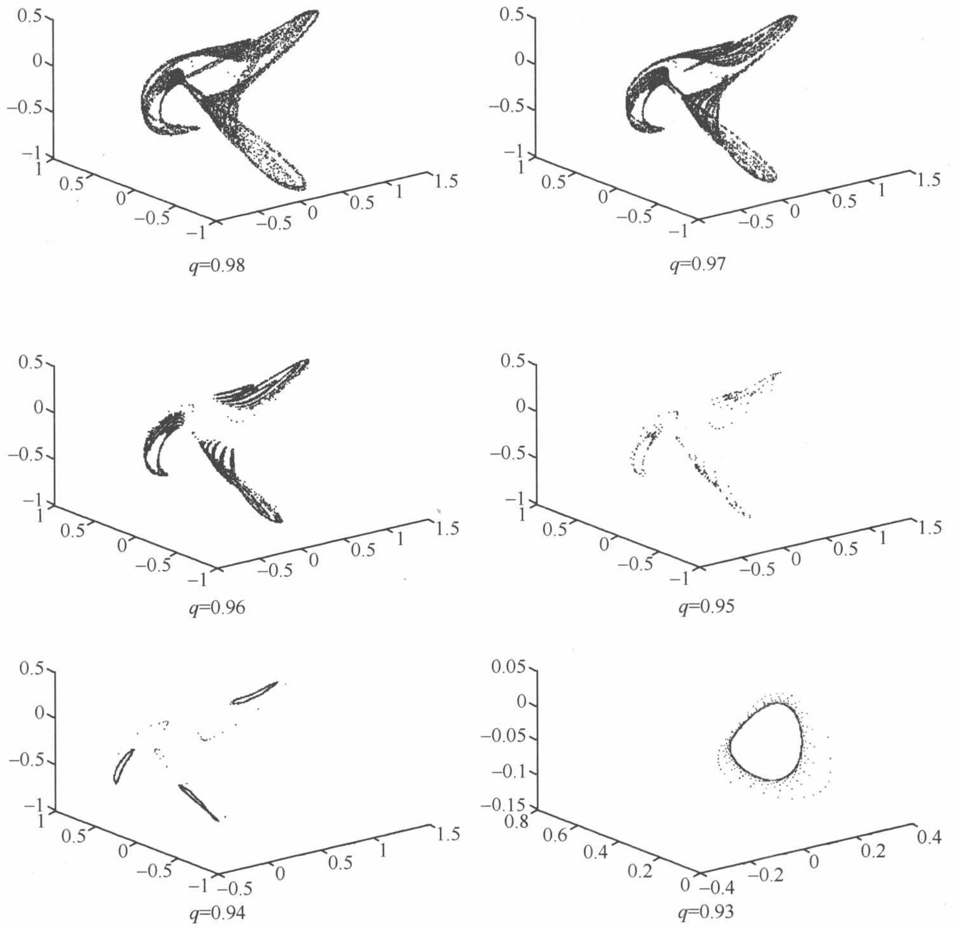


图 1.5　一个三元二次迭代的吸引子

【思考题】

其他的迭代表达式乘以倍数会出现类似的情况吗？

【练习题】

1. 如果例 1.8 使用的表达式都是一次的，是否能绘制出复杂的混沌吸引子？

2. 设计程序，用实验方法寻找一个三元二次迭代，使其能够绘制出漂亮的迭代轨迹图形。

1.1.5　三角函数迭代

三角函数迭代一般绘制出的吸引子比较复杂，理论分析也比多项式迭代困难。

1. Clifford 系统

下面使用著名的 Clifford 系统进行计算实验，其绘制出的混沌吸引子更加“混沌”，从中可以直观地进行推断，三角函数迭代也可以出现混沌。

【例 1.9】 绘制 Clifford 系统的混沌吸引子的三维图形。

Clifford 系统由式(1.9)中的三个式子构成：

$$\begin{cases} x_1 = \sin(ay) - z\cos(bx) \\ y_1 = z\sin(cx) - \cos(dy) \\ z_1 = e\sin(bx) \end{cases} \tag{1.9}$$

设计程序如下：

```
x = -10.0; y = -1.0; z = -1.0; r = 0; s = 0; t = 0;
a = 2.24; b = 0.43; c =- 0.65; d =- 2.43; e = 1.0;
for i = 1:10000
    x1 = sin(a*y) - z*cos(b*x);
    y1 = z*sin(c*x) - cos(d*y);
    z1 = e*sin(b*x);
    if i >3000
       plot3(x1,y1,z1);
    end
    hold on;
    x = x1; y = y1; z = z1;
end
```

绘制出的图形如图 1.6 所示。

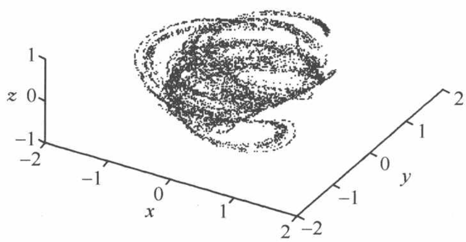


图 1.6　一个三角函数迭代轨迹

【例 1.10】 把 Clifford 系统的第二个式子乘以 $\sin\left(\frac{1}{x}\right)$，分析振荡函数对控

制迭代轨迹的作用。

改动 Clifford 系统的迭代式，变为

$$\begin{cases} x_1 = \sin(ay) - z\cos(bx) \\ y_1 = [z\sin(cx) - \cos(dy)]\sin\left(\dfrac{1}{x}\right) \\ z_1 = e\sin(bx) \end{cases}$$

各个参数值与例 1.9 中相同，绘制出的图形如图 1.7 所示。

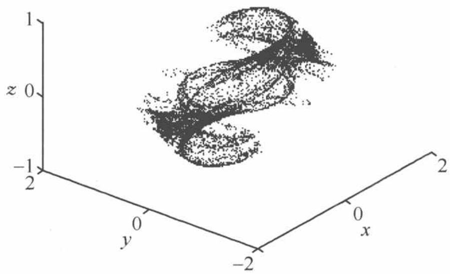


图 1.7　振荡函数对 Clifford 系统迭代轨迹的控制作用(1)

改动 Clifford 系统的迭代式，变为

$$\begin{cases} x_1 = \sin(ay) - z\cos(bx) \\ y_1 = [z\sin(cx) - \cos(dy)]\sin\left(\dfrac{1}{y}\right) \\ z_1 = e\sin(bx) \end{cases}$$

各个参数值与例 1.9 中相同，绘制出的图形如图 1.8 所示。从图 1.8 中可以看出，系统的迭代轨迹已经看不出明显的纹理，呈现出极端的“混沌”状态。

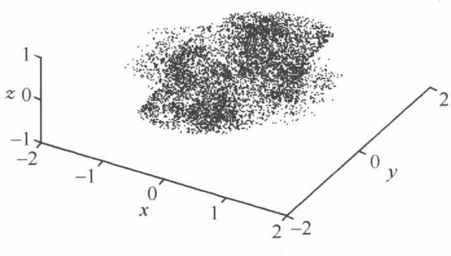


图 1.8　振荡函数对 Clifford 系统迭代轨迹的控制作用(2)

【思考题】

根据上面实验探索中的现象尝试给出一些理论上的分析，例如：式(1.9)迭代表达式有三个式子，第一个式子乘以 $\sin\left(\frac{1}{x}\right)$ 或者第二个式子乘以 $\sin\left(\frac{1}{y}\right)$ 都会对混沌轨迹产生严重的影响，那么是否可以得到这样的结论：在给 x,y,z 赋值的