组　　号	
实验题目	函数迭代与分形
队员姓名	1. 向飞宇
	2.
	3.
	4.

目　录

1 实验一：用循环实现迭代函数（1）

问题描述：以 $f(x) = F(x + \log_{10} 3)$ 为迭代函数，$F(x)$ 为函数的小数部分，取不同的增量，观察迭代点在 $[0, 1]$ 上的分布情况.

本题使用 MatLab 内置的 floor 函数来获得函数的小数部分，将迭代初值设定为 $x_0 = 0.3$，使用一个有限次（此题中取为 1000000 次）的循环来实现迭代. 以下以迭代结果在 $[0, 1]$ 区间上按 1000 等分绘制直方图，统计迭代点落入各个区间的次数，所得到的结果如下图所示.

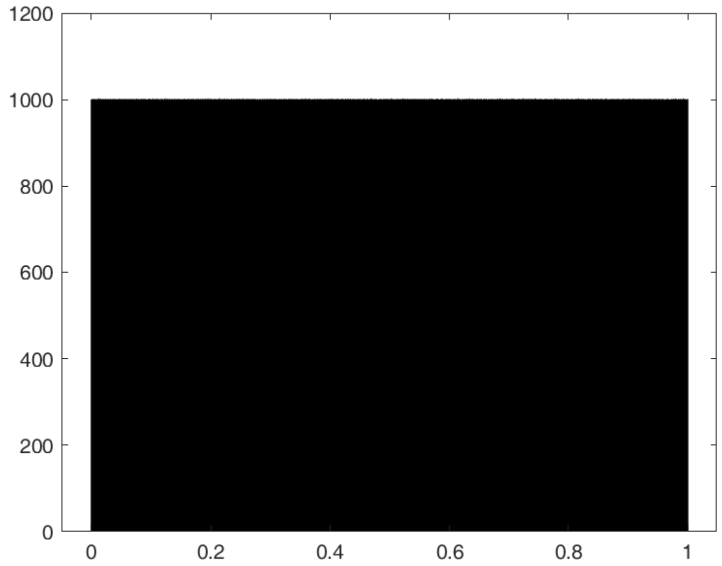

图 1 迭代结果的直方图

可以发现，这个直方图中的每个区间上的计数次数非常均匀，这表明这个函数可以用来生成类似均匀分布的伪随机数，而且 $\log_{10} 3$ 是一个效果非常好的伪随机数种子.

下面考虑一般的情况 $f(x) = F(x + p)$，p 为任意的正无理数. 以下对不同的增量 p 进行实验，分别取 $p = \pi, \sqrt{5}, \mathrm{e}$，实验结果如下所示.

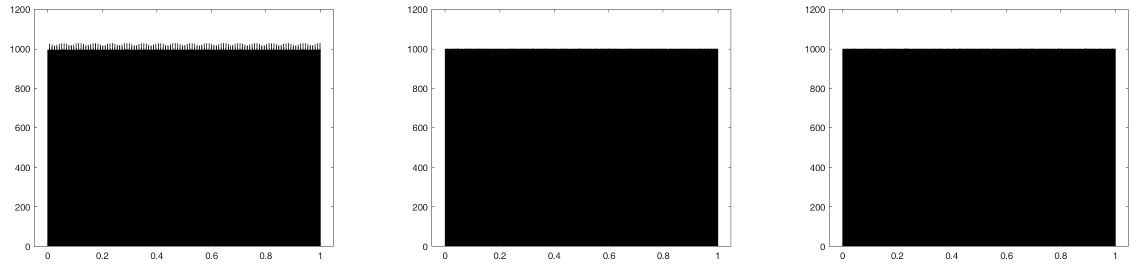

图 2 以不同增量作迭代结果的直方图

从以上的结果可以看出，当 $p = \pi$ 时，每个区间上的计数次数的均匀程度比 $p =$

$\log_{10}3$ 略低，但是 $p=\sqrt{5}$, e 计数的均匀程度与 $p=\log_{10}3$ 基本相同，所以可以做出这样的推断：对于大部分无理数，可以使用函数的小数部分 $F(x)$ 来作为 $[0,1]$ 上的均匀分布的伪随机数.

2 实验二：用循环实现迭代函数（2）

问题描述：绘制 Clifford 迭代系统

$$\begin{cases} x_{n+1}=\sin(ay_n)-z_n\cos(bx_n) \\ y_{n+1}=z_n\sin(cx_n)-\cos(dy_n) \\ z_{n+1}=e\sin(bx_n) \end{cases}$$

的迭代图，其中 $a=2.24, b=0.43, c=-0.65, d=-2.43, e=1.0.$

这个问题与上一个问题类似，都可以通过一个循环解决问题，只不过这个题目中有三个变量，且三个变量相互关联，所以可以采用与上一实验中相同的方法，唯一不同的就是我们使用了三个数组来处理这个问题. 以下是迭代图与实验结果.

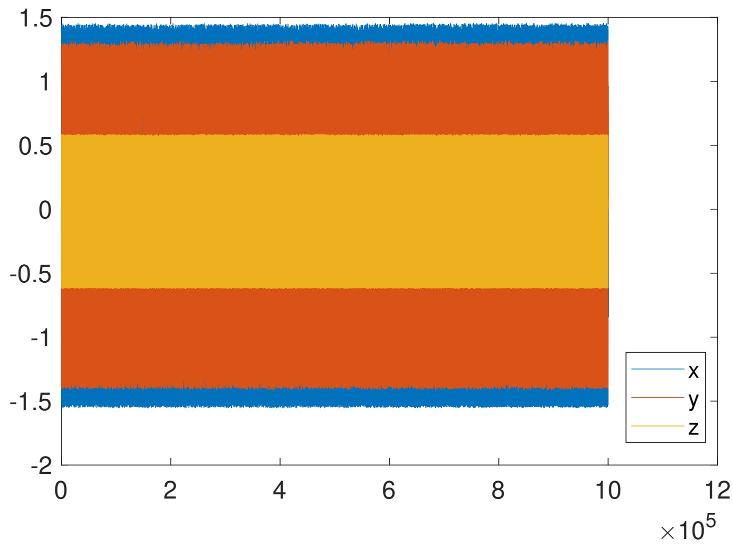

图 3　x,y,z 的二维空间中各个变量的迭代图

从 x, y, z 的二维空间中的迭代图可以看出，x, y, z 三个变量分别能够均匀分布在 $1.4, 1.3, 0.6$ 左右，这说明这个系统可能与上一个系统类似，但是我们仍然需要根据各个变量作出迭代结果的直方图，才能判断变量的分布是否均匀，以下是每个迭代结果的直方图.

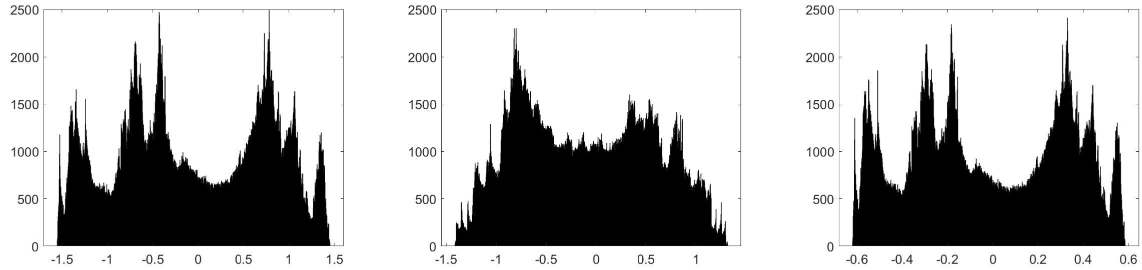

图 4　x, y, z 的迭代结果的直方图

从迭代结果可以看出，变量的分布不是均匀的，x, z 的分布情况相似，但是 y 的分布与其它两个变量不同. 而且这个系统不能够生成均匀分布的随机数. 以下再将 x, y, z 的结果放在三维空间中绘制.

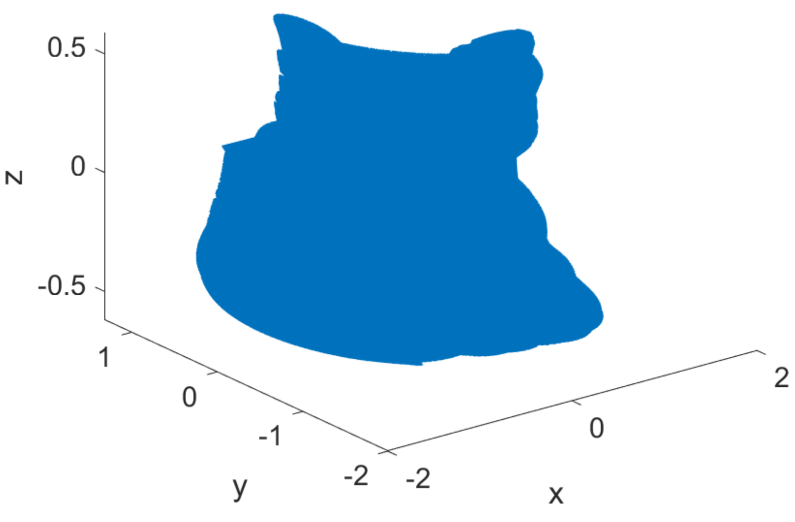

图 5　x, y, z 在三维空间中的迭代图

从三维空间的迭代图可以看出，这个系统的稳定性不好.

3　实验三：用递归实现树形分形图

问题描述：在一初始给定的竖直线段的顶端，引出两个线段，其夹角与给出的初始线段成角，且长度为初始线段的二分之一，在新生成的线段上，以同样的方法便生成下一幅

图，如此进行下去，绘制所生成的树形图.

这一个问题是一个典型的递归问题, 所以需要设计递归模式和递归出口. 很显然, 无论是否使用递归，绘制一条线段都是核心部分，而针对此递归问题，需要设计其传入参数. 因为这个问题涉及到线段夹角，所以单纯使用两个点的坐标是不合适的，本题目中需要用一个点的坐标加上绘制长度和绘制角度来处理，同时将作图终点的坐标作为输出参数，这样每个作图步骤直接可以与下一个步骤联系起来.

然后需要设计递归模式，每一层递归都可以看作以下三个步骤，假设传入的参数为坐标 O，角度 θ，绘制长度 l:

- 画出“树干”，即中线，得到新的坐标点 O';
- 画右子树，递归层数减少 1，递归传入参数 $O', \theta - \frac{\pi}{6}, \frac{l}{2}$;
- 画左子树，递归层数减少 1，递归传入参数 $O', \theta + \frac{\pi}{6}, \frac{l}{2}$;

最后，递归出口设定为递归层数为 1，递归初始传入的参数为递归层数，$O = (0,0), \theta = \frac{\pi}{2}, l = 1$. 以下是递归层数为 12 的作图结果.

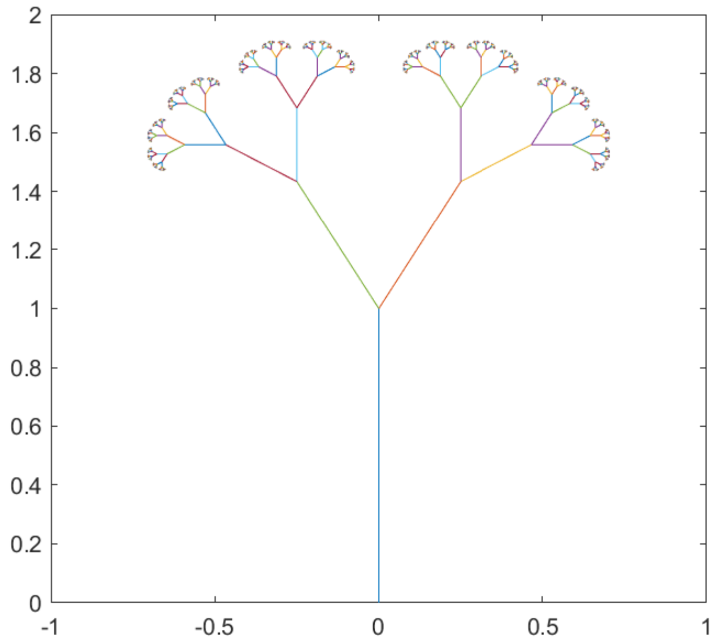

图 6　递归层数为 12 的树形图

4 实验四：用迭代逼近函数方程的解

问题描述：设函数方程 $f(x) = h_0 f(2x) + h_1 f(2x-1) + \cdots + h_5 f(2x-5)$**，且**

$$\begin{cases} h_0 = \dfrac{(1+\cos\alpha+\sin\alpha)(1-\cos\beta-\sin\beta)+2\sin\beta\cos\alpha}{4} \\ h_1 = \dfrac{(1-\cos\alpha+\sin\alpha)(1+\cos\beta-\sin\beta)-2\sin\beta\cos\alpha}{4} \\ h_2 = \dfrac{1+\cos(\alpha-\beta)+\sin(\alpha-\beta)}{2} \\ h_3 = \dfrac{1+\cos(\alpha-\beta)-\sin(\alpha-\beta)}{2} \\ h_4 = 1-h_0-h_2 \\ h_5 = 1-h_1-h_3 \end{cases}, \alpha, \beta \in [-\pi, \pi]$$

请考虑如下函数迭代：

$$\eta_0(x) = \chi_{[0,1)}(x) = \begin{cases} 1, x \in [0,1) \\ 0, x \notin [0,1) \end{cases}, \eta_{n+1}(x) = \sum_{k=0}^{5} h_k \eta_n(2x-k), n = 0, 1, \cdots$$

若上述函数迭代序列 $\eta_n(x)$ **有极限，则极限函数满足函数方程. 绘制** $f(x)$ **的图形.**

这个问题当中的函数 $\eta_0(x)$ 是一个示性函数，所以在 MatLab 中可以使用匿名函数来处理这个问题. 使用匿名函数可以很方便地表达迭代的函数，而且虽然我们不一定能够得知函数的形式，但是仍然可以通过 MatLab 中的内置函数 fplot 来作出匿名函数的图像.

以下分别是 $\alpha = \frac{\pi}{6}, \beta = \frac{\pi}{3}$；$\alpha = -\frac{\pi}{6}, \beta = -\frac{\pi}{3}$；$\alpha = \frac{\pi}{3}, \beta = \frac{\pi}{6}$ 作出 $f(x)$ 的图像. 由于计算原因，我们只计算到 $\eta_5(x)$.

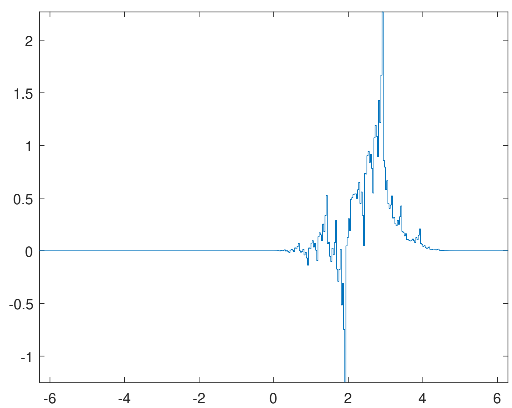

图 7 $\alpha = \frac{\pi}{6}, \beta = \frac{\pi}{3}$ **时** $f(x)$ **图像**

从示性函数 $\eta_0(x)$ 可以看出，这个函数是一个间断函数，但是在经过了几次迭代以后，这个函数的性质慢慢变好了，且从图像中可以看出，这个函数的图像很像一个波形.

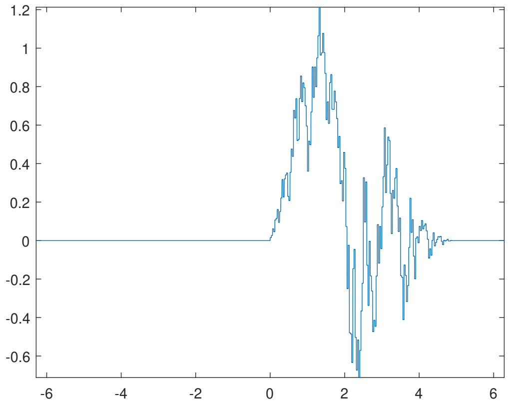

图 8 $\alpha = -\frac{\pi}{6}, \beta = -\frac{\pi}{3}$ **时** $f(x)$ **图像**

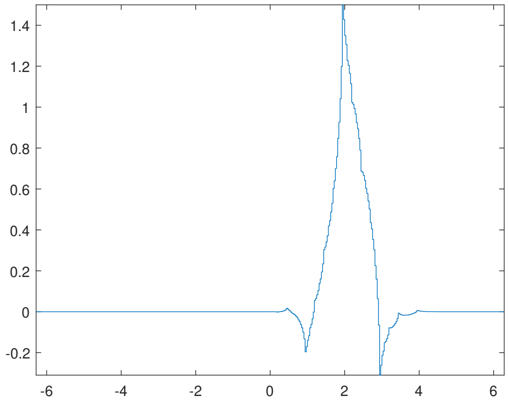

图 9 $\alpha = \frac{\pi}{3}, \beta = \frac{\pi}{6}$ **时** $f(x)$ **图像**

而对于不同的 α, β 取值，得到的结果也有很大差异，这说明如果选取恰当的 α, β 值，可能可以实现滤波器的效果.

附录 A 实验一的代码

```
series1 = iterate(0.3, log10(3), 1000000);
figure(1)
plot(series1);
histogram(series1, 1000)

series2 = iterate(0.3, pi, 1000000);
figure(2)
plot(series2);
histogram(series2, 1000)
```

```
series3 = iterate(0.3, sqrt(5), 1000000);
figure(3)
plot(series3);
histogram(series3, 1000)

series4 = iterate(0.3, exp(1), 1000000);
figure(4)
plot(series4);
histogram(series4, 1000)

function series = iterate(x0, p, iteration)
    series(1) = x0;
    for i = 1: iteration
        series(i + 1) = series(i) + p - floor(series(i) + p);
    end
end
```

附录 B 实验二的代码

```
a = 2.24;
b = 0.43;
c = -0.65;
d = -2.43;
e = 1.0;
iteration = 1000000;
[x, y, z] = clifford(a, b, c, d, e, iteration);
figure(1)
plot(x);
hold on;
plot(y);
plot(z);
hold off;
figure(2)
histogram(x, 1000)
```

```
figure(3)
histogram(y, 1000)
figure(4)
histogram(z, 1000)
figure(5)
plot3(x, y, z)

function [x, y, z] = clifford(a, b, c, d, e, iteration)
    x(1) = 0;
    y(1) = 0;
    z(1) = 0;
    for i = 1: iteration
        x(i + 1) = sin(a * y(i)) - z(i) * cos(b * x(i));
        y(i + 1) = z(i) * sin(c * x(i)) - cos(d * y(i));
        z(i + 1) = e * sin(b * x(i));
    end
end
```

附录 C 实验三的代码

```
recursion(12, [0, 0], pi / 2, 1)

function [x, y] = draw_line(orig, angle, length)
    x = orig(1) + length * cos(angle);
    y = orig(2) + length * sin(angle);
    plot([orig(1), x], [orig(2), y])
    axis([-1, 1, 0, 2])
    hold on
    drawnow;
end

function recursion(iter, orig, angle, length)
    if (iter == 1)
        draw_line(orig, angle, length);
    else
```

```
        [x, y] = draw_line(orig, angle, length);
        recursion(iter - 1, [x, y], angle - pi / 6, length / 2);
        recursion(iter - 1, [x, y], angle + pi / 6, length / 2);
    end
end
```

附录 D 实验四的代码

```
eta = @(x) x >= 0 && x < 1;

alpha = pi / 3; % this value can change
beta = pi / 6; % this value can change
h0 = ((1 + cos(alpha) + sin(alpha)) * (1 - cos(beta) - sin(beta)) + 2 *
↪  sin(beta) * cos(alpha)) / 4;
h1 = ((1 - cos(alpha) + sin(alpha)) * (1 + cos(beta) - sin(beta)) - 2 *
↪  sin(beta) * cos(alpha)) / 4;
h2 = (1 + cos(alpha - beta) + sin(alpha - beta)) / 2;
h3 = (1 + cos(alpha - beta) - sin(alpha - beta)) / 2;
h4 = 1 - h0 - h2;
h5 = 1 - h1 - h3;

for i = 1: 5
    eta = @(x) h0 * eta(2 * x) + h1 * eta(2 * x - 1) + h2 * eta(2 * x -
    ↪  2) + h3 * eta(2 * x - 3) + h4 * eta(2 * x - 4) + h5 * eta(2 * x
    ↪  - 5);
end

fplot(eta, [-2 * pi, 2 * pi])
```