计算物理 作业报告1

PB14203209 张静宁

第一题

以 $x_{n+1}=\lambda sin(\pi x_n)$ 为迭代方程：

(1)画出系统状态随参数λ的变化图，要求包括定值状态、倍周期分叉和混沌状态；

(2)列出各个倍周期分叉处的λ值，求相应的 Feigenbaum 常数。

算法思路

计算系统状态、绘图

对一定区间内的每一个 λ，任意选定 x 的初始值，计算并输出若干次迭代后 x 的值。

迭代公式为：$x_{n+1}=\lambda sin(\pi x_n)$

比如，λ 的区间为 $[Min, Max]=[-100,100]$，间隔 $Step=0.001$ 取一个值；初始值 $x=0.15$，迭代计算$100,000$次，并且输出迭代最后 $Num=100$ 个 x。

每次输出数据为对应的 λ 和 迭代结果 x，是横坐标为 λ、纵坐标为 x 图上的一个数据点。

总的数据点数量 $N=(Max-Min)*Num*Step$，为了程序跑的快一些，一般取$10,000$~ $100,000$个数据点比较合适。根据所有输出的数据点做图，即可得到系统状态随参数λ的变化图。

确定倍周期分叉点

用Python脚本对输出结果进行数据处理，可以得到倍周期分叉点。具体来说，由于执行程序输出的结果是文本文件。取出数组中的数据点后是一个大的 $N\times 2$ 的二维数组。

$[[\lambda_1,x_{11}],[\lambda_1,x_{12}],\cdots,[\lambda_1,x_{1n}],[\lambda_2,x_{21}],\cdots,[\lambda_2,x_{2n}],\cdots,[\lambda_n,x_{nn}]]$

把 λ 和x 分别赋值到一个一维数组，再把 x 根据每Num个赋值成一个二维数组里的项。

$[\lambda_1,\lambda_2,\cdots,\lambda_n]$

$[x_{11},x_{12},\cdots,x_{nn}]$

$x_{list}=[[x_{11},\cdots,x_{1n}],[x_{21},\cdots,x_{2n}],\cdots,[x_{n1},\cdots,x_{nn}]]$

最后每个 x_{list} 里的子列表中相同的元素合并，及得到每个λ对应的一组定态循环的x，就可以得到倍周期分叉点。

$x_{list}=[[x_1],[x_2,x_3],[x_4,x_5],\cdots,[x_k,x_{k+1},x_{k+2}],[x_{k+3},x_{k+4},x_{k+5}],\cdots]$

程序使用说明

编程环境：Ubuntu，要预装 C 和 Python

- chaos.c 为用 C 语言编写的计算程序(1)，负责计算并输出数据点
- chaos 为 C 语言编译后的可执行文件
- plot.py 为用 Python 编写绘图脚本
- analysis.py 为用Python 编写的计算倍周期分叉点的脚本
- feigenbaum.c 为用 C 语言编写的计算程序(2)，负责计算并输出数据点
- feigenbaum 为 C 语言编译后的可执行文件

分别执行以下三条命令：

```
$ gcc chaos.c -o chaos -lm            # 编译 chaos.c 程序
$ ./chaos > data                      # 在当前目录下执行 chaos 程序，并将结果输出到 data 文件
$ python plot.py                      # 执行绘图程序，将生成的图片保存为 pdf 和 png 格式
```

在执行第二条命令后会输出如下信息，提醒用户输入相关参数：

```
Please input some parameters.
The number of data points N = (Max-Min)*Num*Step            # N 是总的数据点x的数量
Min(the min of lambda): -10                                 # Min 是 lambda 下区间
Max(the max of lambda): 0                                   # Max 是 lambda 上区间
Num(the number of x to print out for each lambda): 100      # Num 是每个lambda输出的x个数
Step(the step of lambda): 0.01                              # Step 是 lambda 的间隔
Totally 99099 points!                                       # 本次一共输出99099个数据点
```

在运行第三条命令后，可以在results文件夹中得到三个文件：

- *.txt 当前条件下得到的数据文件
- *.pdf 当前条件下得到的图片
- *.png 当前条件下得到的图片

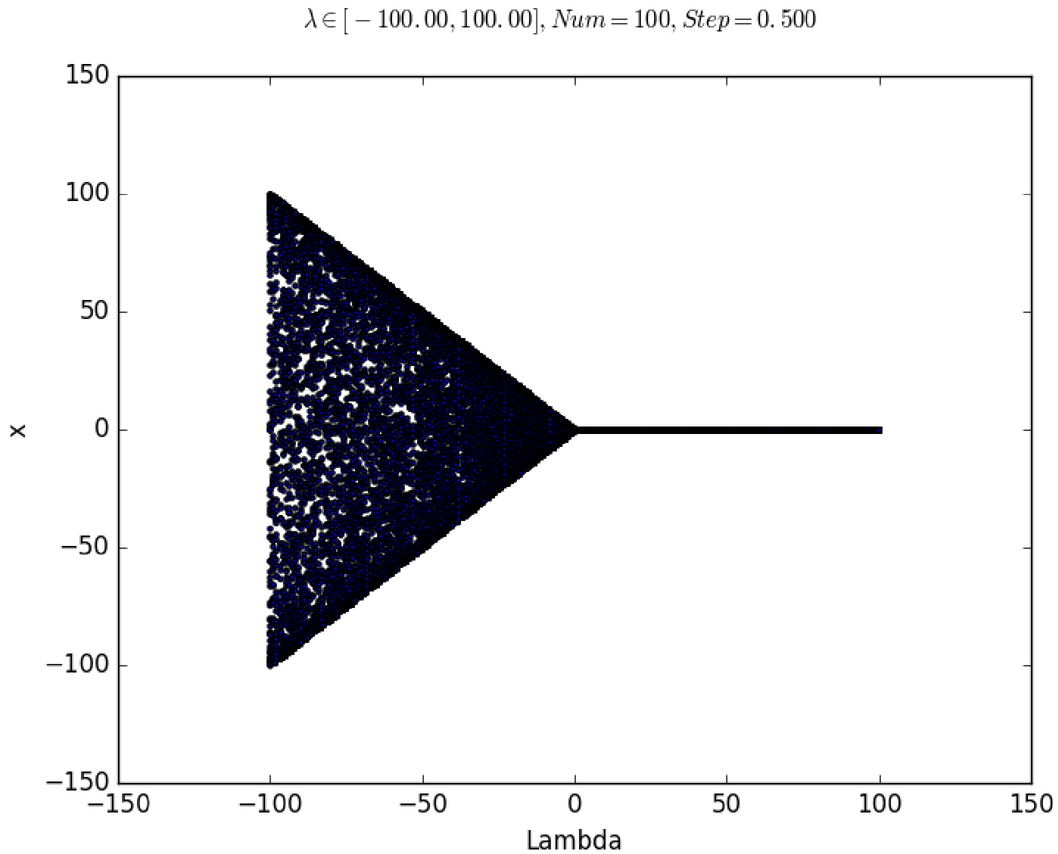
说明：在做图的具体过程中，多次改变输入参数，改变Python绘图脚本可以得到不同效果的图片。以上做图过程中 plot.py 曾被修改，故和目前提交的版本可能有区别。

计算结果与分析

以下计算结果总的迭代次数都是$100,000$次，图中 Num 是每个λ 输出的 x 个数（即同一个横坐标值对应的纵坐标点数），$Step$ 是 λ 的间隔（即在区间中每隔 $Step$ 取一个 λ。

1 大体情况

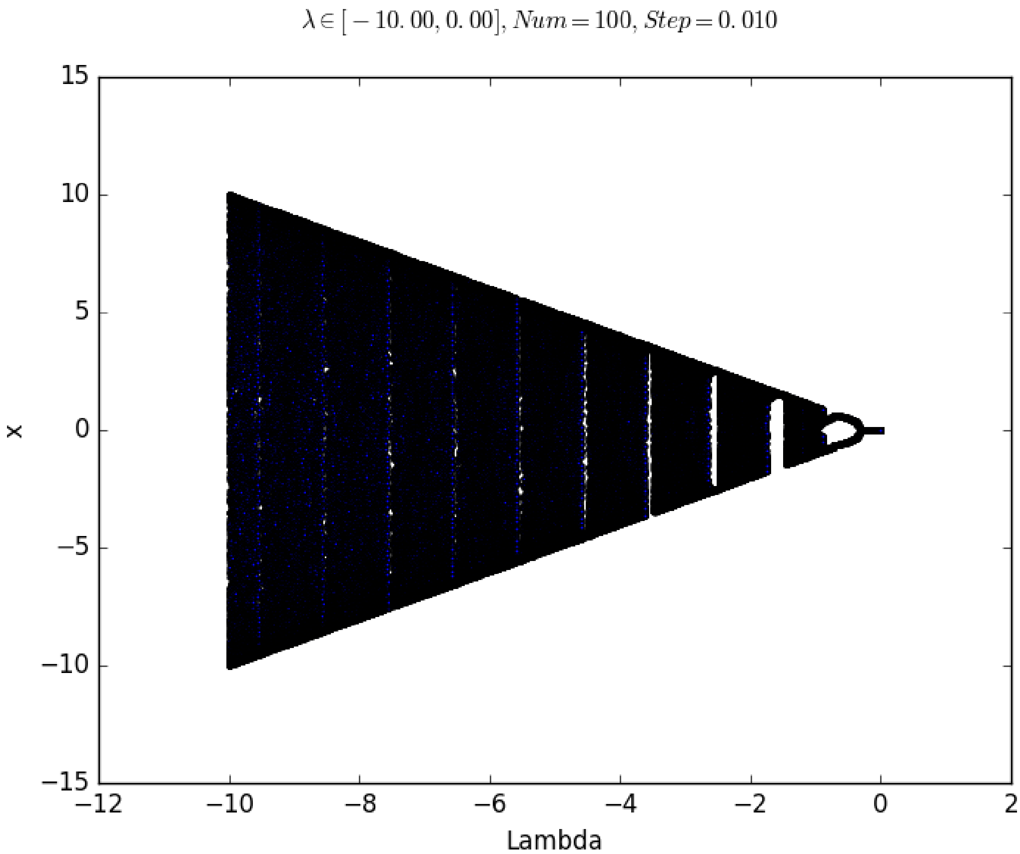
一开始，$\lambda \in [-100, 100]$，将区间取大一些可以看看大体的情况。计算结果绘图如下：

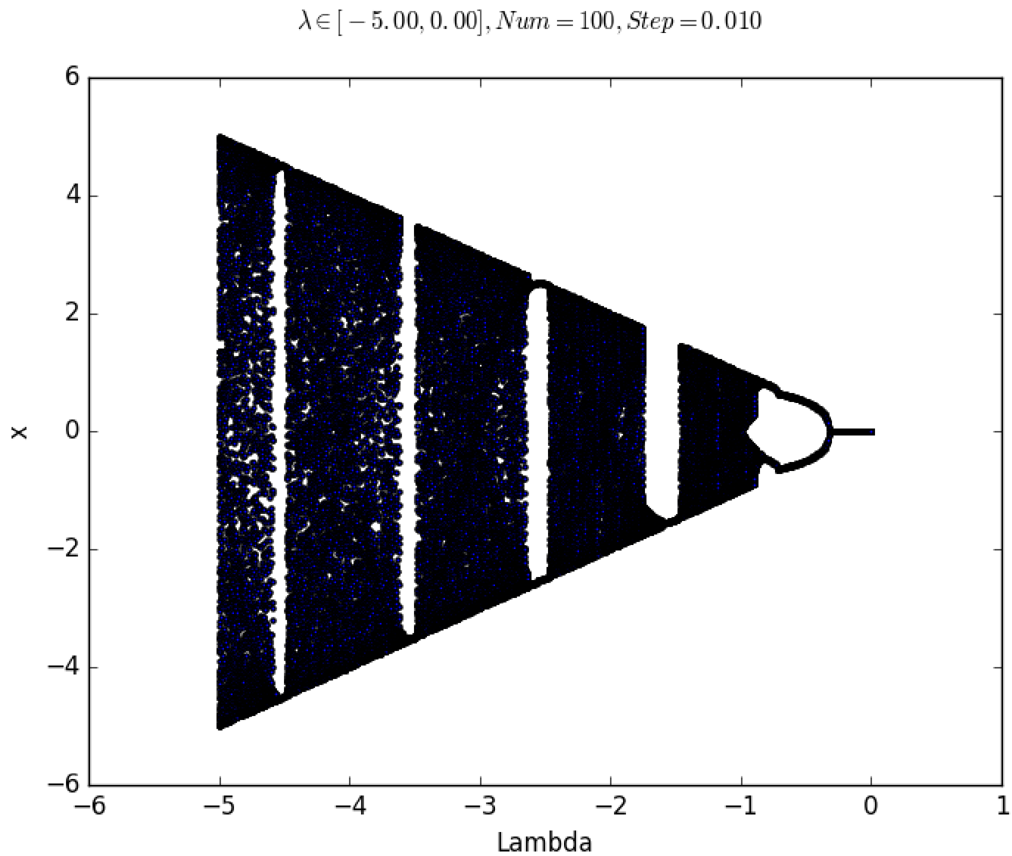


可见当 $\lambda > 0$ 时，x 经过多次迭代后都趋向于0，系统绝灭；$\lambda < 0$ 时，系统的行为比较丰富，值得展开研究。

2 周期窗口

当 λ 区间缩小，计算结果绘图如下：



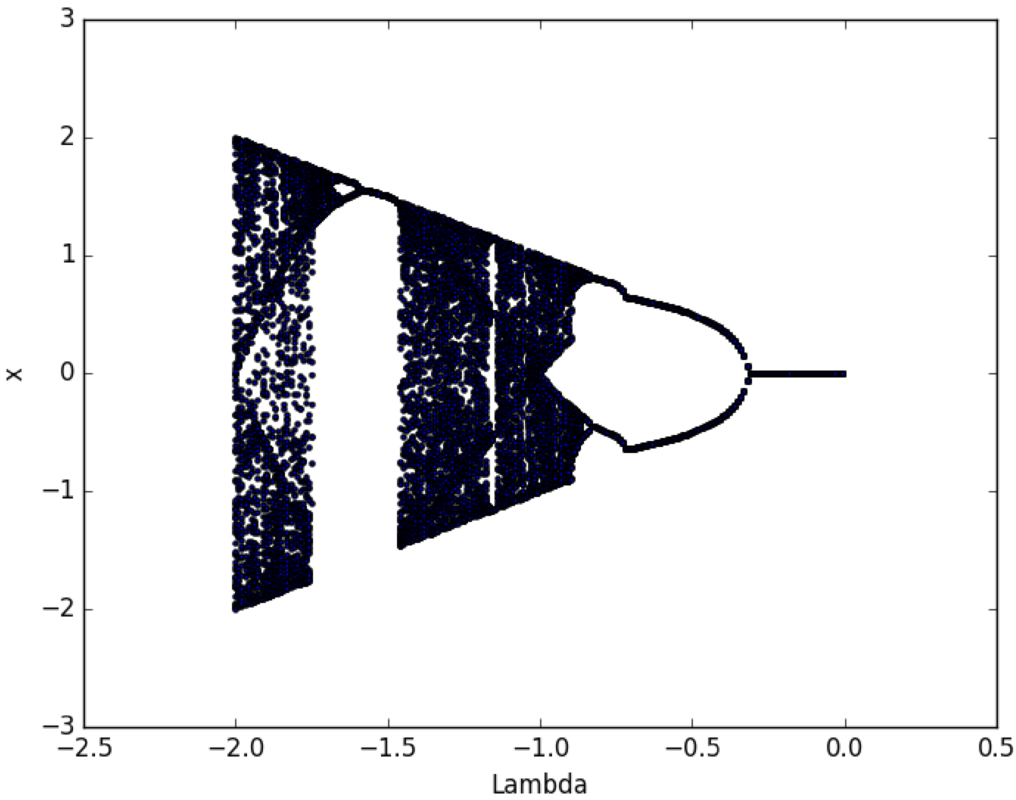


很明显 λ 在 [-1,0] 之间出现了倍周期分叉，随后系统进入混沌状态，之后又好几次出现定值状态→倍周期分叉→混沌状态的循环，白色窗口即为周期窗口。

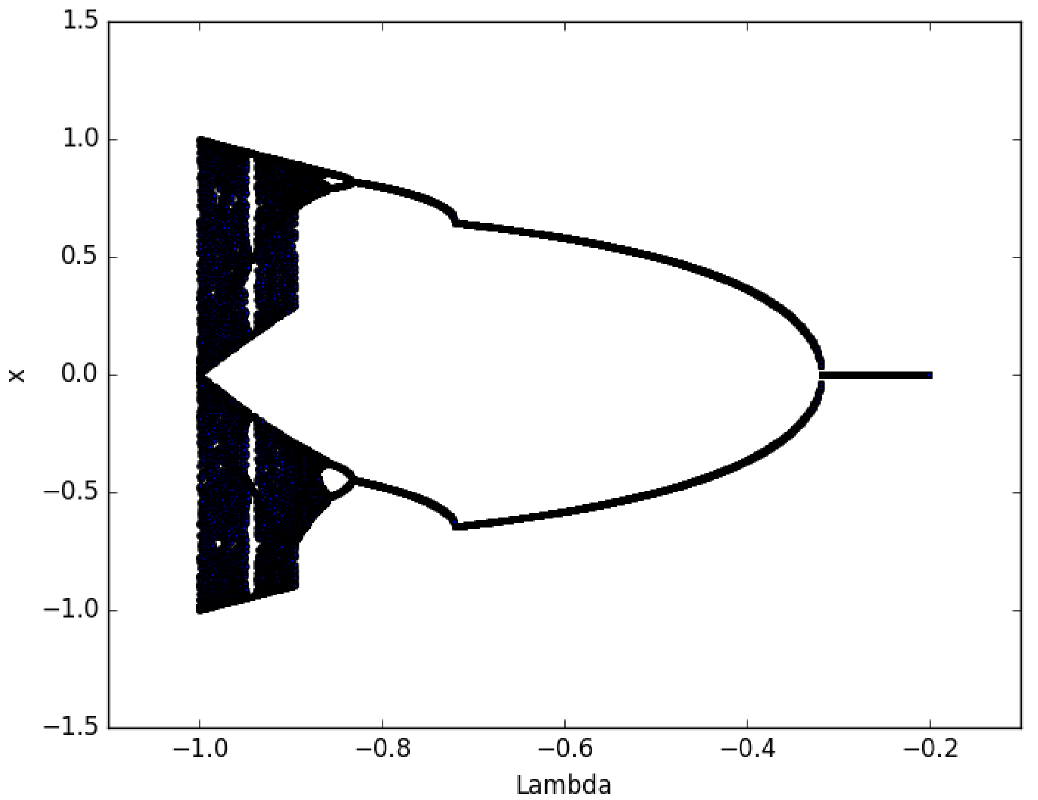
3 向上、向下两种状态

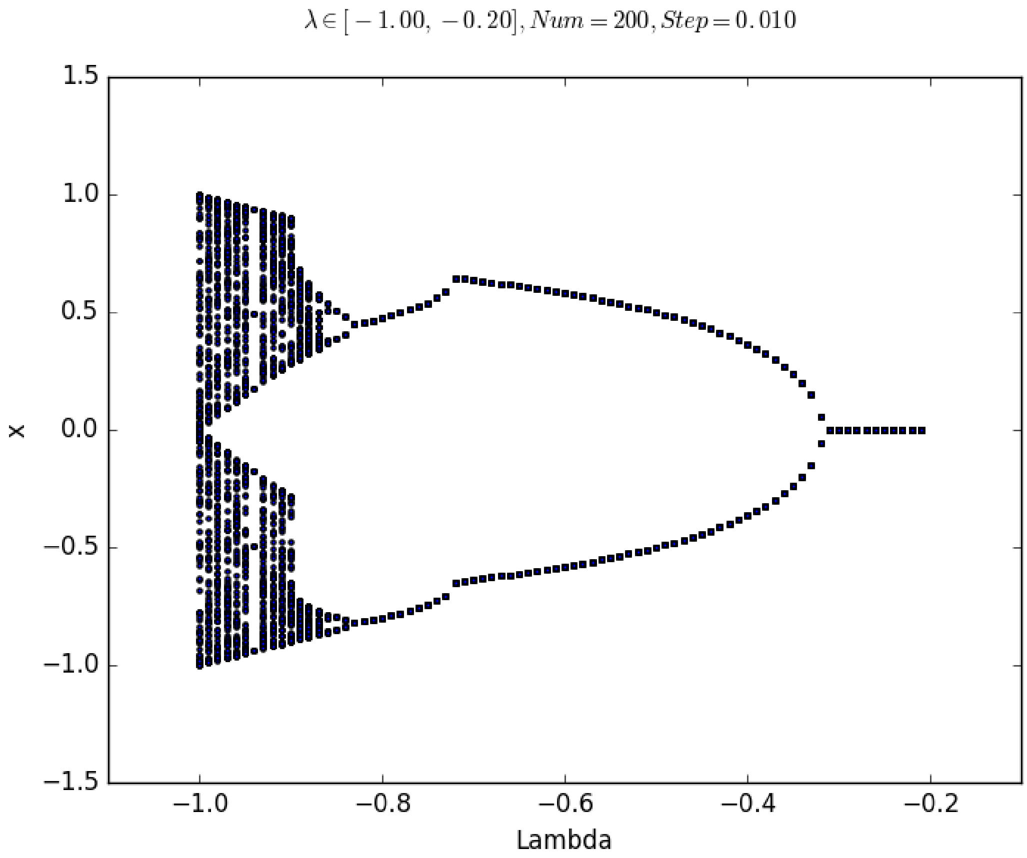
继续放大，研究 $\lambda=0$ 附近的系统状态，计算结果绘图如下：

$\lambda \in [-2.00, 0.00], Num=100, Step=0.010$



$\lambda \in [-1.00, -0.20], Num=100, Step=0.001$





对比以上两张图片，会发现在 $\lambda = -0.8$ 附近两幅图系统状态的区别，前一幅线条是向上弯，后一副是向下弯。改变λ的间隔$Step$ 或者是Num的值，多次出现这种情况。个人猜想，应该不是计算出错，而是系统本身包含的一个性质，即存在两组不同的倍周期分叉状态。具体出现这种现象的原因，还有待研究。

4 相反的情况

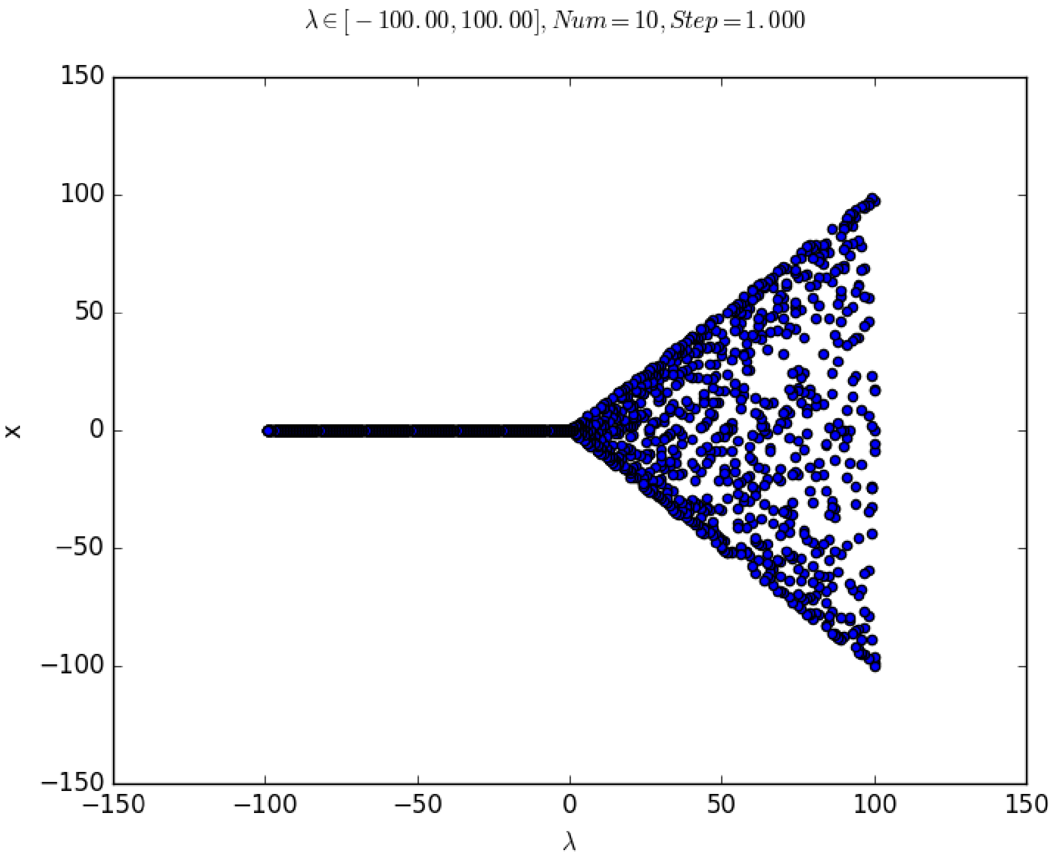
经过和同学们的讨论，发现我的图和他人的结果是相反的。其他同学的计算结果，在 $\lambda > 0$ 的时候出现倍周期分叉等丰富的状态，而 $\lambda < 0$ 时系统是灭绝的。我修改了代码中 λ 的增减方向，发现也出现了这样相反的情况。

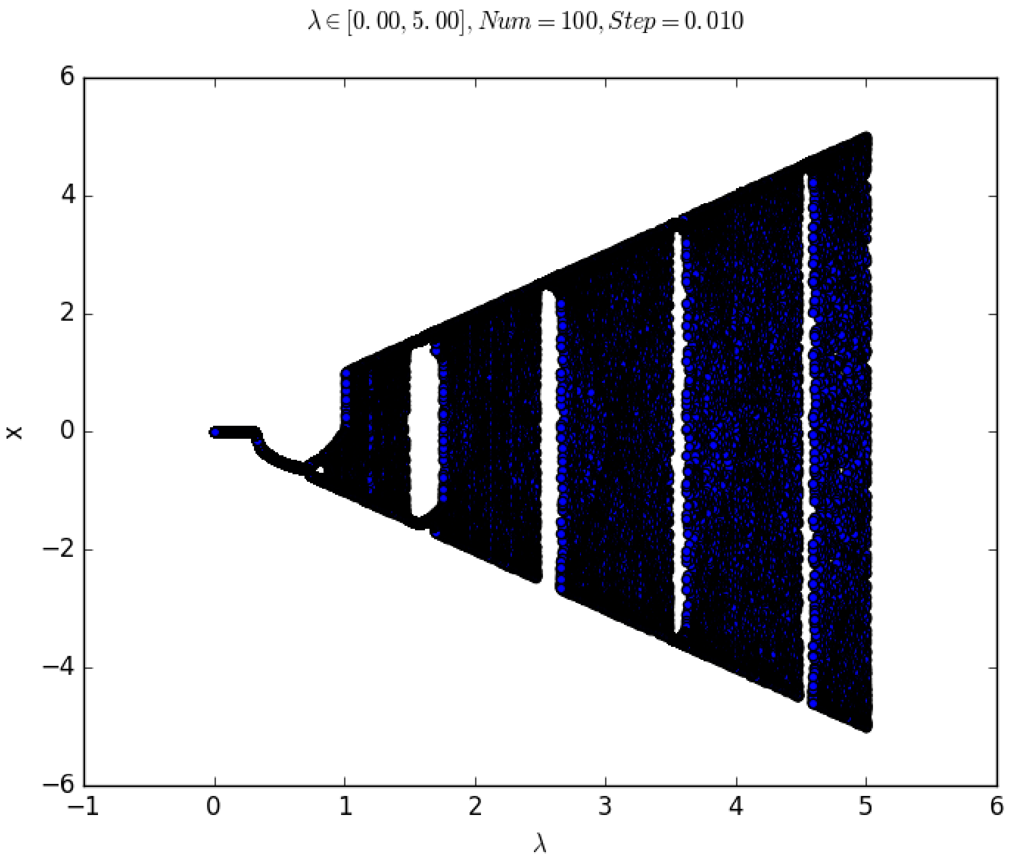
```
for (double lambda = Min; lambda < Max; lambda = lambda+Step) {
/*修改前为chaos.c*/
```

```
for (double lambda = Max; lambda > Min; lambda = lambda-Step) {
/*修改后为feigenbaum.c*/
```

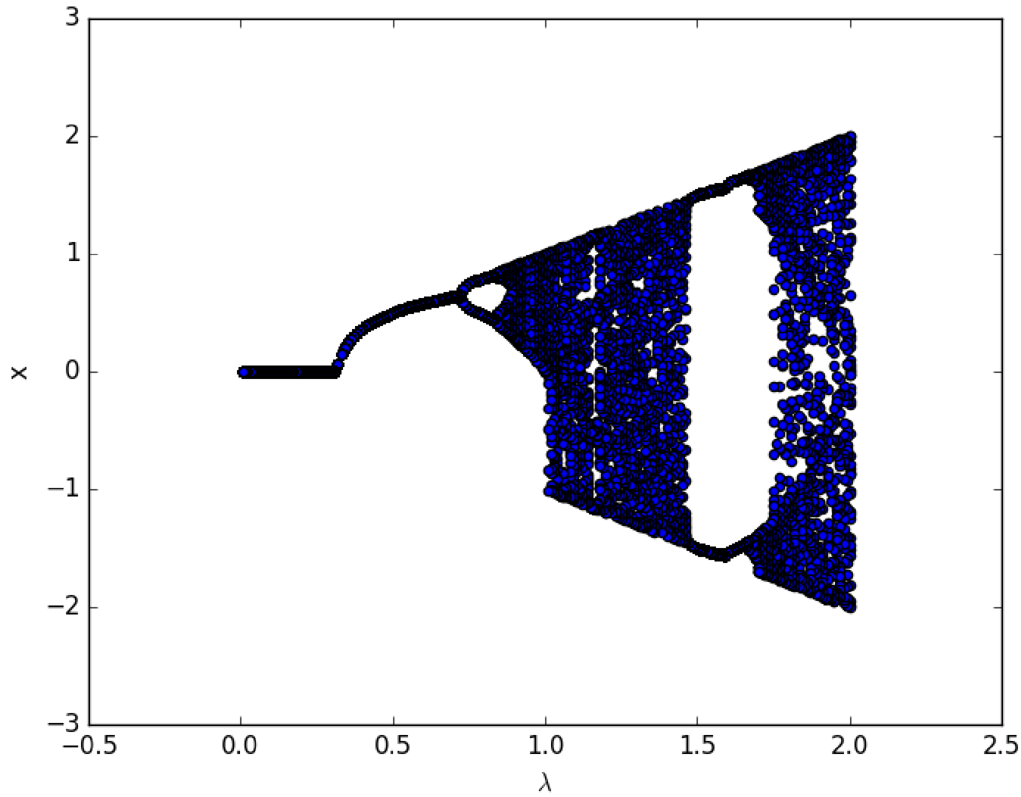
执行以下三行命令，就可以得到计算结果。

```
$ gcc feigenbaum.c -o feigenbaum -lm  # 编译 feigenbaum.c 程序
$ ./feigenbaum > data                 # 在当前目录下执行feigenbaum程序，并将结果输出到 data 文件
$ python plot.py                      # 执行绘图程序，将生成的图片保存为 pdf 和 png 格式
```

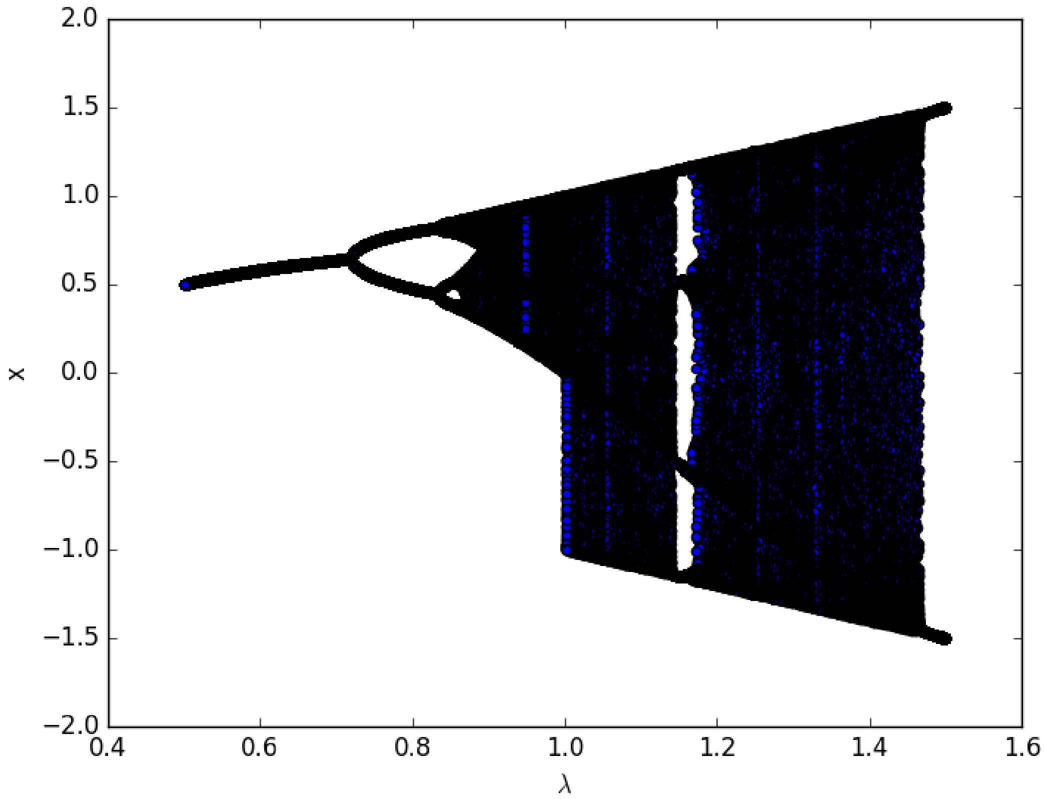

$\lambda \in [-100.00, 100.00], Num = 10, Step = 1.000$
x
λ

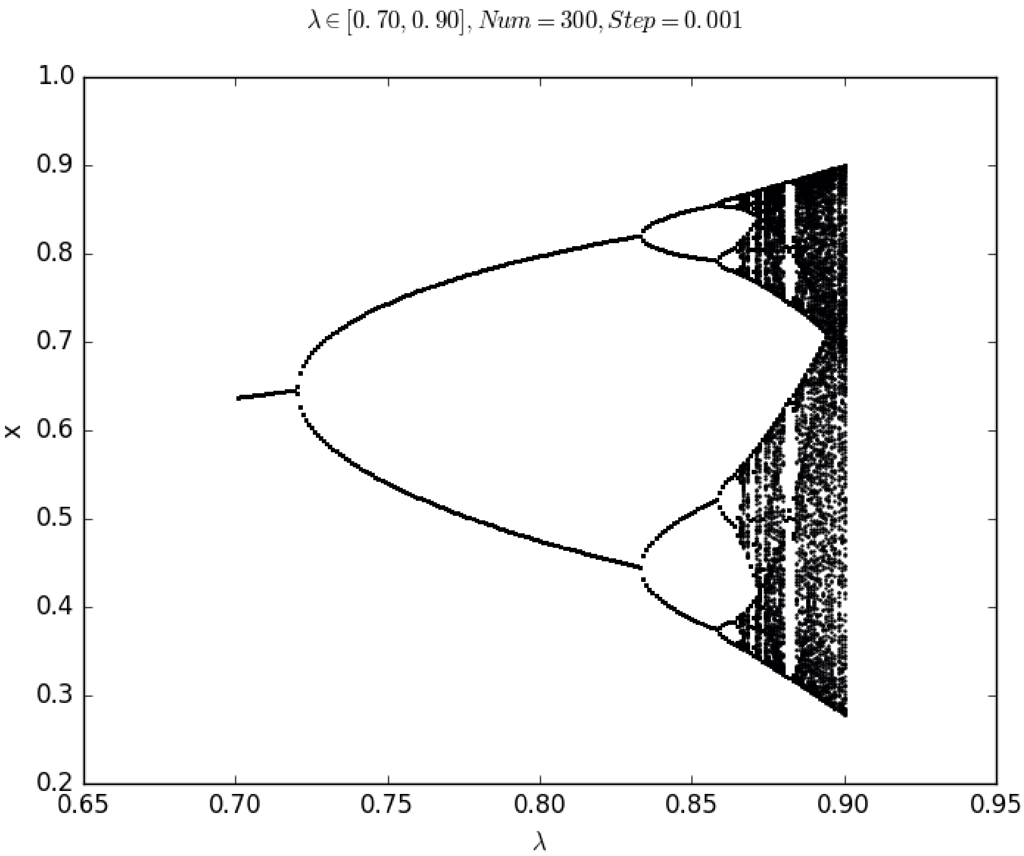

$\lambda \in [0.00, 5.00], Num = 100, Step = 0.010$
x
λ

$\lambda \in [0.00, 2.00], Num = 50, Step = 0.010$



$\lambda \in [0.50, 1.50], Num = 100, Step = 0.001$

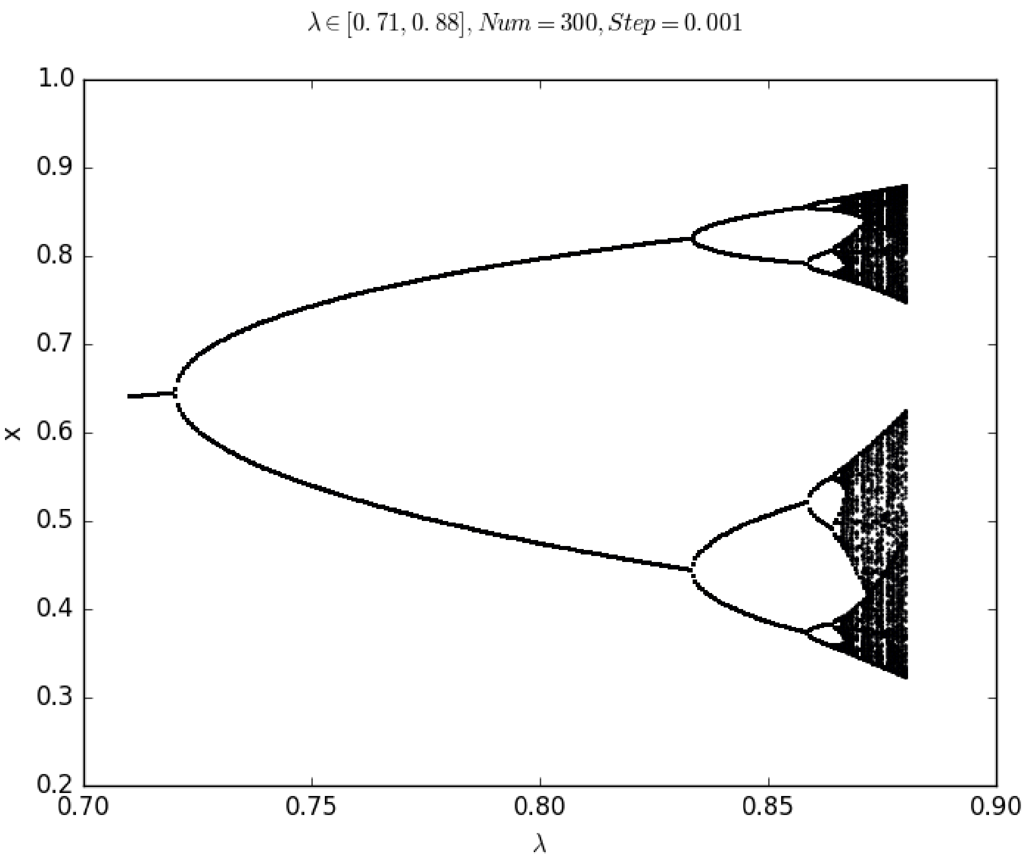




λ 在正值时系统倍周期分叉到混沌，再出现周期窗口，这些过程和之前负区间的状态很相似。但为什么会出现这种情况，还不清楚。

5 倍周期分叉点、Feigenbaum 常数

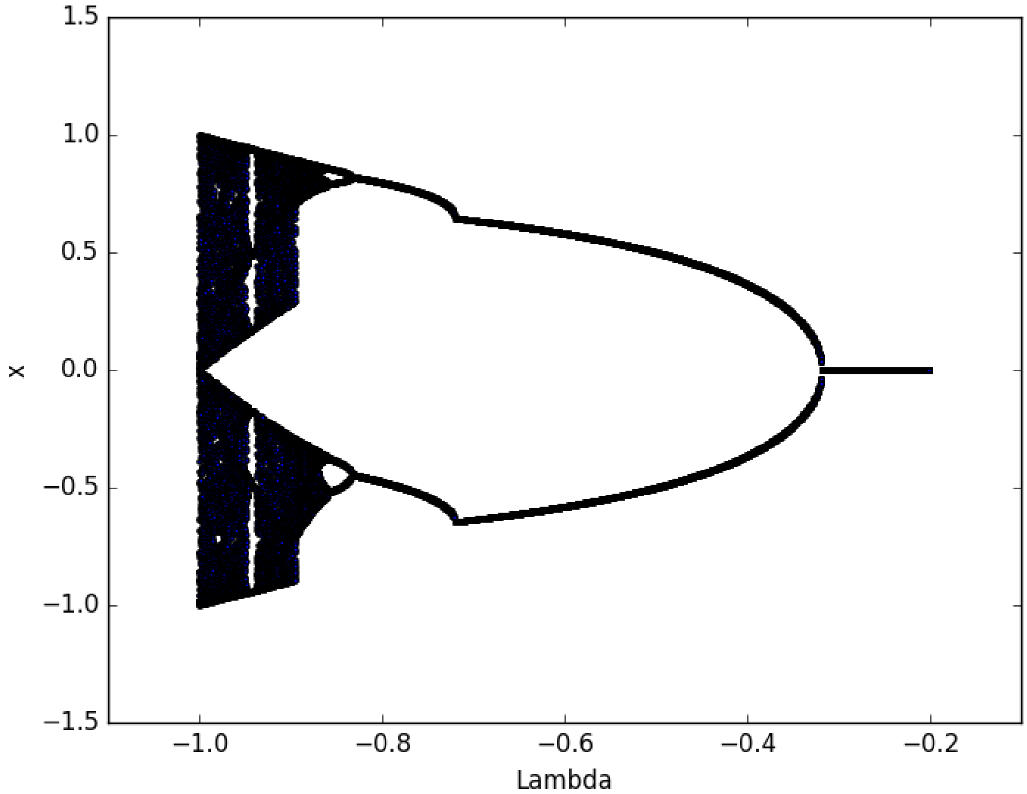
先计算下图这种情况的倍周期分叉点，得Feigenbaum常数为4.666666



m	分叉情况	分叉值	$\frac{\lambda_m-\lambda_{m-1}}{\lambda_{m+1}-\lambda_m}$
1	$1 \to 2$	0.719956	
2	$2 \to 4$	0.833264	4.470801
3	$4 \to 8$	0.858608	4.629041
4	$8 \to 16$	0.864083	4.655612
5	$16 \to 32$	0.865259	4.666666
6	$32 \to 64$	0.865511	4.666666
7	$64 \to 128$	0.865565	

另一种情况，得到的 Feigenbaum 常数为 0.866618，与前一种情况并不相同。

$$\lambda \in [-1.00,\ -0.20], Num = 100, Step = 0.001$$



m	分叉情况	分叉值	$\frac{\lambda_m-\lambda_{m-1}}{\lambda_{m+1}-\lambda_m}$
1	$1 \to 2$	-0.318303	
2	$2 \to 4$	-0.833264	1.433026
3	$4 \to 8$	-0.858608	0.887938
4	$8 \to 16$	-0.864084	0.870412
5	$16 \to 32$	-0.865260	0.866618
6	$32 \to 64$	-0.865511	