01 : Chaos

许传奇 PB16021546

1 题目

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。

2 原理与算法

2.1 原理

2.1.1 迭代法

如果一个物理量的表达式中含有该物理量本身：

$$x = f(x) \tag{1}$$

求解这个物理量时，通常采用数学上的迭代法。

2.1.2 系统状态与参数的关系

当进行迭代时，系统状态会随着参数λ的取值变化。一般而言，λ变化时，系统会有四个主要的阶段，分别是：

1. 绝灭：任意一个初始点，迭代最终得到的结果都是0；
2. 定态：除0外的任意一个初始点，迭代最终结果都是x^{λ}；
3. 倍周期分叉：除0和$x_{n+1} = f(x_n)$与$x_{n+1} = x_n$的交点外，迭代的最终结果都是$x_1, x_2, ..., x_k$；
4. 混沌：除0和$x_{n+1} = f(x_n)$与$x_{n+1} = x_n$的交点外，迭代的最终结果都是互不重复的。

2.1.3 描述混沌的参数

我们用Feigenbaum常数来描述混沌。
Feigenbaum常数有两种：

1. 横轴方向：用λ_m代表第m个分叉点，则$lambda_m$按以下的几何级数（幂函数）收敛到λ_∞：

$$\lambda_\infty - \lambda_m = A\delta^{-m} \tag{2}$$

其中，A是依赖于迭代函数的常数，而δ是不依赖于迭代函数的普适常数。

$$\delta = 4.669201609102990671853203 8...$$

2. 纵轴方向：d_n为迭代函数图像与$x_{n+1} = \frac{1}{2}$的第$n+1$个交点处的λ值对应分叉的纵轴距离，则有关系:

$$\frac{d_m}{d_{m+1}} \to \alpha \qquad (m \to \infty) \tag{3}$$

$$\alpha = 2.502907875095892822283902 87...$$

2.2 算法

2.2.1 宏定义

```
#define X0 0.5
#define LAMBDA0 0.6
#define LAMBDA_LIMIT 0.9
#define STEP 0.0001
#define N 100000
#define ITERATION 10000
#define NUM 100
#define ERROR 0.00001
```

X0为x的初值
LAMBDA0为λ起始值
LAMBDA_LIMIT为λ终值
STEP为λ步长值
N为保存lambda的数组长度，要大于 λ取值范围 / step
ITERATION为初始的函数迭代次数
NUM为ITERATION次后后开始的迭代次数
ERROR为两个double类型数据大小误差

2.2.2 数据结构

首先定义一个线性表sq，sq用来保存一个λ下迭代的x值：

```
typedef struct sq {
double x;
sq *next = NULL;
} sq;
```

其中，x用来保存每次迭代的x值，next指向下一个列表。

然后，设置三个数组。value、unique_value与Feigenbaum。

value中每个元素为指向sq类型的指针，为链表的头指针。第i个元素为保存$\lambda = LAMBDA0 + i * STEP$迭代ITERATION次后再迭代NUM次的NUM个x值的链表的头指针。

unique_value中每个元素为指向sq类型的指针，为链表的头指针。第i个元素为保存$\lambda = LAMBDA0 + i * STEP$迭代NUM次中不相等的x值的链表的头指针

Feigenbaum中每个元素为double型，保存倍周期分叉时的λ。

2.2.3 流程

1. 迭代计算：使用两个for循环。第一个for循环遍历value[i]，i与λ有对应关系$\lambda = LAMBDA0 + i * STEP$。第二个for循环计算迭代的x值，并将每一个迭代出的x值插入到value[i]对应的链表尾部。
2. 保存结果：将value数组中对应的λ与x值输入到文件data.dat中
3. 计算Feigenbaum常数：遍历value数组，将每个链表中不重复的x保存到unique_value数组中。

   unique_value[i]的i与λ同样有对应关系$\lambda = LAMBDA0 + i * STEP$。将unique_value数组中对应的$\lambda$与x值输入到文件unique value.txt中。

   随后遍历unique_value数组，计算unique_value[i]链表中有count个x，count为倍周期分叉数。
4. 保存结果：将unique_value数组中出现新周期时对应的λ与count值输入到文件Feigenbaum.txt中，同时将倍周期分叉时的λ保存到Feigenbaum数组中。
5. 打印Feigenbaum常数
6. 画图：使用plot.py进行画图

3 源文件使用说明

打开01Chaos.exe执行文件，输入λ起始值、终止值、步长和迭代次数，随后等待计算机计算。

计算完毕后，将得到三个文件：

1. data.dat：λ与x保存的文件，用以使用plot.py进行画图。
2. unique value.txt：λ和与其对应的不重复的x保存的文件，用来检查程序正确性。源代码中生成这个文件的代码被注释掉了，需要的时候删除注释即可。
3. Feigenbaum.txt：λ和与其对应的周期数保存的文件，用来检查程序的正确性。

同时，命令行内将会打印出计算的Feigenbaum常数结果。

打开plot.py，编译并运行，即可得到图片。

也可以修改程序，不需要手动输入参数，只需要在01Chaos.cpp文件中修改宏定义的值即可。

注意：

1. 范围不能设置太大，步长不能设置太小，否则会导致数据保存的问题，需要手动在源代码里设置数组长度与相应参数。
2. Feigenbaum常数的计算需要有定态、倍周期与混沌状态，而且λ要是正值，否则计算会出错，另外周期数小时较精确，周期数大的误差很大。可以在Feigenbaum.txt里查看正确的周期数。

4 计算结果及具体分析

4.1 x与λ的关系图

4.1.1 $\lambda \in [-10.00, 10.00]$

取$\lambda \in [-10.00, 10.00], step = 0.010$，$\lambda \in [-10.00, 0.00], step = 0.010$，$\lambda \in [0.00, 10.00], step = 0.010$三种情况，分别如图1，图2，图3所示。

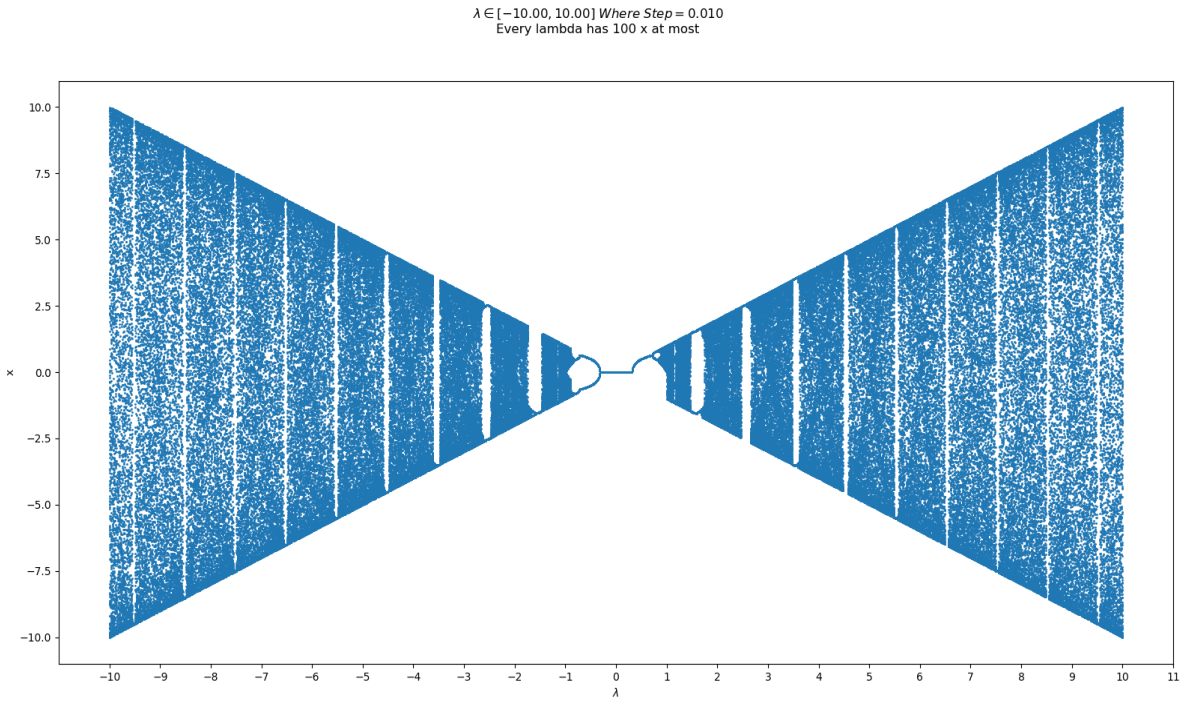

图 1: $\lambda \in [-10.00, 10.00], step = 0.010$

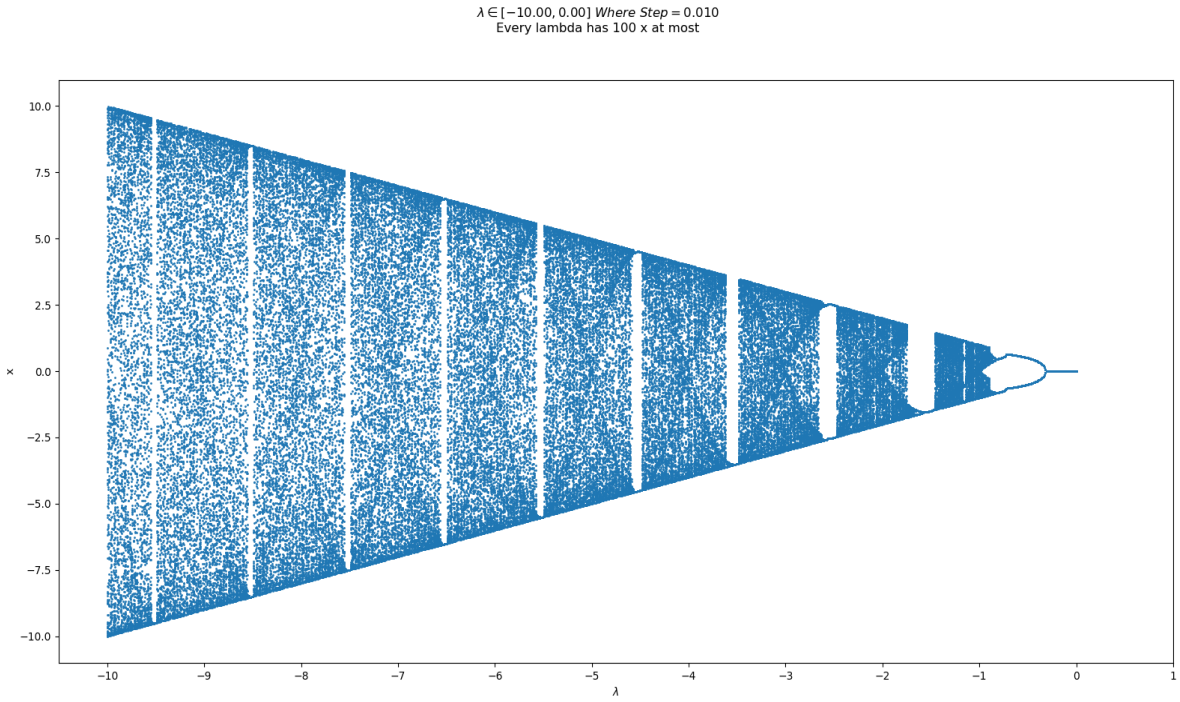

图 2: $\lambda \in [-10.00, 0.00], step = 0.010$

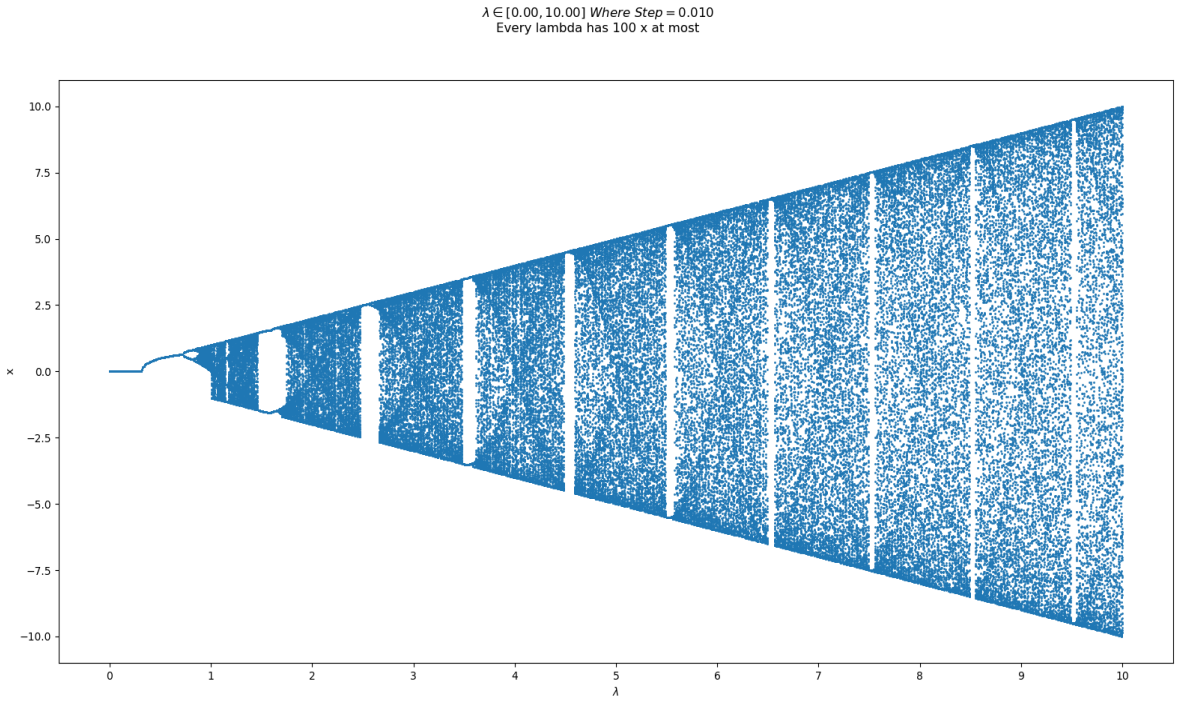

图 3: $\lambda \in [0.00, 10.00], step = 0.010$

可以看到，λ从0开始，往负方向出现了绝灭、倍周期和混沌状态。λ往正方向增大，出现了绝灭、定

态、倍周期分叉和混沌四种状态。

4.1.2 $\lambda \in [-2.00, 2.00]$

取$\lambda \in [-2.00, 2.00], step = 0.001$，$\lambda \in [-2.00, 0.00], step = 0.001$，$\lambda \in [0.00, 2.00], step = 0.001$三种情况，分别如图4，图5，图6所示。

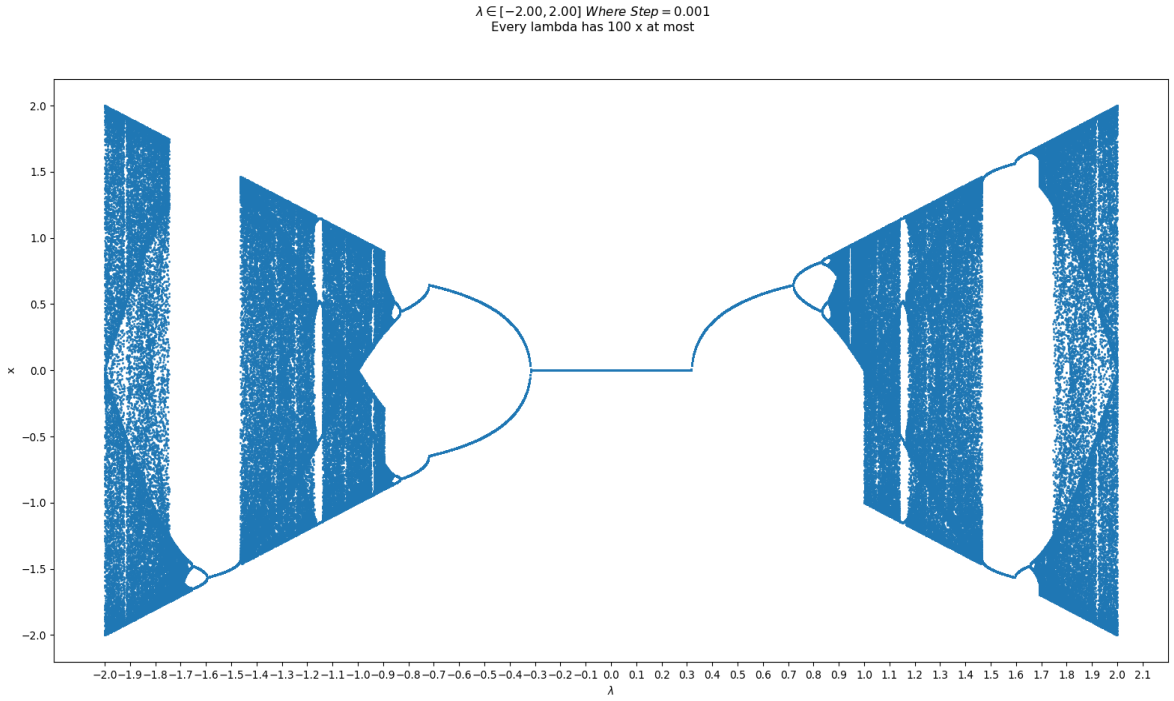

图 4: $\lambda \in [-2.00, 2.00], step = 0.001$

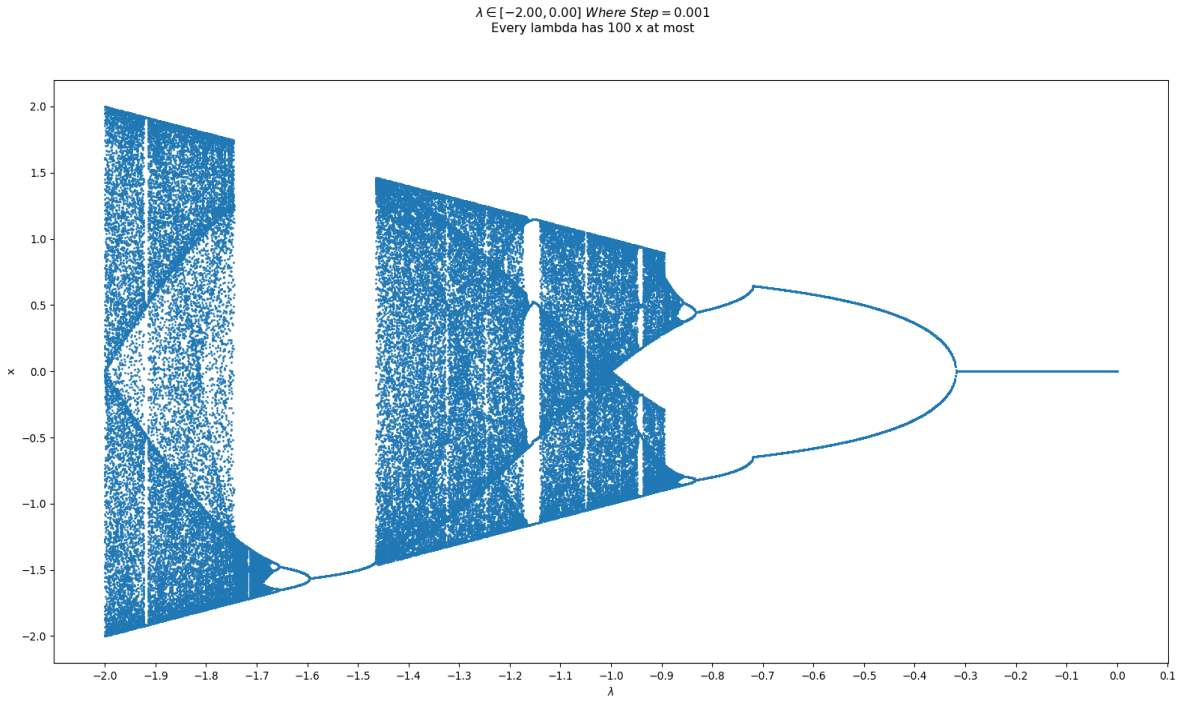

图 5: $\lambda \in [-2.00, 0.00], step = 0.001$

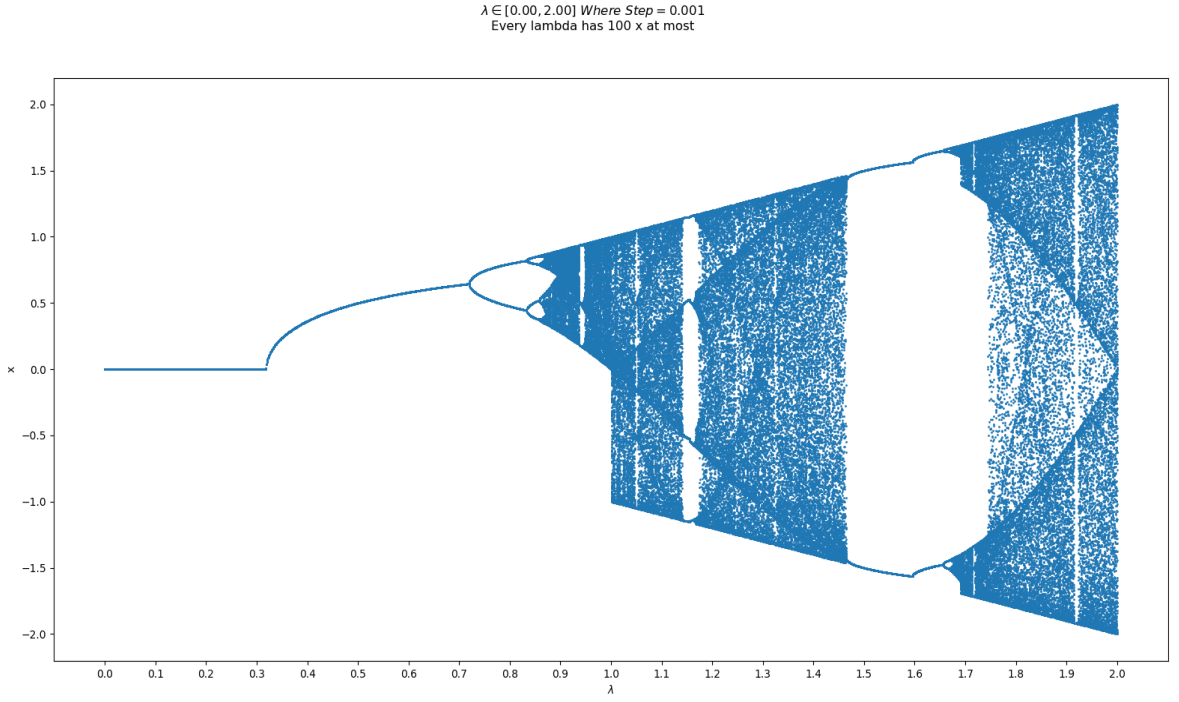

图 6: $\lambda \in [0.00, 2.00], step = 0.001$

可以看出明显的四种状态，且第一次混沌后会接着有类似于二倍周期的取值段，接着再倍周期分叉，

最后又进入混沌状态。

4.1.3 $\lambda \in [-1.10, -0.20]$

取$\lambda \in [-1.10, -0.20], step = 0.00005$,如图7所示。

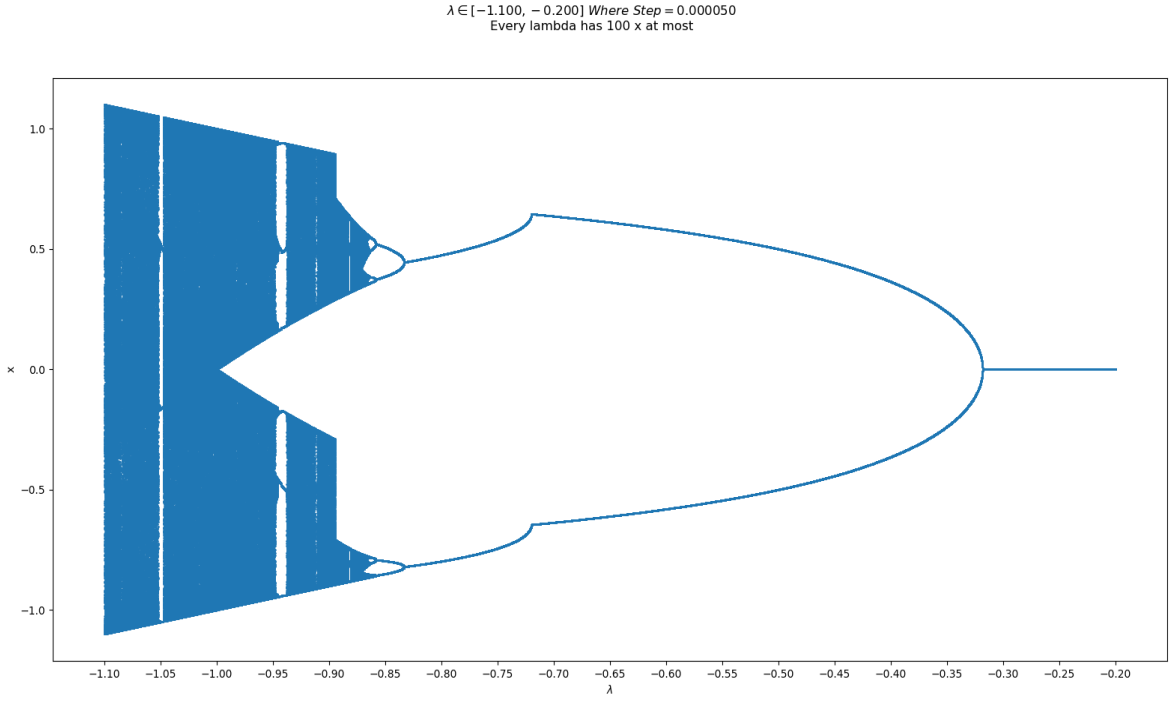

图 7: $\lambda \in [-1.10, -0.20], step = 0.001$

该图比较清晰地展示了λ沿负向的三种状态。

4.1.4 $\lambda \in [0.60, 0.90]$

取$\lambda \in [0.60, 0.90], step = 0.00005$如图8所示。

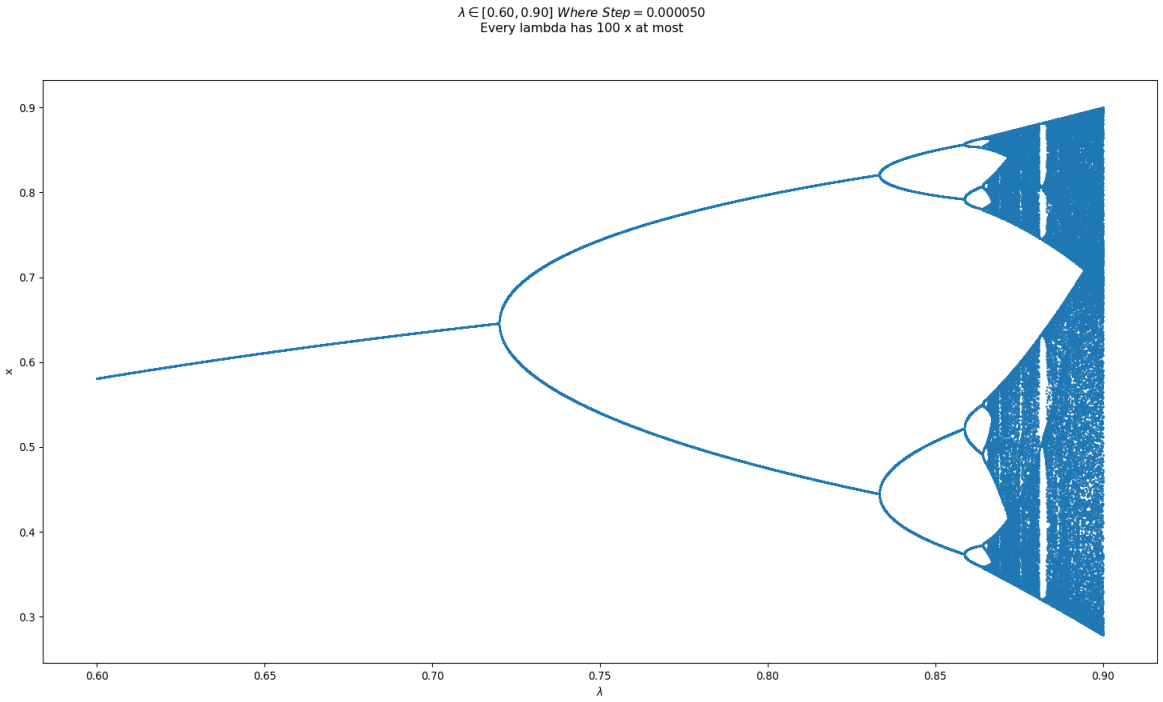

图 8: $\lambda \in [0.60, 0.90], step = 0.00005$

该图比较清晰地展示了λ定态到倍周期分叉再到混沌，等下将用此图计算Feigenbaum常数。

4.2 Feigenbaum常数

我们取$\lambda \in [0.60, 0.90]$时的图进行分析。
我们首先取$step = 0.00001$，得到如图9所示数据。

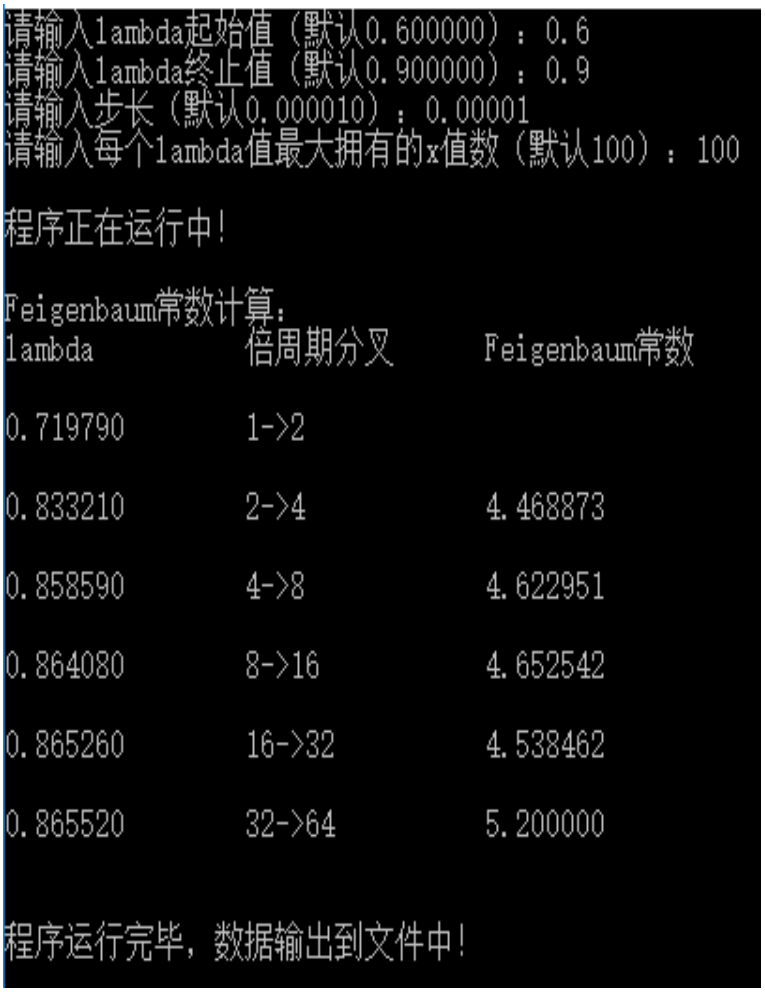

图 9: step=0.00001时计算的Feigenbaum常数

可以看出，当周期较小时Feigenbaum常数计算较精确。但周期数较大时，计算很不精确。我们再将步长减小，取$step = 0.000005$，得到如图10所示数据。

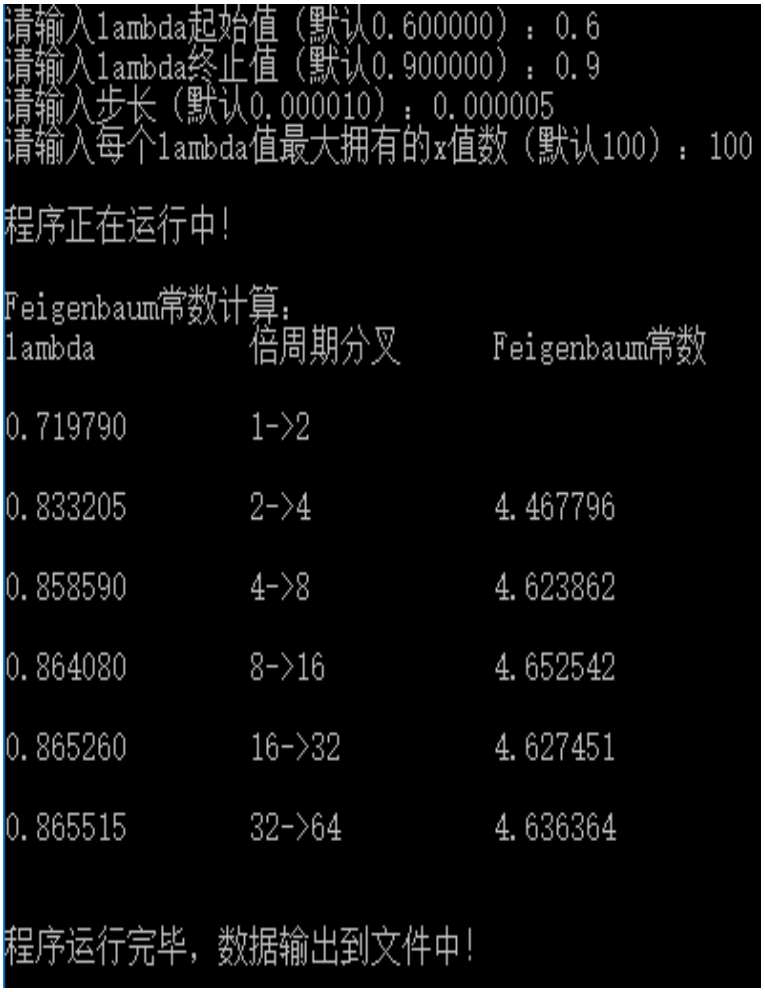

图 10: step=0.000005时计算的Feigenbaum常数

可以看到，当周期大时，Feigenbaum常数计算的比步长为0.00001时精确很多。

这是因为，倍周期分叉周期越大时，两个λ相邻越近。很快两个λ之间的距离就会接近甚至小于步长，这样就导致很大的误差。我们还可以继续减小步长以得到更精确的Feigenbaum常数值，不过由于程序数据保存的问题，继续减小步长将导致内存溢出，因为我们到这为止。

5 讨论

5.1 原理的讨论

从图片中可以看出：

当$\lambda \in [0, 0.318310]$时，系统处于绝灭状态；

当$\lambda \in [0.318310, 0.719790]$时，系统处于定态状态；

当$\lambda \in [0.719790, 0.833205]$时，系统处于周期为2的状态；

当$\lambda \in [0.833205, 0.858590]$时，系统处于周期为4的状态；

……

当λ在接近0.866时，系统进入混沌状态。

我们运用迭代法求解物理量时，希望能够使$x_{n+1} = f(x_n)$和$x_n = f(x_{n+1})$同时成立，这样我们才能求解出所需的物理量。但由于倍周期与混沌状态的存在，并不是对于每个参数的值我们都可以利用迭代法来求解物理量。而且根据结果来看，供我们运用迭代法的区间（绝灭与定态）并不大。因此，我们在运用迭代法时，应该事先考虑这些因素。

计算Feigenbaum常数时，我们也可以看到，当周期变大时，相邻的两个λ相差很小。也就是说，当倍周期分叉出现时，系统周期将随λ变化很快，马上就进入到了混沌状态。这个特性也给Feigenbaum常数的计算带来很大的困难，当周期很大时，要求的步长必须很小，精度必须很高，才能精确计算Feigenbaum常数。

5.2 算法的讨论

由于本人的计算机知识并不是很丰富，下面的讨论可能会欠妥，源代码的编写也肯定会有很多需要改进的地方，希望以后学习中能够不断完善。

5.2.1 数据结构的选取

在本次作业中，我选择用来保存数据的数据结构是链表，同时把链表的头指针装在一维数组里。这样做的目的是想让num（迭代得到的x的个数）可以自由选取，而不必一开始就设置二维数组限制了num的取值。除此之外，也可以选择让这些链表的头指针保存在一个链表中而不是一个数组中，这样λ的范围和步长的选取也是自由的。我不这样做的目的是为了求解当前的λ方便，而不必多设置一个变量来计走过的结点数，也可以快速查找需要的λ对应的x。

当然，这样做的缺点就是运算效率会变低，且不能直接运用已有的数组相关的函数。而且数据中没有保存λ的值，求取λ需要运用关系$\lambda = LAMBDA0 + i * STEP$。

5.2.2 文件的保存

在本次作业中，我将value数组中的数据保存在文件data.dat中，而不是unique_value数组中的数据。这样会导致保存数据的文件变大，而且画图的效率会变低。这样做的原因，将在下面提到，主要是double型变量相等判断的问题。

另外保存数据的unique value.txt和Feigenbaum.txt可以用来方便查看程序的结果，来判断是否出错和需要改进。也可以通过Feigenbaum.txt来手动计算Feigenbaum常数，因为程序源代码中Feigenbaum常数的倍周期数是直接用2的指数计算的，而不是直接读取的周期数。

5.2.3 Feigenbaum常数的计算

在计算Feigenbaum常数时，我们先要除去每个λ迭代产生的相同的x值。由于double型变量存储的误差和迭代过程中本身x_i与x_j就可能存在差距，我们必须要设置一个数值很小的变量error，用两个double型变量的差的绝对值与error进行比较，来判断两个double型变量是否相等。这样做时在周期较小时没什么问题，但当周期很大时，特别是混沌状态时，两个不同的x之间的差可能很小。

由于我的源代码设计的时，将value数组化简为unique_value数组时，当value数组中出现数据与第一个x相等时，就停止保存。上面所述的问题可能会导致提前结束保存，使周期数出错。另外，因为程序源代码中Feigenbaum常数的倍周期数是直接用2的指数计算的，而不是直接读取的周期数；而且周期只

计算到64处。因此正确的周期数需要在Feigenbaum.txt中查看。不过仍需注意，周期很大时，周期数出错概率非常大。因此我舍去了周期数较大的情况。

此外，倍周期分叉周期越大时，两个λ相邻越近。很快两个λ之间的距离就会接近甚至小于步长，这样就导致很大的误差。我们还可以继续减小步长以得到更精确的Feigenbaum常数值，不过由于程序数据保存的问题，继续减小步长将导致内存溢出，因为我们到这为止。

5.3 结果的讨论

本次实验完成的较好，系统对应的四种状态都能够画出来，Feigenbaum常数虽然只能计算周期比较小的时候的值，而且误差较大，但仍然是可行的。

在观察图示的时候，我们可以看到，混沌状态出现后，仍然会出现很大片空白的地方，对应于倍周期分叉状态。混沌状态与倍周期分叉状态交替出现，是一个很神奇的现象。不过由于数学基础的原因，我不能对此给出其他的解释。

此外，可以发现图示中，会出现类似于三角函数的震荡现象，即有一条与三角函数类似的点很密集的线，在图上表示为颜色较深的线。与同学讨论后得知，这条曲线成为关键值曲线(Critical Value Line)，与代数曲线有着很深的联系 [1]。

参考文献

[1] Richard D Neidinger and R John Annen. The road to chaos is flled with polynomial curves. *The American mathematical monthly*, 103(8):640–653, 1996.