《计算方法》课程实验报告

刘铭宸
软件工程 2003 班
U202010783

2022 年 5 月 16 日

目录

1 实际问题 **2**

2 问题求解 **2**

2.1 几何法求解 2

2.1.1 二分法 2

2.1.2 弦截法 4

2.1.3 Steffensen 方法 5

2.1.4 比较分析 7

2.2 迭代法求解 7

2.2.1 Picard 迭代法 7

2.2.2 Aitken 加速迭代法（含导数） 10

2.2.3 Aitken 加速迭代法（不含导数） 11

2.2.4 Newton 迭代法 13

2.2.5 比较分析 14

3 实验结论 **15**

4 致谢 **16**

5 参考资料 **16**

1 实际问题

研究发现, 学生的学习成绩和其所在地区的人均年收入（地区收入）之间存在正相关性。有关机构在对 1998 年美国加利福尼亚州的五年级学生进行抽样调查后发现, 测试成绩和地区收入的相关系数大约为 0.71, 由此建立了负指数生长回归函数。根据加利福尼亚数据集 [1], 可以得知学生的测试成绩和地区收入的负指数模型为:

$$TestScore = 703.2[1 - e^{-0.0552(I+34.0)}] \tag{1}$$

其中 TestScore 表示测试成绩（分）, 参数 I 表示地区收入（千美元）。

根据该模型, 控制其他因素不变, 试问所在地区的人均年收入为多少的学生更有可能在考试中能够得到 700 分?

2 问题求解

该实际问题归结为求解非线性方程 $703.2[1 - e^{-0.0552(I+34.0)}] = 700$ 的根。下面我们将用多种数值解法求解此非线性方程, 并对结果进行分析。

2.1 几何法求解

根据迭代法的要求, 令 $f(I) = 703.2[1 - e^{-0.0552(I+34.0)}] - 700$, 经过初步计算可以知道 [50,90] 为该方程的一个隔离区间, 故在此我们将实验区间定为 [50,90], 允许误差大小定为 0.0001 千美元, 对二分法、弦截法和 Steffensen 方法进行试验, 并对三种方法进行比较分析。

2.1.1 二分法

首先我们可以根据预设精度用公式:

$$k > \frac{ln(b-a) - ln(2\xi)}{ln2} \tag{2}$$

计算出二分的最少次数, 其中 k 为二分次数, b 和 a 分别为实验区间的上界和下界, ξ 为预设精度, 在本问题中为 10^{-4}。

可得满足精度要求的最少二分次数为 18 次。根据课本 2.1 节关于二分法的介绍，编写代码如下：

```
1  public class Dichotomy {
2      public static double f(double I){
3          return 703.2*(1-Math.exp(-0.0552*(I+34.0)))-700;
4      }
5
6      public static void main(String[] args) {
7          double a = 50, b = 90;
8          while(b-a >= 0.0001){
9              double c = (a+b)/2;
10             if(f(c)==0)
11                 break;
12             else if(f(a)*f(c)<0)
13                 b = c;
14             else
15                 a = c;
16             System.out.println(b-a);
17         }
18         System.out.println("I = "+(a+b)/2);
19     }
20 }
```

计算结果如下表：

迭代次数	1	2	3	4	5
误差	20.000000	10.000000	5.000000	2.500000	1.250000
迭代次数	6	7	8	9	10
误差	0.625000	0.312500	0.156250	0.078125	0.039063
迭代次数	11	12	13	14	15
误差	0.019531	0.009766	0.004883	0.002441	0.001221
迭代次数	16	17	18	19	
误差	0.000610	0.000305	0.000153	0.000076	
I	63.690071				

表 1: 二分法的计算结果

结果分析:

结果显示, 二分法总共迭代了 19 次, 最终求得 I=63.690071, 即所在地区的人均年收入达到约 63690 美元时, 学生更有可能获得 700 分。最终迭代误差为 7.62939453125e-5, 显然达到了实验预设误差要求。

2.1.2 弦截法

弦截法的一般迭代表达式为:

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} f(x_k), \quad k = 1, 2, \ldots \tag{3}$$

在本问题中, 取初始迭代值 $x_0 = 50, x_1 = 90$

根据课本 2.2 节关于弦截法的介绍, 编写代码如下:

```
1  public class Secant {
2      public static double f(double I){
3          return 703.2*(1-Math.exp(-0.0552*(I+34.0))
               )-700;
4      }
5
6      public static void main(String[] args) {
7          double x0 = 50, x1 = 90;
```

```
8                double x2 = x1-(x1-x0)*f(x1)/(f(x1)-f(x0))
                     ;
9                double tol = 0.0001;
10               while(Math.abs(x2-x1)>=tol){
11                    x0 = x1;
12                    x1 = x2;
13                    x2 = x1-(x1-x0)*f(x1)/(f(x1)-f(x0));
14                    System.out.println(Math.abs(x2-x1));
15               }
16               System.out.println("I = "+x2);
17           }
18       }
```

计算结果如下表:

迭代次数	1	2	3	4
误差	20.551021	13.239997	2.079753	0.812050
迭代次数	5	6	7	
误差	0.061945	0.001264	0.000002	
I	63.690046			

表 2: 弦截法的计算结果

结果分析:

结果显示, 弦截法总共迭代了 7 次, 最终求得 I=63.690046, 即所在地区的人均年收入达到约 63690 美元时, 学生更有可能获得 700 分。最终迭代误差为 2.119659875177149e-6, 显然达到了实验预设误差要求。

2.1.3 Steffensen 方法

Steffensen 方法的一般迭代表达式为:

$$x_{k+1} = x_k - \frac{f^2(x_k)}{f(x_k) - f(x_k - f(x_k))}, \quad k = 1, 2, \ldots \tag{4}$$

在本问题中，取初始迭代值 $x_0 = 50$

根据课本 2.2 节关于 Steffensen 方法的介绍，编写代码如下：

```
1  public class Steffensen {
2      public static double f(double I){
3          return 703.2*(1-Math.exp(-0.0552*(I+34.0))
               )-700;
4      }
5
6      public static void main(String[] args) {
7          double x0 = 50;
8          double x1 = x0-f(x0)*f(x0)/(f(x0)-f(x0-f(
               x0)));
9          while(Math.abs(x1-x0)>=0.0001){
10             x0 = x1;
11             x1 = x0-f(x0)*f(x0)/(f(x0)-f(x0-f(x0))
                   );
12             System.out.println(Math.abs(x1-x0));
13         }
14         System.out.println("I = "+x1);
15     }
16 }
```

计算结果如下表：

迭代次数	1	2
误差	2.890429	0.201702
迭代次数	3	4
误差	0.000929	1.960079e-8
I	63.690046	

表 3: 弦截法的计算结果

结果分析：

结果显示,Steffensen 方法总共迭代了 4 次, 最终求得 I=63.690046, 即所在地区的人均年收入达到约 63690 美元时, 学生更有可能获得 700 分。最终迭代误差为 1.9600790324147965e-8, 显然达到了实验预设误差要求。

2.1.4 比较分析

将三种方法每次迭代产生的误差大小绘制得到如图 1 所示。从图中可以看到，三种方法最终的误差均趋于零，即我们可以认为三种方法都是收敛的。另外，初值取 50 时的 Steffensen 方法收敛速度最快，仅用 4 次就达到了预设精度要求；弦截法次之，二分法最慢，分别用了 7 次和 19 次迭代才达到预设精度要求。但是，作为一种单步法，Steffensen 方法的收敛速度和初值的选取有很大关系。当将初值 x_0 取为 98 时，得到的结果如图二所示。可以看到，此时 Steffensen 方法的收敛速度与效果低于弦截法。实验中得到的结果也是符合课本上的理论分析的。

2.2 迭代法求解

根据迭代法的要求, 可将原方程等价写为 $I = 703.2[1-e^{-0.0552(I+34.0)}]-700+I$ 的形式。下面我们将用多种数值解法求解此非线性方程，并对结果进行分析。

2.2.1 Picard 迭代法

构造 Picard 迭代法的迭代格式：

$$I_{k+1} = 703.2[1 - e^{-0.0552(I_k+34.0)}] - 700 + I_k, \quad k = 0, 1, 2, \ldots \tag{5}$$

首先我们可以根据课本 2.3 节介绍的 Picard 迭代法的局部收敛性判据对其敛散性进行分析：

(1) 对迭代函数 $\phi(I)$ 求导得 $\phi'(I) = 38.81664e^{-0.0552(I+34.0)} + 1$，可知导函数在任意区间内都连续可微。

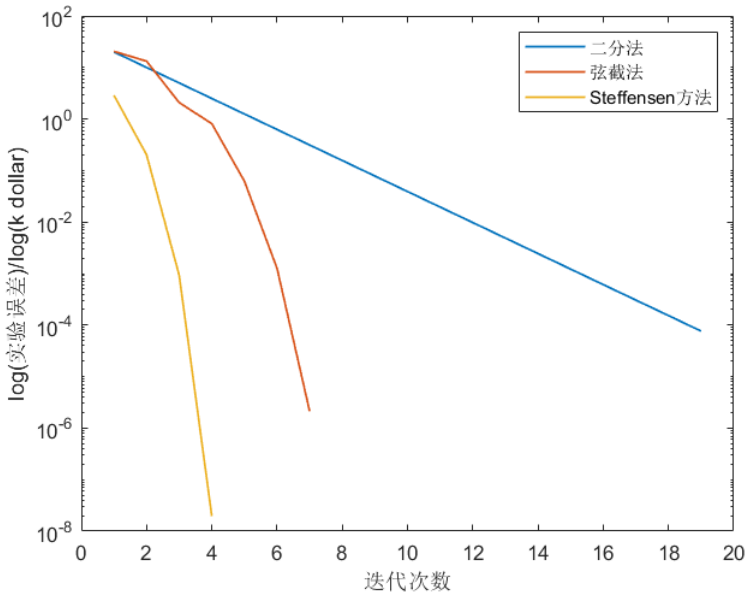

图 1: 几何法的实验误差与迭代次数对比

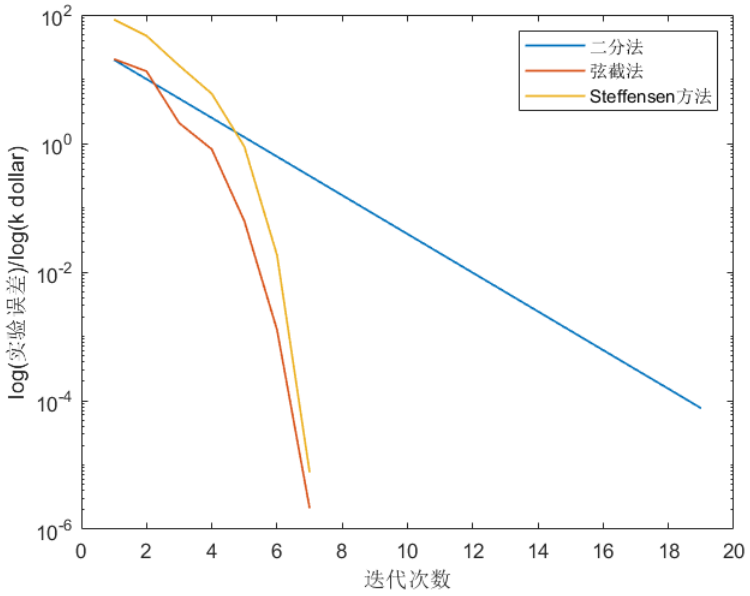

图 2: 初值取 98 时的 Steffensen 方法与二分法、弦截法的对比

(2) 对于任意 I，都有 $|\phi'(I)| > 1$ 成立。

所以该迭代格式不满足局部收敛性判据，同理可以得到此格式亦不满足全局收敛性判据。但由于此二者皆为迭代格式收敛的充分不必要条件，我们仍无法确定其发散。

下面通过编程进行分析。编写代码如下：

```
1  public class Picard {
2      public static double fai(double I){
3          return 703.2*(1-Math.exp(-0.0552*(I+34.0))
               )-700+I;
4      }
5
6      public static void main(String[] args) {
7          double x0 = 50;
8          double x1 = fai(x0);
9          while(Math.abs(x1-x0)>=0.0001){
10             System.out.println(Math.abs(x1-x0));
11             x0 = x1;
12             x1 = fai(x0);
13         }
14         System.out.println(Math.abs(x1-x0));
15         System.out.println("I = "+x1);
16     }
17 }
```

计算结果如下表：

结果分析：

可以看到实验误差趋向于无穷，可知该迭代格式是发散的，对其使用 Picard 迭代法无法求出正确结果。

迭代次数	1	2	3	4	5
误差	3.613060	5.116839	7.831246	13.796654	33.200840
迭代次数	6	7	8	9	
误差	224.329122	54311455	Infinity	NaN	
I	-Infinity				

表 4: Picard 迭代法的计算结果

2.2.2 Aitken 加速迭代法（含导数）

含导数的 Aitken 加速迭代格式为：

$$x_{k+1} = (1-l)^{-1}[\phi(x_k) - lx_k], \quad k = 0, 1, \ldots \tag{6}$$

其中 $l = \phi'(x^*) \approx \phi'(\frac{50+90}{2}) = \phi'(70)$，初始迭代值 $x_0 = 50$

根据课本 2.4 节关于 Aitken 加速迭代法的介绍，编写代码如下：

```
1  public class Aitken_derivative {
2      public static double fai(double I){
3          return 703.2*(1-Math.exp(-0.0552*(I+34.0))
               )-700+I;
4      }
5
6      public static double dfai(double I){
7          return 38.81664*Math.exp(-0.0552*(I+34.0))
               +1;
8      }
9
10     public static void main(String[] args) {
11         double l = dfai((50+90)/2.0);
12         double x0 = 50;
13         double x1 = (fai(x0)-l*x0)/(1-l);
14         while(Math.abs(x1-x0)>=0.0001){
15             System.out.println(Math.abs(x1-x0));
```

```
16                  x0 = x1;
17                  x1 = (fai(x0)-l*x0)/(1-l);
18              }
19              System.out.println(Math.abs(x1-x0));
20              System.out.println("I = "+x1);
21          }
22      }
```

计算结果如下表:

迭代次数	1	2	3	4	5
误差	28.977104	14.627290	0.917859	0.368246	0.155581
迭代次数	6	7	8	9	10
误差	0.064433	0.026914	0.011203	0.004670	0.001945
迭代次数	11	12	13	14	
误差	0.000811	0.000338	0.000141	0.000059	
I	63.690063				

表 5: 含导数的 Aitken 加速迭代法的计算结果

结果分析:

结果显示, 不含导数的 Aitken 加速迭代法总共迭代了 14 次, 最终求得 I=63.690063, 即所在地区的人均年收入达到约 63690 美元时, 学生更有可能获得 700 分。最终迭代误差为 5.864194860549787e-5, 显然达到了实验预设误差要求。

2.2.3 Aitken 加速迭代法（不含导数）

不含导数的 Aitken 加速迭代格式为:

$$\begin{cases} \tilde{x}_{k+1} = \phi(x_k), \quad \hat{x}_{k+1} = \phi(\tilde{x}_{k+1}) \\ x_{k+1} = \hat{x}_{k+1} - \frac{(\hat{x}_{k+1} - \tilde{x}_{k+1})^2}{\hat{x}_{k+1} - 2\tilde{x}_{k+1} + x_k} \end{cases} \tag{7}$$

取初始迭代值 $x_0 = 50$

根据课本 2.4 节关于 Aitken 加速迭代法的介绍，编写代码如下:

```
1  public class Aitken_noderivative {
2      public static double fai(double I){
3          return 703.2*(1-Math.exp(-0.0552*(I+34.0)))-700+I;
4      }
5
6      public static void main(String[] args) {
7          double x0 = 50;
8          double x1 = fai(x0);
9          double x2 = fai(x1);
10         double x3 = x2-(x2-x1)*(x2-x1)/(x2-2*x1+x0);
11         while(Math.abs(x3-x0)>=0.0001){
12             System.out.println(Math.abs(x3-x0));
13             x0 = x3;
14             x1 = fai(x0);
15             x2 = fai(x1);
16             x3 = x2-(x2-x1)*(x2-x1)/(x2-2*x1+x0);
17         }
18         System.out.println(Math.abs(x3-x0));
19         System.out.println("I = "+x3);
20     }
21 }
```

计算结果如下表:

结果分析:

结果显示, 不含导数的 Aitken 加速迭代法总共迭代了 5 次, 最终求得 I=63.690046, 即所在地区的人均年收入达到约 63690 美元时, 学生更有可能获得 700 分。最终迭代误差为 1.0853231756868809e-5, 显然达到了实验预设误差要求。

迭代次数	1	2	3
误差	8.680931	4.254293	0.736539
迭代次数	4	5	
误差	0.018273	0.000011	
I	63.690046		

表 6: 不含导数的 Aitken 加速迭代法的计算结果

2.2.4 Newton 迭代法

Newton 迭代法的一般表达式为:

$$x_k = x_{k-1} - \frac{f(x_k)}{f'(x_k)}, \quad k = 1, 2, \ldots \tag{8}$$

取初始迭代值 $x_0 = 50$

根据课本 2.5 节关于 Newton 迭代法的介绍，编写代码如下:

```
1  public class Newton {
2      public static double f(double I){
3          return 703.2*(1-Math.exp(-0.0552*(I+34.0))
               )-700;
4      }
5
6      public static double df(double I){
7          return 38.81664*Math.exp(-0.0552*(I+34.0))
               ;
8      }
9
10     public static void main(String[] args) {
11         double x0 = 50;
12         double x1 = x0-f(x0)/df(x0);
13         while(Math.abs(x1-x0)>=0.0001){
14             System.out.println(Math.abs(x1-x0));
15             x0 = x1;
```

```
16                    x1 = x0-f(x0)/df(x0);
17                }
18                System.out.println(Math.abs(x1-x0));
19                System.out.println("I = "+x1);
20            }
21        }
```

计算结果如下表:

迭代次数	1	2	3
误差	9.607135	3.655517	0.422392
迭代次数	4	5	
误差	0.005001	0.0000007	
I	63.690046		

表 7: Newton 迭代法的计算结果

结果分析:

结果显示,Newton 迭代法总共迭代了 5 次, 最终求得 I=63.690046, 即所在地区的人均年收入达到约 63690 美元时, 学生更有可能获得 700 分。最终迭代误差为 6.905348826080626e-7, 显然达到了实验预设误差要求。

2.2.5 比较分析

将四种方法每次迭代产生的误差大小绘制得到如图 3 所示。从图中可以看到，Aitken 加速迭代法和 Newton 迭代法最终的误差均趋于零，即我们可以认为这三种方法都是收敛的；而 Picard 迭代法最终趋向于无穷，对于本问题中构造的迭代格式是发散的。根据课本 2.7 节关于迭代法的收敛阶的介绍，我们可以计算出这三种方法的收敛阶。对于含导数的 Aitken 加速迭代法，其迭代函数为 $(1-l)^{-1}[\phi(x_k)-lx_k]$，记为 $\Phi(x)$，可得:

$$\Phi^{'}(x^*) = (1-l)^{-1}[\phi^{'}(x^*)-l] = 0$$

$$\Phi^{''}(x^*) = (1-l)^{-1}\Phi^{''}(x^*) = -38.81664 * 0.0552e^{-0.0552(x^*+34)} < 0$$

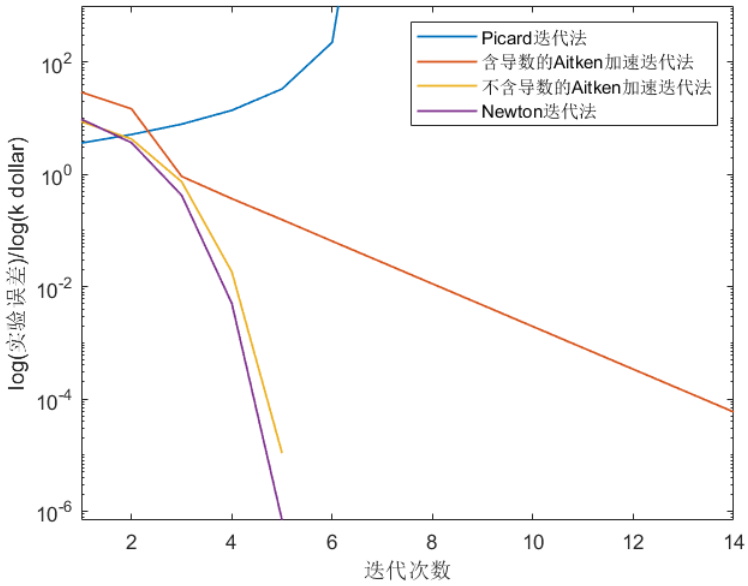

图 3: 迭代法的实验误差与迭代次数对比

由于 $\Phi(x)$ 在 x^* 处连续可微，可以得出含导数的 Aitken 加速迭代法是线性收敛的；同时也可证明在本问题的迭代格式下，含导数的 Aitken 加速迭代法是平方收敛的；又因为 $f'(x^*) \neq 0$，所以 x^* 是单根，故 Newton 迭代法也是平方收敛的。这也与实验得到的结果相符合。从图中可以看到，含导数的 Aitken 加速迭代法收敛速度最慢，而 Newton 迭代法要比不含导数的 Aitken 加速迭代法收敛速度略快。

3 实验结论

上述实验结果表明，当学生所在地区的人均年收入达到约 63690 美元时, 学生更有可能在测试中获得 700 分。

本实验在研究中存在一定的局限性。首先，学生所在地区的人均年收入并不是影响学生考试成绩的唯一原因，其他诸如学生的学习态度、学习能力等是更显著的影响因素。在本问题中为了研究这两者的回归模型对其它因素进行了控制，将无法避免的影响放进了误差项之中，这样可能会引起遗漏变量偏差，造成回归模型的不准确；其次，被放进误差项当中的一些

影响因素和本实验所研究的因变量存在相关性，例如学生所在地区的收入在一定程度上影响着学校的教育水平和教师的授课水平，这种相关性会引发内生性的问题，影响回归模型的一致性和有效性。所以我认为应建立多变量的回归模型，在控制其他变量不变的情况下再用非线性方程的数值解法对其进行分析预测，这样得到的结果会更符合实际情况。

4 致谢

感谢覃婷婷老师的教学！在大学的课程学习中很少能碰见对学生如此认真负责并且授课清楚明了的老师。在计算方法这门课上，老师更注重于教授学生公式、定理的来源与推导的思维过程，从解决问题的角度出发向学生们介绍前人是如何一步步将这些算法构造出来的，让我感受到数学家们也不是一下子就明白，他们也是有一个从无到有的思维过程。在以后，那些公式定理会被我逐渐忘记，但一些解决问题的思维方式却会一直根种在脑海。另外还要感谢老师与助教对作业的认真批改！我在大学中从未见到对学生作业每一道题每一处细微的错误都能详细批改的老师；每周的作业反馈也让我们及时了解自己哪里掌握的还比较薄弱，方便学生进行查缺补漏。最后，希望覃老师的课能越办越好!

5 参考资料

[1] 加利福尼亚数据集

https://wps.pearsoned.com/aw_stock_ie_3/178/45691/11696965.cw/index.html

[2] 张诚坚，何南忠，覃婷婷. 计算方法（第二版）[M]. 北京：高等教育出版社，2021.

[3]（美）斯托克 (Stock.J.H)，（美）沃森 (Waston,M.W.) 著. 沈根祥，孙燕译. 计量经济学（第三版）. 上海：格致出版社：上海人民出版社，2012.