第二章 非线性方程求根 编程实验

孔瑞阳　计科91　2019010175

第二章上机题2：

问题:

编程实现牛顿下山法。 要求:

(1) 设定合适的下山因子初始值 λ_0 及迭代判停准则;

(2) 下山因子 λ 用逐次折半法更新;

(3) 打印每个迭代步的最终值及近似解;

(4) 请用其他方法 (如 fzero 函数) 验证结果, 并考虑采用牛顿下山法的效果。

用所编程序求解:

(1) $x^3 - x - 1 = 0$, 取 $x_0 = 0.6$ 。

(2) $-x^3 + 5x = 0$, 取 $x_0 = 1.35$ 。

解题思路:

将书中的伪代码翻译成 C++。

因为在伪代码中实际上已经包含了 $\lambda = 1$ 的计算，所以从 $\lambda_0 = 1$ 开始迭代。

判停标准：$|x - x'| < 10^{-8}$ 且 $|f(x)| < 10^{-8}$ 。

实验结果:

对于 $x^3 - x - 1 = 0, x_0 = 0.6$，迭代过程为：

```
lambda: 0.03125000, approximate solution: 1.14062500
lambda: 1.00000000, approximate solution: 1.36681366
lambda: 1.00000000, approximate solution: 1.32627980
lambda: 1.00000000, approximate solution: 1.32472023
lambda: 1.00000000, approximate solution: 1.32471796
```

对于 $-x^3 + 5x = 0, x_0 = 1.35$ ，迭代过程为：

```
lambda: 0.12500000, approximate solution: 2.49695856
lambda: 1.00000000, approximate solution: 2.27197621
lambda: 1.00000000, approximate solution: 2.23690171
lambda: 1.00000000, approximate solution: 2.23606844
lambda: 1.00000000, approximate solution: 2.23606798
```

使用以下 MATLAB 代码求出这两个方程的解：

```
format long
x = roots([1 0 -1 -1])
x = roots([-1 0 5 0])
```

结果为：

```
x =

  1.324717957244745 + 0.000000000000000i
 -0.662358978622373 + 0.562279512062301i
 -0.662358978622373 - 0.562279512062301i

x =

                   0
   2.236067977499790
  -2.236067977499790
```

可以发现，牛顿下山法均求出了靠近 x_0 的一组解（小数点后八位正确）。

结果分析:

对于以上两个方程，λ 均只在第一步迭代过程中不等于 1，在后面几步均和普通的牛顿迭代法一样。

说明牛顿下山法可以较好地解决牛顿迭代法前几步过程中 x 离 x^* 太远导致的可能出现的发散问题，并在之后也和牛顿迭代法具有几乎同样的效率。

第二章上机题3：

问题:

利用 2. 6. 3 节给出的 fzerotx 程序，编程求第一类的零阶贝塞尔函数 $J_0(x)$ 的零点。

试求 $J_0(x)$ 的前 10 个正的零点, 并绘出函数曲线和零点的位置.

解题思路:

将书中伪代码翻译成Python，使用 matplotlib 中的 pyplot 包进行绘图。

比较 $J_0*(x)$ 和 $J_0(x+0.01)$ 的正负号，如果不一样则说明有一个零点。

经过测试算得第 10 个零点介于 $[30,31)$ 之间，所以将 x 的范围设为 $[0,31]$ 。

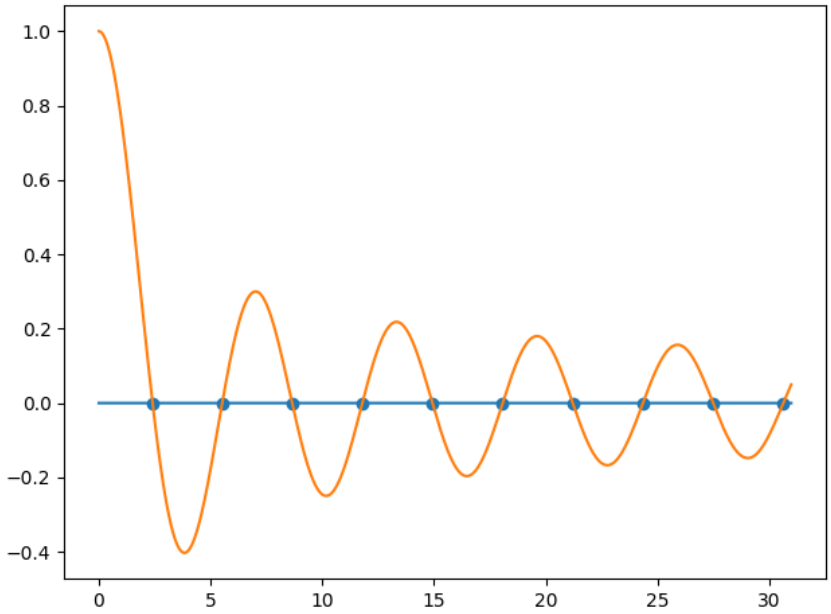
将 x 轴、$J_0(x)$ 和求得的零点都画在笛卡尔坐标系中，比较零点的位置是否准确。

实验结果:

运行结果为：

```
ZERO 1: x = 2.4048255577
ZERO 2: x = 5.5200781103
ZERO 3: x = 8.6537279132
ZERO 4: x = 11.7915344391
ZERO 5: x = 14.9309177085
ZERO 6: x = 18.0710639679
ZERO 7: x = 21.2116366299
ZERO 8: x = 24.3524715308
ZERO 9: x = 27.4934791321
ZERO 10: x = 30.6346064685
```

绘图结果为：



结果分析:

通过图像可以发现，通过这个算法求出的零点非常准确。