Numerical Analysis Programming Assignment #4 非线性方程求根

陈宇轩 PB16060738

问题描述

1. 分别编写用Newton迭代和弦截法求根的通用程序。
2. 分别用如上程序求根 $f(x)=2x^4+24x^3+61x^2-16x+1=0$

   其中，Newton迭代分别取初值$x_0=0$ 和 $x_0=1$；弦截法的初值分别取为 $x_0=0, x_1=0.1$以及 $x_0=0.5, x_1=1.0$;
3. 取误差限 ϵ 为1.0e-10，即当$|f(x_k)|<\epsilon$ 时，停止迭代。

   将计算结果列成表格，要求给出初值、每步的迭代结果，以及最终的迭代结果（包括迭代步数）；比较或分析两种计算方法的优劣。

计算结果

Newton 迭代结果1 ($x_0=0$)

迭代步数k	x_k	$f(x_k)$
k=0	0.0000000000e+00	1.0000000000e+00
k=1	6.2500000000e-02	2.4417114258e-01
k=2	9.2675144823e-02	6.0357821710e-02
k=3	1.0750916023e-01	1.4994760152e-02
k=4	1.1485323376e-01	3.7248898748e-03
k=5	1.1848368152e-01	9.1626064336e-04
k=6	1.2024260677e-01	2.1577268802e-04
k=7	1.2102581790e-01	4.2847681852e-05
k=8	1.2128383271e-01	4.6530959359e-06
k=9	1.2131962667e-01	8.9569062833e-08
k=10	1.2132034327e-01	3.5900726836e-11

Newton 迭代结果2 ($x_0=1$)

迭代步数k	x_k	$f(x_k)$
k=0	1.0000000000e+00	7.2000000000e+01
k=1	6.1290322581e-01	1.9916142676e+01
k=2	3.8571317721e-01	5.3253351976e+00
k=3	2.5960364887e-01	1.3863606793e+00
k=4	1.9251296856e-01	3.5450987947e-01
k=5	1.5779817659e-01	8.9686549723e-02
k=6	1.4012814469e-01	2.2547538196e-02
k=7	1.3122111065e-01	5.6408367721e-03
k=8	1.2676835045e-01	1.3986621658e-03
k=9	1.2457983507e-01	3.3654327128e-04
k=10	1.2356510397e-01	7.2212049151e-05
k=11	1.2318374768e-01	1.0190518405e-05
k=12	1.2310877106e-01	3.9378256023e-07
k=13	1.2310563114e-01	6.9058347929e-10
k=14	1.2310562562e-01	2.1094237468e-15

弦截法迭代结果1 ($x_0 = 0, x_1 = 0.1$)

迭代步数k	x_k	$f(x_k)$
k=0	0.0000000000e+00	1.0000000000e+00
k=1	1.0000000000e-01	3.4200000000e-02
k=2	1.0354110582e-01	2.4179518330e-02
k=3	1.1208582811e-01	7.0954731540e-03
k=4	1.1563468594e-01	2.9655182228e-03
k=5	1.1818294682e-01	1.0792226046e-03
k=6	1.1964090533e-01	4.0681640637e-04
k=7	1.2052299333e-01	1.4401732385e-04
k=8	1.2100638906e-01	4.6100547817e-05
k=9	1.2123397833e-01	1.1307750276e-05
k=10	1.2130794544e-01	1.5591680875e-06
k=11	1.2131977559e-01	7.0957434706e-08
k=12	1.2132033965e-01	4.8875359315e-10
k=13	1.2132034356e-01	1.5543122345e-13

弦截法迭代结果2 ($x_0 = 0.5, x_1 = 1.0$)

迭代步数k	x_k	$f(x_k)$
k=0	5.0000000000e-01	1.1375000000e+01
k=1	1.0000000000e+00	7.2000000000e+01
k=2	4.0618556701e-01	6.2280271014e+00
k=3	3.4995656522e-01	3.9299549639e+00
k=4	2.5379881582e-01	1.2691171507e+00
k=5	2.0793527302e-01	5.3000859083e-01
k=6	1.7504690856e-01	1.9898234938e-01
k=7	1.5527746672e-01	7.7353635526e-02
k=8	1.4270446360e-01	2.9543153285e-02
k=9	1.3493532719e-01	1.1322065118e-02
k=10	1.3010780715e-01	4.3180216959e-03
k=11	1.2713162110e-01	1.6401064889e-03
k=12	1.2530883671e-01	6.1561581910e-04
k=13	1.2421352668e-01	2.2446674520e-04
k=14	1.2358496664e-01	7.6000812034e-05
k=15	1.2326320210e-01	2.1431886963e-05
k=16	1.2313682942e-01	3.9677813739e-06
k=17	1.2310811800e-01	3.1191157945e-07
k=18	1.2310566840e-01	5.3467786865e-09
k=19	1.2310562568e-01	7.4584782794e-12

结果分析

1. 根据计算所得的结果，可以看出初始值越接近根，迭代的次数越少。
2. 对于两种计算方法，在有相同初始值的情况下，对于较小的误差限Newton迭代最终迭代的步数更少；在迭代步数较少时，弦截法得到的误差更小。

代码实现(Python)

```
# Newton迭代和弦截法求根的通用程序
import numpy as np
```

```
E = 1e-10


def f(x):
    return 2*x**4+24*x**3+61*x**2-16*x+1


def f1(x):
    return 8*x**3+72*x**2+122*x-16


def Newton_Iteration(x_0):
    x_k = x_0
    k = 0
    while(1):
        print("|  k=%d  |  %.10e  |  %.10e  |" % (k, x_k, f(x_k)))
        if np.fabs(f(x_k)) < E:
            break
        k = k + 1
        x_k = x_k - f(x_k)/f1(x_k)


def Secant_Method(x_0, x_1):
    x_k = x_1
    x_km1 = x_0
    k = 0
    print("|  k=%d  |  %.10e  |  %.10e  |" % (k, x_0, f(x_0)))
    while(1):
        k = k+1
        print("|  k=%d  |  %.10e  |  %.10e  |" % (k, x_k, f(x_k)))
        if np.fabs(f(x_k)) < E:
            break
        d = (x_k-x_km1)/(f(x_k)-f(x_km1))
        x_kp1 = x_k - f(x_k)*d
        x_km1 = x_k
        x_k = x_kp1


if __name__ == "__main__":
    print("Newton迭代: x0=0")
    Newton_Iteration(0)
    print("Newton迭代: x0=1")
    Newton_Iteration(1)
    print("弦截法求根: x0=0, x1=0.1")
    Secant_Method(0, 0.1)
    print("弦截法求根: x0=0.5, x1=1.0")
    Secant_Method(0.5, 1.0)
```