第二章

1. 题目

（1）给定初值 x_0 及容许误差 ε，编制 Newton 法解方程 f(x)=0 根的通用程序。

（2）给定方程 f(x)=x3/3-x=0,易知其有三个根 $x_1^※=-\sqrt{3}$，$x_2^※=0$，$x_3^※=\sqrt{3}$。

①由 Newton 方法的局部收敛性可知存在 $\delta>0$，当 $x_0\in(-\delta，\delta)$，Newton 迭代序列收敛于根 $x_2^※$，试确定尽可能大的 δ；

②试取若干个初始值，观察当 $x_0\in$（-∞，-1），（-1，$-\delta$），（$-\delta$，δ），（δ，1），（1，+∞）时，Newton 序列是否收敛以及收敛于哪一个根。

（3）通过本上机题，你明白了什么？

2. 程序代码

代码一：

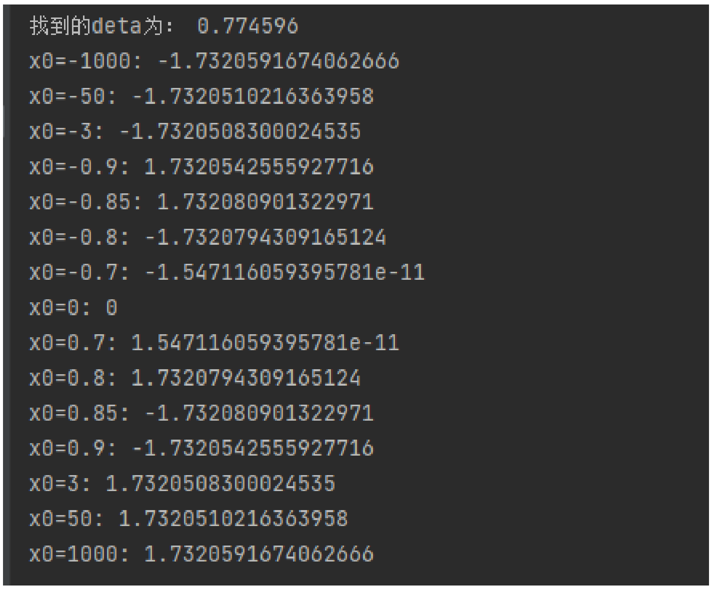
```python
#数值分析第二次上机作业:
#自定义函数f(x)，这里定义f(x)=x^3/3-x
def Newton(x0,c,epsilon):
    def f(x):
        f=x**3/3-x
        return f
    def f_grad(x,c):
        f_grad=x**2-1
        if f_grad==0:
            f_grad=f_grad+c
        return f_grad
    def newton_method(x0,epsilon):
        xi=x0
        while True:
            xi_plus_1=xi-f(xi)/f_grad(xi,c)
            if abs(xi_plus_1-xi)<epsilon:
                break
            #print(xi)
            xi=xi_plus_1
        return xi
    return newton_method(x0,epsilon)
#使用示例:
#x0=1
#c=0.0001
#epsilon=0.0001
#root=Newton(x0,c,epsilon)
#print("方程的根为: ",root)
```

代码二：

```
#数值分析第二次上机题(2)①
from newton import Newton

#x0=1
step=0.000001
c=0.0001
epsilon=0.0001
deta0=0
i=0
while(True):
    i = i + 1
    deta = deta0 + (i-1)*step
    if Newton(deta, c, epsilon)>1:
        break
#root = Newton(x0, c, epsilon)
deta_max=deta-step
# 打印结果
print("找到的deta为: ", deta_max)
print("x0=-1000:",Newton(-1000, c, epsilon))#x0∈(-∞, -1)
print("x0=-50:",Newton(-50, c, epsilon))#x0∈(-∞, -1)
print("x0=-3:",Newton(-3, c, epsilon))#x0∈(-∞, -1)
print("x0=-0.9:",Newton(-0.9, c, epsilon))#x0∈(-1, -deta)
print("x0=-0.85:",Newton(-0.85, c, epsilon))#x0∈(-1, -deta)
print("x0=-0.8:",Newton(-0.8, c, epsilon))#x0∈(-1, -deta)
print("x0=-0.7:",Newton(-0.7, c, epsilon))#x0∈(-deta, deta)
print("x0=0:",Newton(0, c, epsilon))#x0∈(-deta, deta)
print("x0=0.7:",Newton(0.7, c, epsilon))#x0∈(-deta, deta)
print("x0=0.8:",Newton(0.8, c, epsilon))#x0∈(deta, 1)
print("x0=0.85:",Newton(0.85, c, epsilon))#x0∈(deta, 1)
print("x0=0.9:",Newton(0.9, c, epsilon))#x0∈(deta, 1)
print("x0=3:",Newton(3, c, epsilon))#x0∈(1,+∞)
print("x0=50:",Newton(50, c, epsilon))#x0∈(1,+∞)
print("x0=1000:",Newton(1000, c, epsilon))#x0∈(1,+∞)
```

3. 运行结果



```
找到的deta为:  0.774596
x0=-1000: -1.7320591674062666
x0=-50: -1.7320510216363958
x0=-3: -1.7320508300024535
x0=-0.9: 1.7320542555927716
x0=-0.85: 1.732080901322971
x0=-0.8: -1.7320794309165124
x0=-0.7: -1.547116059395781e-11
x0=0: 0
x0=0.7: 1.547116059395781e-11
x0=0.8: 1.7320794309165124
x0=0.85: -1.732080901322971
x0=0.9: -1.7320542555927716
x0=3: 1.7320508300024535
x0=50: 1.7320510216363958
x0=1000: 1.7320591674062666
```

图 1 程序结果

最后能满足使 x 收敛于 0 的最大 deta 值是 0.774596，取不同初值的收敛结果如下表所示

表 1 收敛结果

x0	收敛值	是否收敛
-1000	$-\sqrt{3}$	是
-50	$-\sqrt{3}$	是
-3	$-\sqrt{3}$	是
-0.9	$\sqrt{3}$	是
-0.85	$\sqrt{3}$	是
-0.8	$-\sqrt{3}$	是
-0.7	0	是
0	0	是
0.7	0	是
0.8	$\sqrt{3}$	是
0.85	$-\sqrt{3}$	是
0.9	$-\sqrt{3}$	是
3	$\sqrt{3}$	是
50	$\sqrt{3}$	是
1000	$\sqrt{3}$	是

4.结果分析与上机体会

4.1结果分析：

首先编写了Newton迭代法解方程f(x)=0根的通用程序，可以实现给定初值x0及容许误差epsilon后，迭代出函数的根。

之后编写程序找出最大的能满足使x收敛于0的deta，找到的deta是0.774596，之后在不同的范围内多次取初值进行迭代计算，观察最后x收敛的值，可以得出如下结论：

①当x0∈（$-\infty$，-1）时，最后的根收敛于$-\sqrt{3}$；

②当x0∈（-1，-deta）时，最后的根收敛于$-\sqrt{3}$或$\sqrt{3}$，取x0=-0.9时，收敛于$\sqrt{3}$，取x0=-0.85时，收敛于$-\sqrt{3}$，取x0=-0.8时，收敛于$-\sqrt{3}$；

③当x0∈（-deta，deta）时，最后的根收敛于0；

④当x0∈（deta，1）时，最后的根收敛于$-\sqrt{3}$或$\sqrt{3}$，取x0=0.9时，收敛于$-\sqrt{3}$，取x0=0.85时，收敛于$-\sqrt{3}$，取x0=0. 8时，收敛于$\sqrt{3}$；

⑤当x0∈（1,+∞）时，最后的根收敛于$\sqrt{3}$；

4.2上机体会：

本次上机实验采用了python编写，深刻理解了Newton迭代法的函义。程序运行花费了2.42613秒，需要平衡准确性和算力能耗之间的矛盾。