数值实验报告 I

实验名称	计算方法上机实践				实验时间
姓名	秦浩政 郭凯平 刘桂凡 刘佳鑫	班级	数据 2301	学号	23060302 23060205 23090501 23090501

一、实验目的，内容

实验 2.4 牛顿迭代法

实现牛顿迭代法，要求考虑算法的通用性，考虑算法迭代失败的情况，做出分析。

实验内容：

（1）运用牛顿迭代法的思想，求解方程想 x^3+4x^2-10=0,在区间[1,2]上的根。（考虑算法通用性和迭代失

（2）运用牛顿迭代法的思想，求解方程想 e^x+10x-2=0,在区间[1,2]上的根。（考虑算法通用性和迭代失败的

实验 2.5 割线法与抛物线法

牛顿迭代法改进的一种实验，与牛顿迭代法进行对比。

实验内容：

测试用例：(1)求解方程想 x^3+4x^2-10=0,在区间[1,2]上的根。

(2 求解方程想 e^x+10x-2=0,在区间[1,2]上的根。

分别用割线法与抛物线法求解测试用例，与牛顿迭代法对比各种迭代方法的特点。

二、算法描述

实验 2.4:：牛顿迭代法是一种求解方程 f(x)=0 的数值方法，利用泰勒展开构造迭代公式：$x_{n+1}=x_n-f(x_n)/f'($

实验 2.5：

（1）割线法：解决了求解函数 f 的导数可能比较困难的问题，对牛顿迭代法进行了改进，用割线斜率代替切

代算法：

$X_{k+1}=x_k-f(x_k)(x_k-x_{k-1})/(f(x_k)-f(x_{k-1}))=[x_{k-1}f(x_k)-x_kf(x_{k-1})]/f(x_k)-f(x_{k-1})$

(2) 抛物线法：利用三个已知点构造一条抛物线，取其与 x 轴的交点构造下一次迭代值。

算法为：

$X_{k+1}=x_k+\&_4(x_k-x_{k-1})$

三程序代码

实验 2.4

```python
import math
def new_ton(x0, f, f_, max_iterations, tol):
    x_next = x0
    for i in range(0, max_iterations + 1):
        if f_(x_next) == 0:
            result = "False ， 导数为0，无法继续迭代"
            return result
        x_new = x_next - f(x_next) / f_(x_next)
        ##判断是否结束迭代
        if abs(x_new - x_next) < tol:
            result = [x_new, i+1]
            return result
        x_next = x_new  # 更新x_next的值
    result = "超过最大迭代次数，无法继续迭代，请判断该方法是否收
    return result
max_iterations = 100000
tol = 1e-9
# 测试用例1
def f1(x):
    result = x ** 3 + 4 * x ** 2 - 10
    return result
def f_1(x):
    result = 3 * x ** 2 + 8 * x
    return result
# 测试用例2
def f2(x):
    result = math.exp(x) + 10 * x - 2
    return result
def f_2(x):
    result = math.exp(x) + 10
    return result
result1 = new_ton(2, f1, f_1, max_iterations, tol)
result2 = new_ton(0, f2, f_2, max_iterations, tol)
print("牛顿迭代法方程的根和迭代次数：", result1)
print("牛顿迭代法方程的根和迭代次数：", result2)
```

实验 2.5

(1)割线法：

```python
import math
def gexian(x0, x1, f, max_iterations, tol):
    x_k0 = x0
    x_k1 = x1
    for i in range(0, max_iterations + 1):
        x_new = x_k1 - f(x_k1) * (x_k1 - x_k0) / (f(x_k1
        ##判断是否结束迭代
        if abs(x_new - x_k1) < tol:
            result = [x_k1, i+1]
            return result
        x_k0 = x_k1
        x_k1 = x_new  # 更新两个迭代点的值
    result = "超过最大迭代次数，无法继续迭代，请判断该方法是否收敛"
    return result
max_iterations = 100000
tol = 1e-9
# 测试用例1
def f1(x):
    result = x ** 3 + 4 * x ** 2 - 10
    return result
# 测试用例2
def f2(x):
    result = math.exp(x) + 10 * x - 2
    return result
result1 = gexian(1, 2, f1, max_iterations, tol)
result2 = gexian(0, 2, f2, max_iterations, tol)
print("弦割法方程的根和迭代次数：",result1)
print("弦割法方程的根和迭代次数：",result2)
```

(2)抛物线法：

```python
import math
def sign(x):
    if x > 0:
        return 1
    elif x < 0:
        return -1
    else:
        return 0
def paowuxian(x0, x1, x2, f, max_iterations, tol):
    xk0 = x0
    xk1 = x1
    xk2 = x2
    for i in range(max_iterations):
        lam3 = (xk2 - xk1) / (xk1 - xk0)
        sita3 = lam3 + 1
        a = f(xk0) * lam3 ** 2 - f(xk1) * lam3 * sita3 + f(xk2)
        b = f(xk0) * lam3 ** 2 - f(xk1) * sita3 ** 2 + f(xk2) *
        c = f(xk2) * sita3
        lam4 = -2 * c / (b + sign(b) * math.sqrt(b ** 2 - 4 * a
        x_new = xk2 + lam4 * (xk2 - xk1)##迭代公式
        #判断是否结束迭代
        if abs(x_new - xk2) < tol:
            result = [x_new, i + 1]
            return result
        xk0 = xk1
        xk1 = xk2
        xk2 = x_new#更新迭代点的值
max_iterations = 100000
tol = 1e-9
# 测试用例1
def f1(x):
    result = x ** 3 + 4 * x ** 2 - 10
    return result
# 测试用例2
def f2(x):
    result = math.exp(x) + 10 * x - 2
    return result
result1 = paowuxian( x0: 1, x1: 1.1, x2: 2, f1, max_iterations, t
result2 = paowuxian( x0: 0, x1: 1.5, x2: 2, f2, max_iterations, t
print("抛物线法方程的根和迭代次数：", result1)
```

四数值结果

实验 2.4

```
运行:  new_ton ×
C:\Users\wll_114001\AppData\Local\Programs\Python\Python31
牛顿迭代法方程的根和迭代次数:  [1.3652300134140969, 5]
牛顿迭代法方程的根和迭代次数:  [0.090525101307255, 4]

进程已结束,退出代码0
```

实验 2.5

(1)

```
gexian ×
C:\Users\wll_114001\AppData\Local\Programs\Python\Pytho
弦割法方程的根和迭代次数:  [1.3652300134142061, 7]
弦割法方程的根和迭代次数:  [0.09052510109606701, 5]
```

(2)

```
paowuxian ×
C:\Users\wll_114001\AppData\Local\Programs\Python\Python310\p
抛物线法方程的根和迭代次数:  [1.3652300134140969, 5]
抛物线法方程的根和迭代次数:  [0.09052510130725501, 5]

进程已结束,退出代码0
```

五．计算结果分析

实验 2.4

在测试用例 1 中，牛顿迭代法成功找到方程 x^3+4x^2-10=0 的根，迭代次数表明方法收敛良好。然而在测试

迭代过程较快。这表明对不同类型的函数，牛顿迭代法都具有较好的适应性和有效性，能在设定的容差和迭代

中的可靠性。

实验 2.5

（1）弦割法通常对较简单的函数收敛速度较快，根据结果两个方程分别迭代了 7,5 次，数值误差在允许的范围

（2）抛物线法在这两个测试用例中表现出良好的收敛能力，成功找到了方程的根。

六. 计算中出现的问题，解决方法及体会

实验 2.4

问题：在递推的过程中，迭代可能失败

解决办法：在程序中做出判断

体会：在看一个办法是否可行的时候，要从原理上去证明该方法的可行性以及有什么限制条件

实验 2.5

问题：即使选择合理的初始点，但如果根的逼近速度较慢，可能会超过设定的最大迭代次数。

计算过程中涉及平方根。

解决办法：在实际应用中，可以适当调整以达到更好的收敛效果。

谨慎处理以避免复杂的数值问题。

体会：要学会如何判断该方法是否适用，弦割法通常对较简单的函数收敛速度较快，但对于更复杂的或

波动较大的函数，可能需要谨慎选择初始点以确保收敛，抛物线法初值的选取范围更宽泛，而且可以一次求得

方程的一对复根。

教 师 评 语	