**北京科技大学实验报告**

学院： 计算机与通信工程 专业： 计算机科学与技术 班级： 计科184

姓名： 马亮 学号： 41824208 实验日期：2020 年 3月 18日

**实验名称：**求解非线性方程

**实验目的：**编程实现求解非线性方程的相关算法，以更好的掌握知识

**实验仪器：**

编程语言：c++、python

操作系统：ubuntu 18.04LTS

IDE: vscode

C++编译器: g++

python解释器： python3.6.9

**实验原理：**

牛顿迭代法求解非线性方程、弦割法求解非线性方程

**实验内容与步骤：**

**实验要求**：

至少采用两种不同的算法求解ex+3\*x3-2\*x2+x-2=0在[0,1]范围内的一个根，要求两次迭代误差小于10-4

**编程实现（c++）:**

#include<iostream>

#include<cstdlib>

#include<string>

#include <iomanip>

#include<cmath>

using namespace std;

// 定义常量e

double const M\_E = 2.7182818284590452353602874713527;

// 原函数Fx

double f(double x){

    return pow(M\_E, x) + 3 \* pow(x, 3) - 2 \* pow(x, 2) + x - 2;

}

// 导函数

double fp(double x){

    return pow(M\_E, x) + 9 \* pow(x, 2) - 4 \* x + 1;

}

// 牛顿迭代法Gx函数

double gN(double xk){

    return xk - (f(xk) / fp(xk));

}

// 弦割法Gx函数

double gX(double x1, double x0){

    return x1 - (f(x1) \* (x1 - x0)) / (f(x1) - f(x0));

}

// 弦割法求近似解, eps为精度

void XG(double x0, double x1, double eps){

    int index = 1;

    double temp;

    cout << "count" << '\t' << "Xk-1" << "\t\t" << "Xk" << endl;

    cout << index << '\t' << x0 << "\t\t" << x1 << endl;

    // 控制精度和最大迭代次数

    while (fabs(x0 - x1) >= eps && index <= 40)

    {

        index++;

        temp =  x1;

        x1 = gX(x1, x0);

        x0 = temp;

        cout  << index << '\t' << x0 << "\t\t" << x1 << endl;

    }

    // 输出结果

    cout << "The solution satisfying the accuracy requirement is: x" << index << "=" << x1 << endl;

}

// 牛顿法求解方程组

void N(double x0, double eps){

    int index = 0;

    // 初始化xk-1, 避免第一步就跳出

    double xk, xk\_1=x0+eps \* 100;

    xk = x0;

    cout << "count" << '\t' << "Xk" << endl;

    cout << index << '\t' << xk << endl;

    while ( fabs(xk - xk\_1) >= eps && index <= 40 )

    {

        index++;

        xk\_1 = xk;

        xk = gN(xk\_1);

        cout << index << '\t' << xk << endl;

    }

    // 输出结果

    cout << "The solution satisfying the accuracy requirement is: x" << index << "=" << xk << endl;

}

int main()

{

    cout << "String cut method:\n" << endl;

    XG(0.5, 0.4, pow(10,-4));

    cout << setw(100) << setfill('-') << endl;

    cout << "\nNewton method:\n" << endl;

    N(0.7, pow(10, -5));

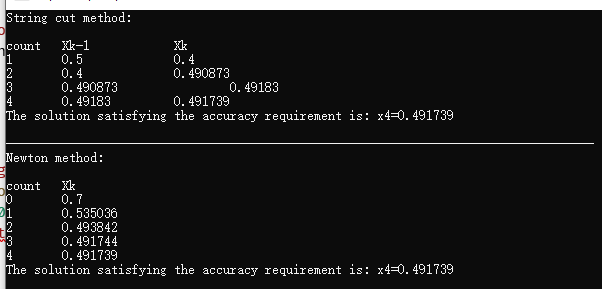
    // 暂停

    getchar();

    return 0;

}

**实验结果与分析：**

****

分析：

可以证明牛顿迭代法的收敛速度和弦割法差不多

**北京科技大学实验报告**

学院： 计算机与通信工程 专业： 计算机科学与技术 班级： 计科184

姓名： 马亮 学号： 41824208 实验日期：2020 年 4月 1日

**实验名称：**用最小二乘法求拟合函数

**实验目的**：编程实现最小二乘算法，以更好地掌握相关知识

**实验仪器：**

编程语言：c++、python

操作系统：ubuntu 18.04LTS

IDE: vscode

C++编译器: g++

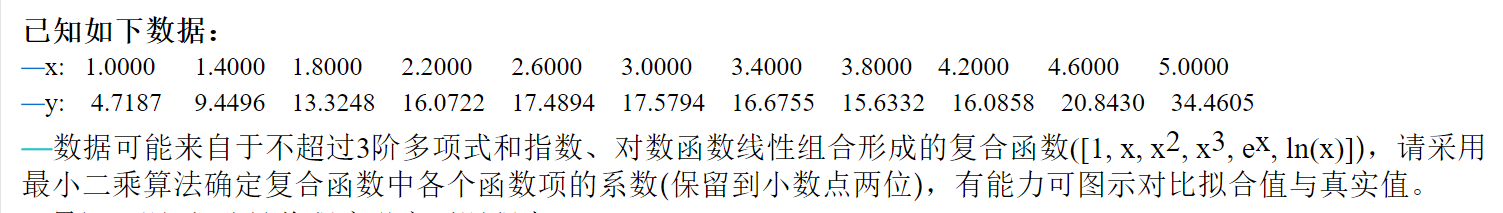
python解释器： python3.6.9

**实验原理：**

最小二乘算法

**实验内容与步骤：**

**实验要求：**

****

**编程实现：**

import numpy as np

from numpy.linalg import solve

import matplotlib.pyplot as plt

def f1(k, j):

    res = 0

    for i in range(m):

        res += f[k](x[i]) \* f[j](x[i])

    return res

def f2(k):

    return sum([f[k](x[i]) \* y[i] for i in range(m)])

# 获取系数矩阵

def getA():

    A = []

    for i in range(n):

        temp = []

        for j in range(n):

            temp.append(f1(i, j))

        A.append(temp)

    return A

# 获取常数列矩阵

def getB():

    B = []

    for i in range(n):

        B.append(f2(i))

    return B

def getfas(A, B):

    a = solve(np.mat(A), np.mat(B).T).T

    return [round(i, 2) for i in a.tolist()[0]]

m = 11

x = [1, 1.4, 1.8, 2.2, 2.6, 3, 3.4, 3.8, 4.2, 4.6, 5]

y = [4.7187, 9.4496, 13.3248, 16.0722, 17.4894, 17.5794, 16.6755, 15.6332, 16.0858, 20.8430, 34.4605]

n = 6

f = [lambda x: 1, lambda x: x, lambda x: pow(x, 2), lambda x: pow(x, 3), lambda x: pow(np.e, x), lambda x: np.log(x)]

def getRes():

    res = getfas(getA(), getB())

    print(f"拟合函数为：{res[0]}+{res[1]}x+{res[2]}x^2{res[3]}x^3+{res[4]}e^x+{res[5]}ln(x)")

    Gx = lambda x: sum([f[i](x) \* res[i] for i in range(n)])

    return Gx

Gx = getRes()

# 将真实值点绘出来

plt.scatter(np.array(x), np.array(y), label="rel point")

# 绘制拟合函数

plt.plot(np.linspace(1, 5), np.array([Gx(i) for i in np.linspace(1, 5)]), 'r', label="fitting function")

plt.xlabel("x")

plt.ylabel("y")

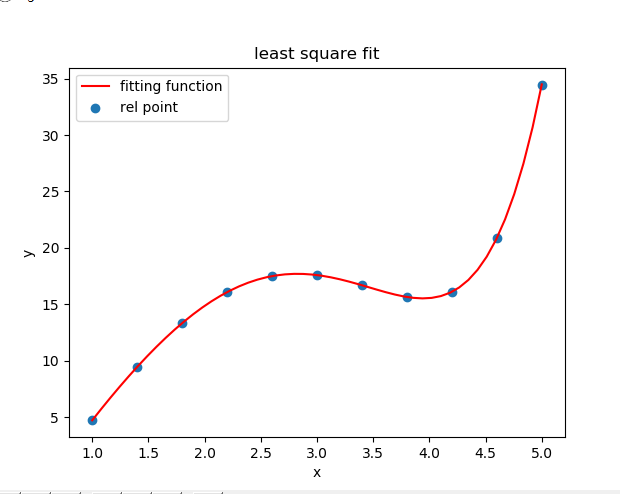
plt.title("least square fit")

plt.legend()

plt.show()

**实验结果与分析：**

****

****

从拟合函数与真实值对比可以看出拟合度很高

**北京科技大学实验报告**

学院： 计算机与通信工程 专业： 计算机科学与技术 班级： 计科184

姓名： 马亮 学号： 41824208 实验日期：2020 年 4月 15日

**实验名称：**求积分

**实验目的：**编程实现求积分的相关算法

**实验仪器：**

编程语言：c++、python

操作系统：ubuntu 18.04LTS

IDE: vscode

C++编译器: g++

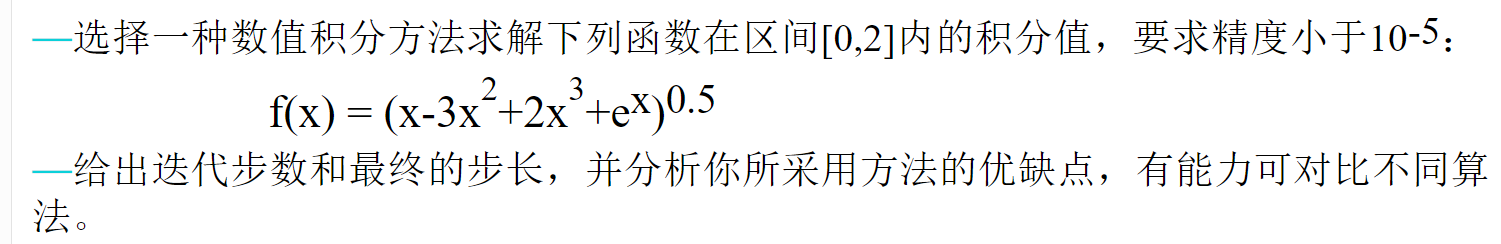
python解释器： python3.6.9

**实验原理：**

变步长的梯形公式求积分算法

**实验内容与步骤：**

**实验要求：**

****

**编程实现：**

from math import \*

def f(x):

    return pow(x-3\*pow(x, 2)+2\*pow(x, 3)+pow(e, x), 0.5)

def T2n(Tn, k):

    temp = 0

    for i in range(1, 2\*\*(k-1) + 1):

        temp += f((2\*i-1)\*(2/(pow(2, k))))

    return (1/2) \* Tn + ((2/(pow(2, k))) \* temp)

tn = (1/2) \* (f(0) + f(2))

t2n = T2n(tn, 2)

print("k    T2k")

print(f"{0:<5}{tn:.6f}")

k = 2

while fabs(t2n - tn) >= 3 \* pow(10, -5):

    print(f"{k-1:<5}{t2n:.9f}")

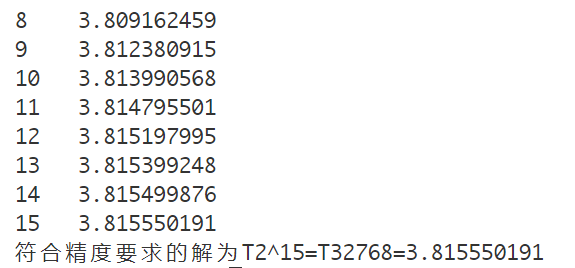
    k += 1

    tn = t2n

    t2n = T2n(tn, k)

print(f"符合精度要求的解为T2^{k-2}=T{2\*\*(k-2)}={tn:.9f}")

**实验结果与分析：**



可见变步长的求积分算法收敛速度并不快.