# 实验五 非线性方程求解(2 课时)

## 一、实验目的

1.了解求解非线性方程的解的常见方法

2.编写牛顿迭代法程序求解非线性方程

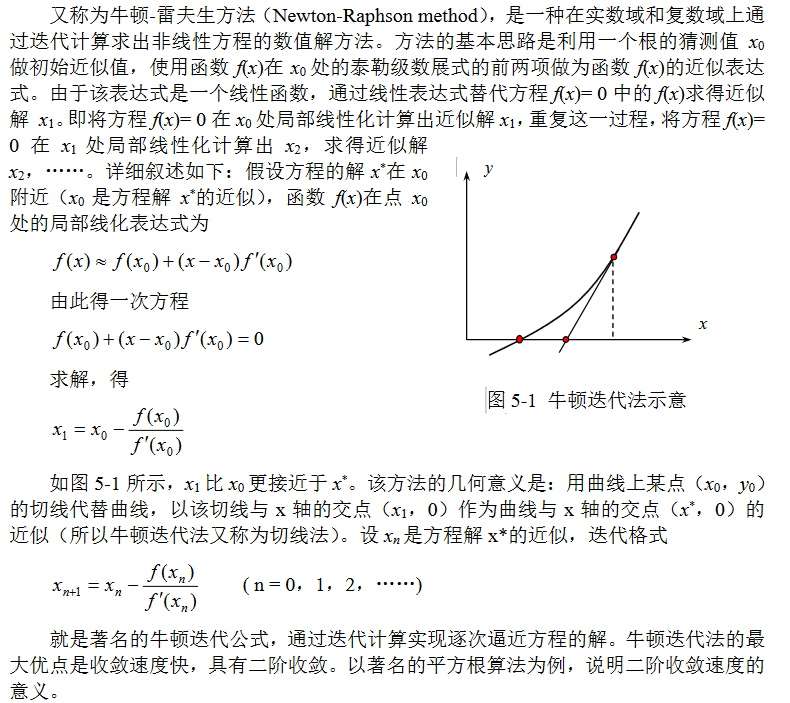
## 二、实验要求

1.设计牛顿迭代法算法，编写程序上机调试。

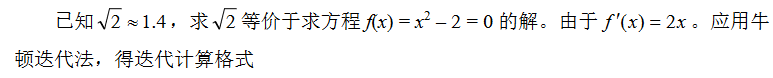
2.进一步加深对迭代法的理解。

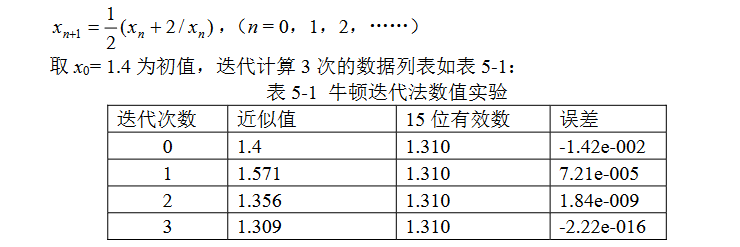
## 三、实验原理

**(一)牛顿迭代法**



**(二) 例子**

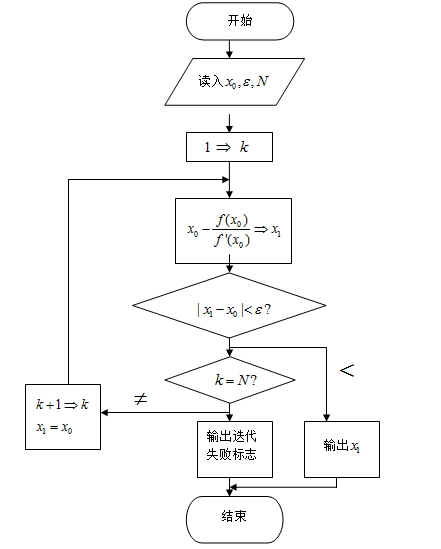
****

****

## 四、实验内容

**（一）算法流程图**

牛顿迭代法算法流程图



**(二)编程作业**

编写 Newton 迭代法通用子程序。实现方程 f(x)=x^6-x-1-0 的满足精度要求的解

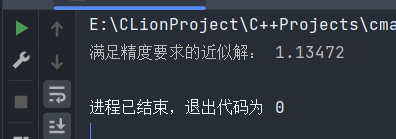
要求求解过程中用一个变量I控制三种状态，其中:

i=0 表示求解满足给定精度的近似解:

i=1 表示f(x0)=0，计算中断;

i=2 表示迭代n次后精度要求仍不满足

*#include* <iostream>  
*#include* <cmath>  
  
*double* f(*double* x) {  
 *return* pow(x, 6) - x - 1;  
}  
  
*double* df(*double* x) {  
 *return* 6 \* pow(x, 5) - 1;  
}  
  
*double* newton\_iteration(*double* x0, *double* acc, *int* max\_iter, *int* &i) {  
 *double* x = x0;  
 *for* (*int* iter = 0; iter < max\_iter; ++iter) {  
 *double* fx = f(x);  
 *if* (fabs(fx) < acc) { *// 找到满足精度的近似解* i = 0;  
 *return* x;  
 }  
 *double* dfx = df(x);  
 *if* (dfx == 0) {*// f`(x0)=0, 计算中断* i = 1;  
 *return* x;  
 }  
 x = x - fx / dfx; *// 迭代* }  
 *// 迭代n次后精度仍未满足* i = 2;  
 *return* x;  
}  
  
*int* main() {  
 *double* x0 = 1.0;*// 初始近似解  
 double* acc = 1e-6;*// 精度  
 int* max\_iter = 1000;*// 最大迭代次数  
 int* i;  
 *double* root = newton\_iteration(x0, acc, max\_iter, i);  
 *if* (i == 0) {  
 std::cout << "满足精度要求的近似解： " << root << std::endl;  
 } *else if* (i == 1) {  
 std::cout << "f(x0)=0，计算中断" << std::endl;  
 } *else* {  
 std::cout << "迭代n次后精度要求仍不满足" << std::endl;  
 }  
 *return* 0;  
}



**（三）选做题**

1.用简化牛顿法计算上述例题。

2.用牛顿下山法计算上述例题。

*#include* <iostream>  
*#include* <cmath>  
  
*double* f(*double* x) {  
 *return* pow(x, 6) - x - 1;  
}  
  
*double* df(*double* x) {  
 *return* 6 \* pow(x, 5) - 1;  
}  
  
*double* newton\_iteration(*double* x0, *double* acc, *int* max\_iter, *int* &i) {  
 *double* x = x0;  
 *for* (*int* iter = 0; iter < max\_iter; ++iter) {  
 *double* fx = f(x);  
 *if* (fabs(fx) < acc) { *// 找到满足精度的近似解* i = 0;  
 *return* x;  
 }  
 *double* dfx = df(x);  
 *if* (dfx == 0) {*// f`(x0)=0, 计算中断* i = 1;  
 *return* x;  
 }  
 *// 牛顿下山法  
 double* step = fx / dfx;  
 *while* (f(x - step) > f(x) && step > 1e-8) {  
 step /= 2;  
 }  
 x = x - step;  
}  
 *// 迭代n次后精度仍未满足* i = 2;  
 *return* x;  
}  
  
*int* main() {  
 *double* x0 = 1.0;*// 初始近似解  
 double* acc = 1e-6;*// 精度  
 int* max\_iter = 1000;*// 最大迭代次数  
 int* i;  
 *double* root = newton\_iteration(x0, acc, max\_iter, i);  
 *if* (i == 0) {  
 std::cout << "满足精度要求的近似解： " << root << std::endl;  
 } *else if* (i == 1) {  
 std::cout << "f(x0)=0，计算中断" << std::endl;  
 } *else* {  
 std::cout << "迭代n次后精度要求仍不满足" << std::endl;  
 }  
 *return* 0;  
}

