

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

又称为牛顿-雷夫生方法 (Newton-Raphson method)，是一种在实数域和复数域上通过迭代计算求出非线性方程的数值解方法。方法的基本思路是利用一个根的猜测值 x_0 做初始近似值，使用函数 $f(x)$ 在 x_0 处的泰勒级数展开的前两项做为函数 $f(x)$ 的近似表达式。由于该表达式是一个线性函数，通过线性表达式替代方程 $f(x)=0$ 中的 $f(x)$ 求得近似解 x_1 。即将方程 $f(x)=0$ 在 x_0 处局部线性化计算出近似解 x_1 ，重复这一过程，将方程 $f(x)=0$ 在 x_1 处局部线性化计算出 x_2 ，求得近似解 x_2 ，……。详细叙述如下：假设方程的解 x^* 在 x_0 附近 (x_0 是方程解 x^* 的近似)，函数 $f(x)$ 在点 x_0 处的局部线性化表达式为

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0)$$

由此得一次方程

$$f(x_0) + (x - x_0)f'(x_0) = 0$$

求解，得

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

如图 5-1 所示， x_1 比 x_0 更接近于 x^* 。该方法的几何意义是：用曲线上某点 (x_0, y_0) 的切线代替曲线，以该切线与 x 轴的交点 $(x_1, 0)$ 作为曲线与 x 轴的交点 $(x^*, 0)$ 的近似（所以牛顿迭代法又称为切线法）。设 x_n 是方程解 x^* 的近似，迭代格式

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (n = 0, 1, 2, \dots)$$

就是著名的牛顿迭代公式，通过迭代计算实现逐次逼近方程的解。牛顿迭代法的最大优点是收敛速度快，具有二阶收敛。以著名的平方根算法为例，说明二阶收敛速度的意义。

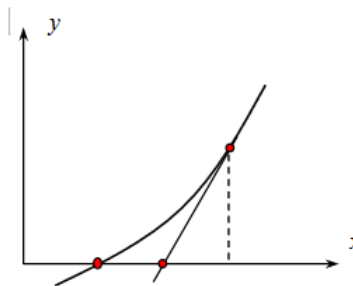


图 5-1 牛顿迭代法示意

(二) 例子

已知 $\sqrt{2} \approx 1.4$ ，求 $\sqrt{2}$ 等价于求方程 $f(x) = x^2 - 2 = 0$ 的解。由于 $f'(x) = 2x$ 。应用牛顿迭代法，得迭代计算格式

$$x_{n+1} = \frac{1}{2}(x_n + 2/x_n), \quad (n = 0, 1, 2, \dots)$$

取 $x_0 = 1.4$ 为初值，迭代计算 3 次的数据列表如表 5-1：

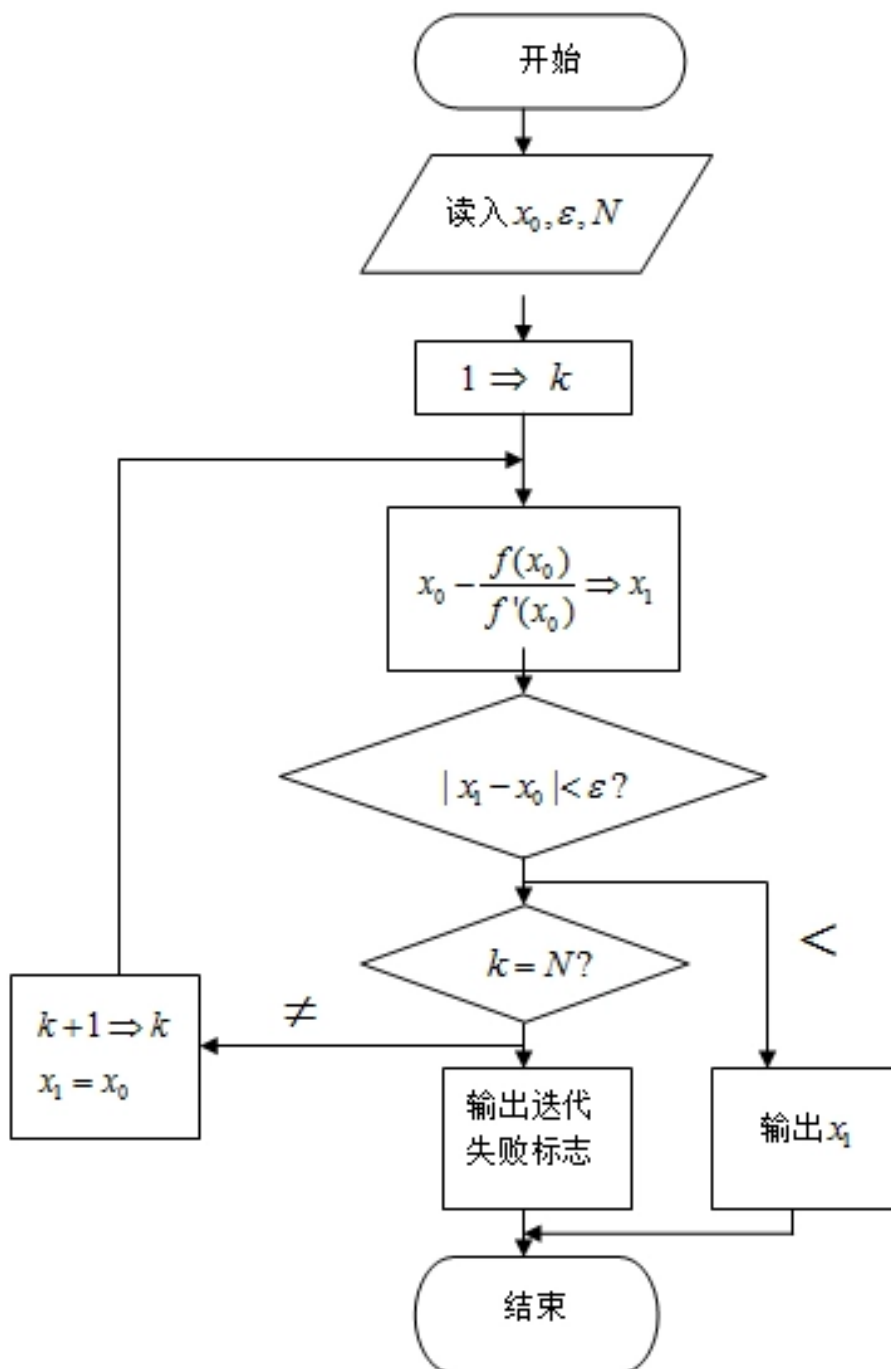
表 5-1 牛顿迭代法数值实验

迭代次数	近似值	15 位有效数	误差
0	1.4	1.310	-1.42e-002
1	1.571	1.310	7.21e-005
2	1.356	1.310	1.84e-009
3	1.309	1.310	-2.22e-016

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 $f(x_0)=0$ ，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>
```

```

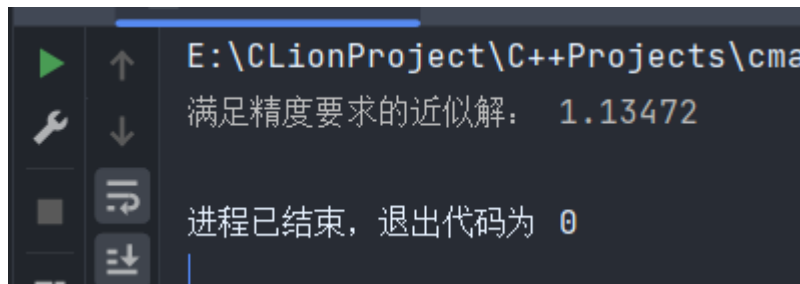
double f(double x) {
    return pow(x, 6) - x - 1;
}

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;
}

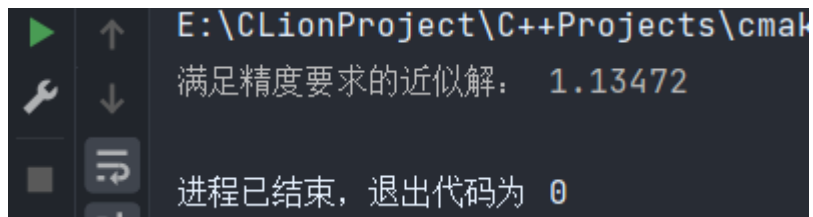
double newton_iteration(double x0, double acc, int max_iter, int &i)
{
    double x = x0;
    for (int iter = 0; iter < max_iter; ++iter) {
        double fx = f(x);
        if (fabs(fx) < acc) { // 找到满足精度的近似解
            i = 0;
            return x;
        }
        double dfx = df(x);
        if (dfx == 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;
}

```



E:\CLionProject\C++Projects\cmak

满足精度要求的近似解: 1.13472

进程已结束, 退出代码为 0