天津理工大学 实验报告

学院（系）名称：计算机科学与工程学院

姓名	王帆	**学号**	20152180	**专业**	计算机科学与技术
班级	15-计算机一班	**实验项目**	实验一　方程求根		
课程名称		数值计算方法		**课程代码**	0665026
实验时间		2017 年 5 月 15 日 第 7 – 8 节		**实验地点**	7-219
批改意见				**成绩**	
				教师签字：	

一、　实验目的

掌握用二分法，迭代法，牛顿迭代法和弦截法求解非线性方程的根，能够熟练的将上述算法编程实现。

二、　实验环境

- 硬件环境：IBM—PC 或兼容机
- 软件环境：Windows 操作系统，VC6.0
- 编程语言：C 或 C++

三、　实验内容

1.用二分法求方程 $x^5+3x-1=0$ 在[0,1]的根，要求准确到 $1/2\times10^{-2}$。

要求：

(1) 对该区间使用二分法求方程的满足精度要求的根，每二分一次，用新生成区间长度的一半作为是否二分结束的判断条件；

(2) 要求精度 ε 从键盘输入；

(3) 将二分法求方程根的实现过程用算法框图进行描述；

(4) 输出每一次二分过程所得到的区间端点 a_k、b_k 以及区间中点 x_k 的信息，最后打印输出满足精度要求的方程根的近似值。

2.用迭代法、牛顿迭代法和双点弦截法求解方程 $x=e^{-x}$ 在 x=0.5 附近的一个根，要求精确到小数点后五位。

要求：

(1) 在同一个程序里面将三种算法编程实现；

(2) 精度 ε 要求从键盘输入；

(3) 将三种算法的每一步迭代计算结果打印输出，最后输出满足精度要求的方程的根；

(4) 根据计算结果，比较三种算法的收敛速度。

四、 实验要求

1. 每一个实验内容要求自己独立完成，不允许抄袭别人，否则按不及格处理；

2. 按照实验要求，根据自己的程序编写情况绘制相应的算法框图或描述算法步骤；

3. 按照实验内容和相应的要求书写实验报告；

4. 在实验过程部分，要求根据实验内容和要求书写每一个实验相应的算法步骤或框图、运行过程和运行结果的截图、运行结果分析、以及程序源代码。每一个实验要求书写下述内容：

（1） 算法步骤描述或算法框图

（2） 程序源代码

（3） 运行结果（要求截图）

（4） 运行结果分析

5. 在规定的时间内上交实验报告。

五、 实验过程

1. 用二分法求方程 x5+3x-1=0 在[0,1]的根，要求准确到 1/2×10-2。

算法描述：

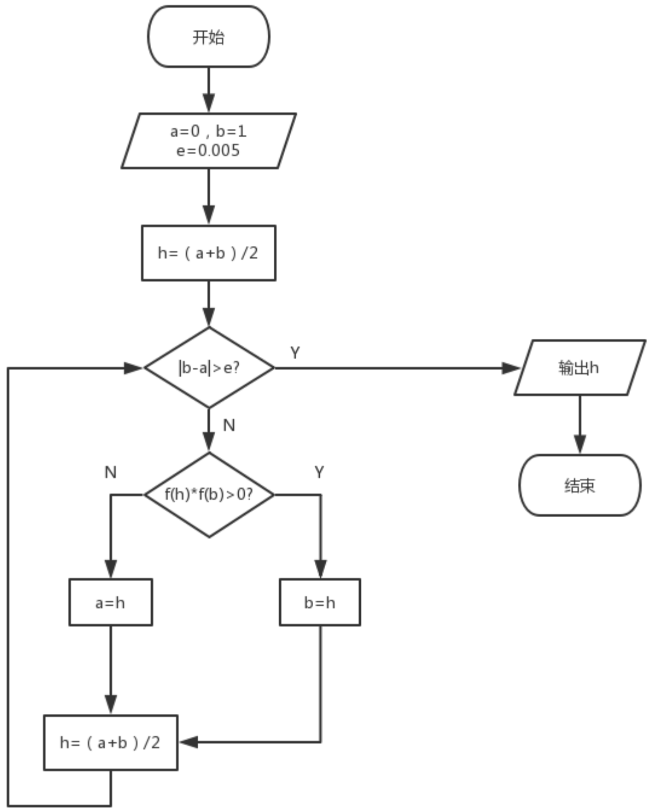

图 1-1 二分法框图

代码实现：

```
#include <iostream>
#include <cmath>
using namespace std;
double f(double x){
    return pow(x,5)+3*x-1;
}
void dichotomia(){
    double a=0,b=1,e;
    cin>>e;
    while(abs(b-a)>e){
        double h=(a+b)/2;
        cout<<"["<<a<<","<<b<<"]"<<" "<<h<<endl;
        if(f(h)*f(a)<0){
            b=h;
        }
        else if(f(h)*f(b)<0){
            a=h;
        }
    }
    cout<<"x="<<(a+b)/2;
}
int main()
{
    dichotomia();
    return 0;
}
```

运行结果：

```
C:\Users\Du\Documents\数值计算方法\实验\实验一\二分法.exe
0.005
[0,1] 0.5
[0,0.5] 0.25
[0.25,0.5] 0.375
[0.25,0.375] 0.3125
[0.3125,0.375] 0.34375
[0.3125,0.34375] 0.328125
[0.328125,0.34375] 0.335938
[0.328125,0.335938] 0.332031
x=0.330078
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

图 1-2 二分法运行结果

运行分析：

根据输入的精度（$1/2*10^{-2}$）进行二分比较，最终获得结果 0.330078。

其中，0.330078 是由 0.332031 作为前一次二分中值进行比较后，最终得到 a，b 的中值。

2. 用迭代法、牛顿迭代法和双点弦截法求解方程 $x=e^{-x}$ 在 $x=0.5$ 附近的一个根，要求精确到小数点后五位。

算法描述：

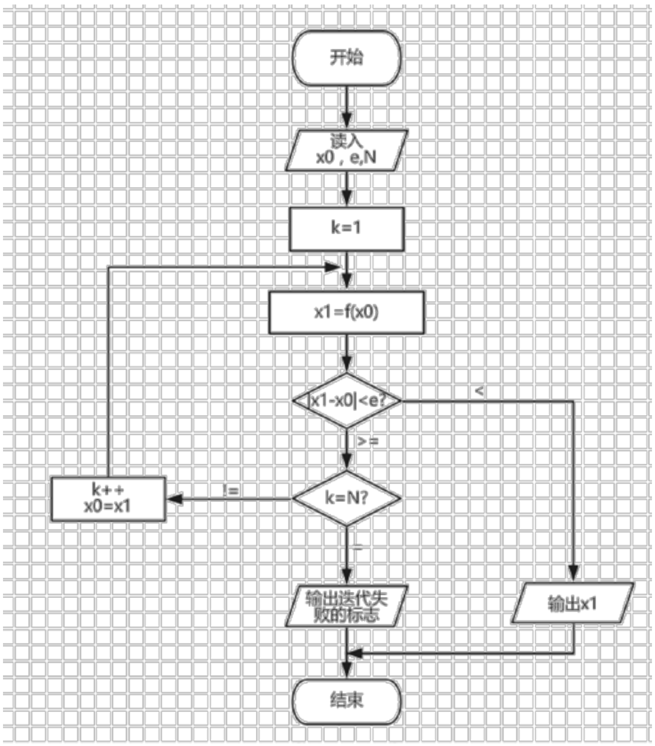

图 2-1 迭代法算法框图

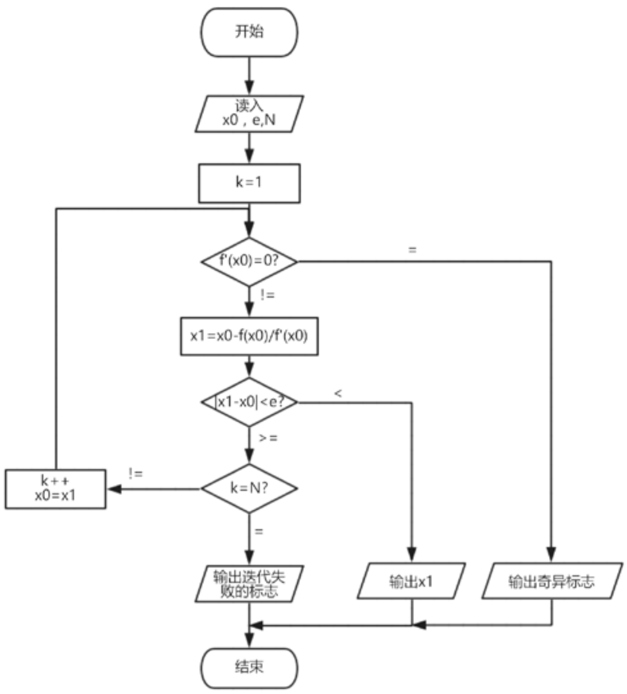

图 2-2 牛顿迭代法算法框图

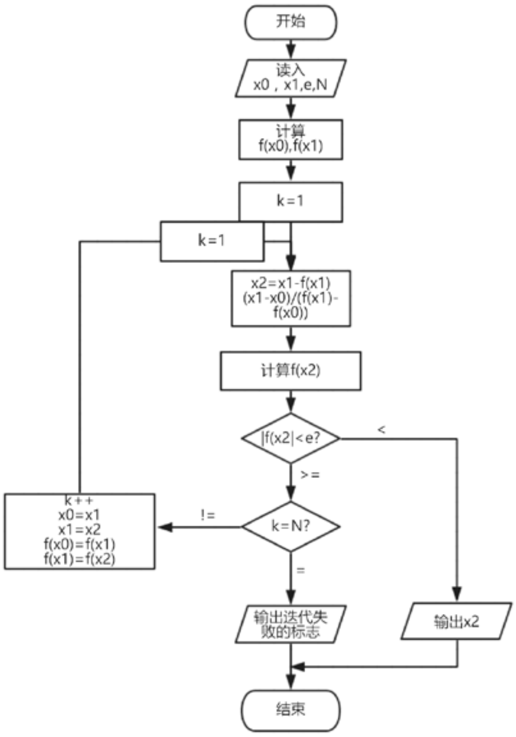

图 2-3 双点弦截法算法框图

代码实现：

```
#include <iostream>
#include <cmath>
using namespace std;
double e=0.5e-5;//精度
const double N=1000;//最大迭代次数
//迭代法迭代格式
double f(double x){
    return exp(-x);
}
//求导
double _f(double x){
    return x-(x-exp(-x))/(x+1);
}
//求函数
double __f(double x){
    return x*exp(x)-1;
}
//双点弦截法迭代格式
double __f(double x1, double x0){
    return x1-(__f(x1)*(x1-x0))/(__f(x1)-__f(x0));
}

int Iteration(){
    cout<<"迭代法开始：" <<endl;
    cout<<"k     X"<<endl;
```

```
    double x=0.5;
    double X=f(x);
    int k=1;
    while(fabs(X-x)>=e){
        if(k==N) return 0;
        else{
            k++;
            x=X;
        }
        X=f(x);
        cout<<k<<" "<<X<<endl;
    }
    cout<<"迭代法最终结果："<<X<<endl;
    return k;
}
int Newton(){
    cout<<"牛顿迭代法开始：" <<endl;
    cout<<"k     X"<<endl;
    double x=0.5;
    double X=_f(x);
    int k=1;
    while(1){
        if(_f(x)!=0){
            X=_f(x);
            cout<<k<<" "<<X<<endl;
            if(fabs(X-x)>=e){
                if(k==N) return 0;
                else{
                    k++;
                    x=X;
                }
            }
            else{
                cout<<"牛顿迭代法最终结果："<<X<<endl;
                return k;
            }
        }
        else return 0;
    }
}

int Secant(){
    cout<<"双点弦截法法开始：" <<endl;
    cout<<"k     X"<<endl;
    double x0=0.5;
```

```
    double x1=0.6;
    double x2;
    int k=1;

    while(1){
        x2=__f(x1,x0);
        cout<<k<<" "<<x2<<endl;
        if(fabs(__f(x2))>e){
            if(k==N) return 0;
            else{
                k++;
                x0=x1;
                x1=x2;
            }
        }
        else{
            cout<<"双 点弦截法法最终结果："<<x2<<endl;
            return k;
        }
    }
}

void compare(int k1, int k2, int k3){
    cout<<endl<<"比较三种方法迭代次数："<<endl;
    cout<<"迭代法：    "<<k1<<endl;
    cout<<"牛顿迭代法："<<k2<<endl;
    cout<<"双点弦截法："<<k3<<endl;
    if(k1>k2){
        if(k2>k3) cout<<"双 点弦截法效率最高！";
        else cout<<"牛 顿迭代法效率最高！";
    }
    else if(k1<k3) cout<<"迭 代法效率最高";
    return;
}

int main(){
    int k1=Iteration();
    cout<<endl;
    int k2=Newton();
    cout<<endl;
    int k3=Secant();
    compare(k1,k2,k3);
    return 0;
}
```

运行结果：

```
C:\Users\Du\Documents\数值计算方法\实验\实验一\迭代.exe
迭代法开始：
k       X
2  0.545239
3  0.579703
4  0.560065
5  0.571172
6  0.564863
7  0.568438
8  0.566409
9  0.56756
10  0.566907
11  0.567277
12  0.567067
13  0.567186
14  0.567119
15  0.567157
16  0.567135
17  0.567148
18  0.567141
19  0.567145
迭代法最终结果：    0.567145

牛顿迭代法开始：
k       X
1  0.57102
2  0.567156
3  0.567143
4  0.567143
牛顿迭代法最终结果：    0.567143

双点弦截法法开始：
k       X
1  0.565315
2  0.567095
3  0.567143
双点弦截法法最终结果：    0.567143

比较三种方法迭代次数：
迭代法：       19
牛顿迭代法：    4
双点弦截法：    3
双点弦截法效率最高！
```

图 2-4 迭代法/牛顿迭代法/双点弦截法运行结果

运行分析：

通过对于三种迭代法的比较，直观地从迭代次数来看，迭代法迭代次数最多，牛顿迭代法其次，双点弦截法最少。

而从设计思想来看，迭代法最简单，但是初值选取很重要。如果初值选择较差则会导致迭代次数过多，迭代效率降低，甚至无法得到结果；牛顿迭代法是迭代法的优化，通过对某点切线交点的迭代处理得到最终结果。这种方法最大的问题在于求导数的过程，对于计算机来讲求导需要运用插值计算，而在插值的过程中也会存在一定的误差；双点弦截法则是对于牛顿迭代法求导的优化，直接对两点求斜率，并代替牛顿迭代法的导数值求解。这样的好处是能够简化计算过程，也避免了一定的误差出现。综上，三种方法各有优劣。

六、 实验总结及心得体会

数值计算方法是一门理论结合实践的课程。此前的理论课中我们学习了许多处理数据的理论知识，而真正在计算机上编程实现的过程中，则会出现很多之前在理论课上没有体会到的问题所在。第一个实验是一个开始，也是一个入门的过程，在今后的实验过程中我将加深对于理论联系实际的理解，对于每个算法都认真地实现，从而更好地体会他们在计算机科学中的用途，为以后的学习生活打好基础。