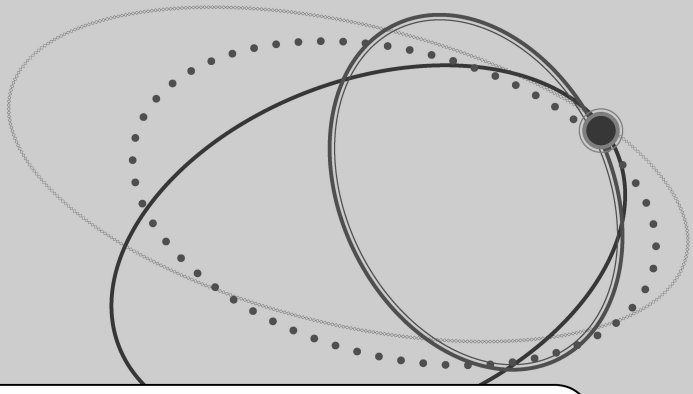

第5章

非线性方程与方程组的求解

5.1 求非线性方程实根的对分法

【功能】

用对分法搜索方程 $f(x)=0$ 在区间$[a,b]$内的实根。

【方法说明】

从区间左端点 $x=a$ 开始，以 h 为步长，逐步往后进行搜索。

对于在搜索过程中遇到的每一个子区间$[x_k,x_{k+1}]$(其中 $x_{k+1}=x_k+h$)做如下处理：

若 $f(x_k)=0$，则 x_k 为一个实根，且从 $x_k+h/2$ 开始往后再搜索；

若 $f(x_{k+1})=0$，则 x_{k+1}为一个实根，且从 $x_{k+1}+h/2$ 开始往后再搜索；

若 $f(x_k)f(x_{k+1})>0$，则说明在当前子区间内无实根或 h 选得过大，放弃本子区间，从 x_{k+1}开始往后再搜索；

若 $f(x_k)f(x_{k+1})<0$，则说明在当前子区间内有实根，此时利用对分法，直到求得一个实根为止，然后从 x_{k+1}开始往后再搜索。

上述过程一直进行到区间右端点 b 为止。

特别要注意，在根的搜索过程中，要合理选择步长，尽量避免根的丢失。

【函数语句与形参说明】

```
int dhrt(double a, double b, double h, double eps, double x[],
                                int m, double (* f)(double))
```

形参与函数类型	参 数 意 义
double a	求根区间的左端点
double b	求根区间的右端点
double h	搜索求根时采用的步长
double eps	控制精度要求
double x[m]	返回在区间$[a,b]$内的实根。实根个数由函数值返回

续表

形参与函数类型	参 数 意 义
int m	在区间[a,b]内实根个数的预估值
double (*f)()	指向计算方程左端函数 $f(x)$值的函数名(由用户自编)
int dhrt()	函数返回在区间[a,b]内实际搜索到的实根个数。若此值等于 m,则有可能没有搜索完

计算方程左端函数 $f(x)$值的函数形式为

```
double  f(double x)
{ double  z;
  z=f(x)的表达式;
  return(z);
}
```

【函数程序】

```
//方程求根对分法.cpp
#include <cmath>
#include <iostream>
using namespace std;
//a         求根区间的左端点
//b         求根区间的右端点
//h         搜索求根所采用的步长
//eps       控制精度要求
//x[m]      存放返回的实根。实根个数由函数值返回
//m         实根个数的预估值
//f         方程左端函数 f(x)的函数名
//函数返回搜索到的实根个数。若此值等于 m,则可能没有搜索完
int dhrt(double a, double b, double h, double eps, double x[],
                                    int m, double (*f)(double))
{
    int n,js;
    double z,y,z1,y1,z0,y0;
    if (a>b)
    {
        z =a; a =b; b =z;
    }
    n=0; z=a; y=(*f)(z);
    while ((z<=b+h/2.0)&&(n!=m))
    {
        if (fabs(y)<eps)
        {
            n=n+1; x[n-1]=z;
            z=z+h/2.0; y=(*f)(z);
```

```
        }
        else
        {
            z1=z+h; y1=(*f)(z1);
            if (fabs(y1)<eps)
            {
                n=n+1; x[n-1]=z1;
                z=z1+h/2.0; y=(*f)(z);
            }
            else if (y*y1>0.0)
            {
                y=y1; z=z1;
            }
            else
            {
                js=0;
                while (js==0)
                {
                    if (fabs(z1-z)<eps)
                    {
                        n=n+1; x[n-1]=(z1+z)/2.0;
                        z=z1+h/2.0; y=(*f)(z);
                        js=1;
                    }
                    else
                    {
                        z0=(z1+z)/2.0; y0=(*f)(z0);
                        if (fabs(y0)<eps)
                        {
                            x[n]=z0; n=n+1; js=1;
                            z=z0+h/2.0; y=(*f)(z);
                        }
                        else if ((y*y0)<0.0)
                        {
                            z1=z0; y1=y0;
                        }
                        else { z=z0; y=y0;}
                    }
                }
            }
        }
    }
    return(n);
}
```

【例】 求方程

$$f(x) = x^6 - 5x^5 + 3x^4 + x^3 - 7x^2 + 7x - 20 = 0$$

在区间[－2,5]内的所有实根。取步长 $h=0.2$，控制精度要求为 $\varepsilon=0.000\ 001$。

由于本方程为6次代数方程，最多有6个实根，因此取 $m=6$。

主函数程序与计算方程左端函数 $f(x)$值的函数程序如下：

```
//方程求根对分法例
#include <cmath>
#include <iostream>
#include "方程求根对分法.cpp"
using namespace std;
int main()
{
    int i,n;
    int m=6;
    double x[6];
    double dhrtf(double);
    n=dhrt(-2.0,5.0,0.2,0.000001,x,m,dhrtf);
    cout <<"根的个数 =" <<n <<endl;
    for (i=0; i<=n-1; i++)
        cout <<"x(" <<i <<") =" <<x[i] <<endl;
    return 0;
}
//f(x)
double dhrtf(double x)
{
    double z;
    z=(((((x-5.0)*x+3.0)*x+1.0)*x-7.0)*x+7.0)*x-20.0;
    return(z);
}
```

运行结果为

```
根的个数=2
x(0)=-1.40246
x(1)=  4.33376
```

5.2 求非线性方程一个实根的牛顿迭代法

【功能】

用牛顿迭代法求方程 $f(x)=0$ 的一个实根。

【方法说明】

设方程 $f(x)=0$ 满足下列条件：

(1) $f(x)$在区间$[a,b]$上的 $f'(x)$与 $f''(x)$均存在，且 $f'(x)$与 $f''(x)$的符号在区间$[a,b]$上均各自保持不变；

(2) $f(a)f(b)<0$;

(3) $f(x_0)f''(x_0)>0, x_0, x\in[a,b]$。

则方程 $f(x)=0$ 在区间$[a,b]$上有且只有一个实根,由牛顿迭代公式

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

计算得到的根的近似值序列收敛于方程 $f(x)=0$ 的根。

结束迭代过程的条件为

$$|f(x_{n+1})|<\varepsilon \quad 与 \quad |x_{n+1}-x_n|<\varepsilon$$

同时成立。其中,ε 为预先给定的精度要求。

【函数语句与形参说明】

```
int newt(double *x, double eps, double (*f)(double), double (*df)(double))
```

形参与函数类型	参 数 意 义
double　*x	指向迭代初值。返回时指向迭代终值
double　eps	控制精度要求
double (*f)()	指向计算 $f(x)$值的函数名(由用户自编)
double (*df)()	指向计算 $f'(x)$值的函数名(由用户自编)
int　newt()	函数返回迭代次数。若返回−1,则表示出现 $f'(x)=0$ 的情况。程序最多迭代次数为 500

计算 $f(x)$与 $f'(x)$值的函数形式为

```
double  f(double x)
{
    double y;
    y=f(x)的表达式;
    return(y);
}
double  df(double x)
{
    double dy;
    dy=f'(x) 的表达式;
    return(dy);
}
```

【函数程序】

```
//方程求根 newton 法.cpp
#include <iostream>
#include <cmath>
using namespace std;
//x          存放方程根的初值。返回迭代终值
```

```
//eps       控制精度要求
//f         方程左端函数 f(x)的函数名
//df        方程左端函数 f(x)一阶导函数名
//函数返回迭代次数。若返回 -1,则表示出现 df/dx=0 的情况。程序最多迭代次数为 500
int newt(double *x, double eps, double (*f)(double), double (*df)(double))
{
    int k, interation;
    double y, dy, d, p, x0, x1;
    interation =500;                              //最大迭代次数
    k =0; x0 = *x;
    y = (*f)(x0); dy = (*df)(x0);
    d=eps+1.0;
    while ((d>=eps)&&(k!=interation))
    {
        if (fabs(dy)+1.0==1.0)                    //出现 df(x)/dx=0
        {
            cout <<"dy ==0 !";  return(-1);
        }
      x1 =x0 -y/dy;                               //迭代
      y = (*f)(x1); dy = (*df)(x1);
      d=fabs(x1-x0); p=fabs(y);
      if (p>d) d=p;
      x0 =x1; k =k +1;
    }
    *x =x0;
    return(k);
}
```

【例】 用牛顿迭代法求方程

$$f(x) = x^3 - x^2 - 1 = 0$$

在 $x_0 = 1.5$ 附近的一个实根。取 $\varepsilon = 0.000\ 001$。其中

$$f'(x) = 3x^2 - 2x$$

主函数程序以及计算 $f(x)$与 $f'(x)$值的函数程序如下：

```
//方程求根 newton 法例
#include <cmath>
#include <iostream>
#include "方程求根 newton 法.cpp"
using namespace std;
int main()
{
    int k;
    double x,eps;
    double newtf(double x), newtdf(double x);
    eps=0.000001;  x=1.5;
    k=newt(&x,eps,newtf,newtdf);
```

```
    if (k>=0)
    {
        cout <<"迭代次数 =" <<k <<endl;
        cout <<"迭代终值 x =" <<x <<endl;
    }
    return 0;
}
//f(x)
double newtf(double x)
{
    return(x * x * (x-1.0)-1.0);
}
//df(x)/dx
double newtdf(double x)
{
    return(3.0 * x * x -2.0 * x);
}
```

运行结果为

```
迭代次数=4
迭代终值 x=1.46557
```

5.3　求非线性方程一个实根的埃特金迭代法

【功能】

用埃特金(Aitken)迭代法求非线性方程 $x=\varphi(x)$的一个实根。

【方法说明】

设非线性方程为

$$x = \varphi(x)$$

取初值 x_0。埃特金迭代法迭代一次的过程如下。

预报

$$u = \varphi(x_n)$$

再预报

$$v = \varphi(u)$$

校正

$$x_{n+1} = v - \frac{(v-u)^2}{v-2u+x_n}$$

结束迭代过程的条件为

$$|v-u|<\varepsilon$$

此时 v 即为非线性方程的一个实根。其中，ε 为预先给定的精度要求。

埃特金迭代法具有良好的收敛性。一方面，埃特金迭代法的收敛速度比较快；另一方面，一个简单迭代法不收敛的迭代公式经埃特金迭代法处理后一般就会收敛。

【函数语句与形参说明】

```
int atkn(double * x, double eps, double ( * f)(double))
```

形参与函数类型	参 数 意 义
double * x	指向迭代初值。返回时指向迭代终值
double eps	控制精度要求
double (* f)()	指向计算 $\varphi(x)$ 值的函数名（由用户自编）
int atkn()	函数返回迭代次数。程序最多迭代次数为 500

计算 $\varphi(x)$ 值的函数形式为

```
double  f(double x)
{ double  y;
  y=φ(x)的表达式;
  return(y);
}
```

【函数程序】

```
//方程求根 aitken 迭代法.cpp
#include <cmath>
#include <iostream>
using namespace std;
//x         存放方程根的初值。返回迭代终值
//eps       控制精度要求
//f         简单迭代公式右端函数 φ(x)的函数名
//函数返回迭代次数。程序最多迭代次数为 500
int atkn(double * x, double eps, double ( * f)(double))
{
    int flag, k, interation;
    double u, v, x0;
    interation =500;                    //最大迭代次数
    k =0; x0 = * x; flag =0;
    while ((flag==0)&&(k!=interation))
    {
        k =k +1;
        u =( * f)(x0); v =( * f)(u);
        if (fabs(u-v)<eps)
        {
            x0 =v; flag =1;
        }
```

```
        else
            x0 =v-(v-u) * (v-u)/(v-2.0 * u+x0);
    }
    * x =x0;
    return(k);
}
```

【例】 用埃特金迭代法求方程

$$x = 6 - x^2$$

在 $x_0 = 0.0$ 附近的一个实根。取 $\varepsilon = 0.000\ 000\ 1$。

主函数程序以及计算 $\varphi(x)$值的函数程序如下：

```
//方程求根 aitken 迭代法例
#include <cmath>
#include <iostream>
#include "方程求根 aitken 迭代法.cpp"
using namespace std;
int main()
{
    int k;
    double x,eps,atknf(double);
    eps =0.0000001;  x =0.0;
    k =atkn(&x,eps,atknf);
    cout <<"迭代次数 =" <<k <<endl;
    cout <<"迭代终值 x =" <<x <<endl;
    return 0;
}
//φ(x)
double atknf(double x)
{
    return(6.0-x * x);
}
```

运行结果为

```
迭代次数=8
迭代终值 x=2
```

5.4 求非线性方程一个实根的试位法

【功能】

利用试位法求非线性方程 $f(x)=0$ 在给定区间$[a, b]$内的一个实根。

【方法说明】

试位法也称割线法。

设 $f(a)f(b)<0$，则在区间$[a, b]$内有实根。

用直线（即弦）连接点$(a,f(a))$和$(b,f(b))$。弦的方程是

$$y(x)=\left[\frac{f(b)-f(a)}{b-a}\right]x+\left[f(a)-\frac{f(b)-f(a)}{b-a}a\right]$$

令 $y(x)=0$，解出的 x 值作为方程根的估值，得到

$$x_{new}=\frac{af(b)-bf(a)}{f(b)-f(a)}$$

有了估值 x_{new}后，检验乘积 $f(a)f(x_{new})$的符号。如果这个乘积小于0，则置 $b=x_{new}$，否则置 $a=x_{new}$。如果这个乘积的值是0（或小于指定的公差 eps），则 x_{new}就是要求的根。

【函数语句与形参说明】

```
int fals(double a, double b, double eps, double (* f)(double), double * x)
```

形参与函数类型	参 数 意 义
double a	求根区间的左端点
double b	求根区间的右端点
double eps	控制精度要求
double (* f)()	指向计算方程左端函数 $f(x)$值的函数名（由用户自编）
double * x	存放方程根的初值。返回迭代终值
int fals()	函数返回迭代次数。若为－1则表示 $f(a)f(b)>0$

计算方程左端函数 $f(x)$值的函数形式为

```
double  f(double x)
{ double  z;
  z=f(x)的表达式;
  return(z);
}
```

【函数程序】

```
//方程求根试位法.cpp
#include <iostream>
#include <cmath>
using namespace std;
//a         求根区间的左端点
//b         求根区间的右端点
//eps       控制精度要求
//f         方程左端函数 f(x)的函数名
//x         存放方程根的初值。返回迭代终值
//函数返回迭代次数。若为-1则表示 f(a)f(b)>0
int fals(double a, double b, double eps, double (* f)(double), double * x)
{
```