21 世纪化学丛书

计 算 化 学

郭纯孝　编著

化学工业出版社
化学与应用化学出版中心
·北　京·

(京) 新登字 039 号

图书在版编目(CIP)数据

计算化学/ 郭纯孝编著. —北京：化学工业出版社，
2004.3
（21 世纪化学丛书）
ISBN 7-5025-5235-9

Ⅰ. 计… Ⅱ. 郭… Ⅲ. 计算化学 Ⅳ. 06-04

中国版本图书馆 CIP 数据核字（2004）第 016822 号

21 世纪化学丛书
计算化学
郭纯孝 编著
责任编辑：陈 丽 刘俊之
责任校对：顾淑云
封面设计：蒋艳君
*
化 学 工 业 出 版 社
化学与应用化学出版中心 出版发行
（北京市朝阳区惠新里 3 号 邮政编码 100029）
发行电话：（010）64982530
http ：// www.cip.com.cn
*
新华书店北京发行所经销
聚鑫印刷有限责任公司印刷
三河市延风装订厂装订
开本 720 毫米×1000 毫米 1/16 印张 24¾ 字数 447 千字
2004 年 4 月第 1 版 2004 年 4 月北京第 1 次印刷
ISBN 7-5025-5235-9/O・41
定 价：42.00 元

前　　言

应《21 世纪化学丛书》主编陈洪渊院士和化学工业出版社的邀请编写了《计算化学》一书。

现代的化学研究中，计算机已成为不可缺少的有力工具。学习使用计算机解决化学领域中遇到的各种数值计算，已成为化学中不可缺少的部分。计算化学的领域非常丰富，但作为基础还是数值计算的编制程序能力，解决实际问题的基本技能。

本书采用当今世界上最流行的 C 程序设计语言。让读者进一步学会用 C 程序设计语言解决化学领域中的各种数值计算问题，重点在于培养读者编制程序的能力。

主要内容包括：数值计算的基本概念、计算误差、方程求根、函数插值、数值积分、常微分方程求解、线性方程组的求解、非线性拟合、一元和多元回归。

本书的后三章是属于专题科学研究性质，是计算化学中的高级应用，也是计算化学的新成果、新趋势。本书最大的特点是不仅给出数值计算方法，而且给出了大量的科研上很实用的计算程序。

本书力图做到选材合适、概念准确、简明扼要、条理清楚、通俗易懂、着重于应用，不过多地进行理论推导，使本书成为宜教宜学的教材。给出算例较丰富，书中程序均在 Turboc 2.0 编译通过。已经进行过实际应用，具有可靠性和通用性。输入输出采用文件系统，方便简单。每章后附有一定量的习题，供研究生和科技工作者使用。

本书第 10 章由吉林大学张涛博士编写。第 11 章由吉林大学郭慧云教授编写。作者编制通用的正交试验极差、方差分析计算程序，并统编全书。在本书的写作过程中，作者受聘长春大学光华学院，得到了康中集团董事长和长春大学光华学院董事长康井山先生和长春大学光华学院于福院长的大力资助，在此表示衷心的感谢！

由于时间紧和作者水平所限，在内容安排、材料选取上难免存在缺点和错误，恳切希望读者予以批评指正。

郭纯孝

2004 年 2 月

目　　录

第 1 章　绪　　论

1.1　微机在化学中的应用

自 1946 年美国宾夕法尼亚大学发明第一台电子管计算机以来，近 50 年计算机经历了电子管、晶体管、集成电路和大规模集成电路四代。1980 年第一代微机“Personal Computer”型出现以来，微机的发展速度非常迅速。目前常用微机的内存可达 256M，CPU 主频达到 1.8GHz 以上，硬盘存储已达 80G。多媒体已经普遍应用。网络计算机的时代已经开始。

计算机的发展已深入到社会各个领域，引起了巨大变革。计算机在化学中的应用也日益广泛，更加深入。我国高等学校和科研单位的实验室中已经普遍配备了各种型号性能优良的微机。

计算机在化学中的应用，归纳起来有以下几个方面：

(1) 化学中的数值计算问题；

(2) 化学中的红外、质谱、核磁、晶体结构，以及各学科试题等大型数据库的建立；

(3) 化学中的文献、情报检索查询；

(4) 化学中的图形学，显示化学中的二维及三维空间图形；

(5) 化学中的专家系统，如有机化合物的解析、合成路线专家系统等；

(6) 化学中的教学与辅导系统，如计算机的远程教学，以及化学各学科的辅导系统。

本书的主要目的是介绍化学中的数值计算问题，也就是用微机来解决化学中遇到的各种各样的数值计算问题。

1.2　计算机解题一般步骤

用计算机解化学中数值计算问题首先要把化学问题模型化，转变为数学公式，选择一种计算方法(包括算术运算和逻辑运算)。这种计算方法常称为算法。然后选择一种高级语言，把上述的算法书写成源程序。这项工作称为程序设计。再把源程序进行编程，调试得到执行程序。最后用已知的算例去试算检查。分析结果正确无误，才能用于未知的算例。归纳起来如下图所示：

1.3 计算机语言和源程序

目前在微机上使用的高级语言种类很多，适于科学计算的语言有FORTRAN，BASIC，PASCAL 以及 C 语言。本书使用目前流行的 C 程序设计语言编程。用全屏幕编辑软件(如 EDIT 或 QEDIT 等)，把计算公式书写成源程序，也称为源代码，使用 Turboc 2.0 编译系统编译，书中全部程序都在 586 微机上 Turboc 2.0 调试通过，形成 EXE 文件可以单独使用。

1.4 算 法 简 介

算法是我们事前选定的计算方案和规划计算步骤。对同一个数值问题有很多算法可选用。首先要选择一种计算量尽量小、占用机器内存储量小、逻辑结构简单的算法。

在算法上尽量采用递推化公式。递推的基本思想是将一个复杂的问题简化为简单的过程多次重复。这种重复在程序上表现为循环，描述很容易。

描述算法方法有很多，其中使用框图方法可以形象描述算法。每个框表示要做的内容，用箭头表示各框执行的顺序。按照框图编制一个源程序更加容易。

1.5 数值计算误差简介

数值计算总是存在各种误差，计算结果总是近似的。误差是不可避免的，根本不存在绝对严格和精确的结果。但在数值计算中我们要求满足给定的精度总是可以实现的。

误差的来源首先决定你选用的数学模型。将化学问题归结为数学问题时，总要加上很多限制，总要忽略一些次要的因素，这样建立的理想化数学模型只是客观化学问题的一种近似。这种近似必然会产生误差，称为模型误差。描述问题的数学模型是近似的，所以要求解法的绝对准确是没有意义的。

在计算数值解中，常常存在对某种无穷过程的“截断”，由此产生的误差称为截断误差。

例如，计算指数函数 e^x 的值，常常用泰勒(Taylor)定理展开成幂级数形式：

$$e^x = 1 + x + \frac{x^2}{2!} + \Lambda + \frac{x^n}{n!} + \Lambda$$

实际计算中，我们只能取有限项之和，而忽略其余各项，这时必然产生截断误差。根据泰勒的余项定理，e^x 的截断误差为：

$$R_n(x)=\frac{x^{n+1}}{(n+1)!}\mathrm{e}^{\theta x},\ 0<\theta<1$$

假定$|x|\le 1$**，要求截断误差小于**5×10^{-3}**，注意到**$\mathrm{e}^{\theta x}\le \mathrm{e}\le 3$**，**

$$|R_n(x)|<\frac{3}{(n+1)!}$$

于是，只要 n=5 **即满足要求。**

数值计算中，另一种误差为舍入误差。受机器字长的限制，用机器代码表示数据的位数有一定限制，由此产生的误差称为舍入误差。少量的运算产生的舍入误差是微不足道的，但千百万次的舍入误差有时是很惊人的。在微机数值计算中的实数尽量采用双精度字长，以减少舍入误差。总之，对任何一项计算，首先必须保证精度。在满足给定的精度下，选择一种好的算法。

第 2 章　代数方程求根

2.1　引　　言

在化学问题中常常遇到代数方程的求根问题。这些问题有时是复杂化学问题的组成部分，有时自身是一个完整的问题。

例 2-1　化学中真实气体服从范德华(van derWaals)方程

$$\left(p+\frac{a}{V^2}\right)(V-b)=RT$$

其中：a，b 是常数；R 是气体常数；p，V，T 三个变量的关系是一个三次方程。已知 p，T，求它的体积 V。

原方程变为

$$pV^3-(pb+RT)V^2+aV-ab=0 \tag{2-1}$$

这是一个求三次方程根的问题。

例 2-2　在分析化学中，已知醋酸 HAc 的浓度为 C，求溶液的 pH 值。

在 HAc 稀溶液中存在

$$\mathrm{HAc} \rightleftharpoons \mathrm{H^+ + Ac^-}\ ,\quad K_a=\frac{[\mathrm{H^+}][\mathrm{Ac^-}]}{[\mathrm{HAc}]} \tag{2-2}$$

$$\mathrm{H_2O} \rightleftharpoons \mathrm{H^+ + OH^-}\ ,\quad K_w=[\mathrm{H^+}][\mathrm{OH^-}] \tag{2-3}$$

溶液中 HAc 浓度　　$C_{\mathrm{HAc}}=[\mathrm{HAc}]+[\mathrm{Ac^-}]$

由电中性原理　　$[\mathrm{H^+}]=[\mathrm{Ac^-}]+[\mathrm{OH^-}]$

把 $[\mathrm{Ac^-}]$，$[\mathrm{HAc}]$ 代入(2-2)式，有

$$K_a=\frac{[\mathrm{H^+}]([\mathrm{H^+}]-[\mathrm{OH^-}])}{C_{\mathrm{HAc}}-[\mathrm{H^+}]+[\mathrm{OH^-}]} \tag{2-4}$$

再把 $[\mathrm{OH^-}]=K_w/[\mathrm{H^+}]$ 代入(2-4)展开变为 $[\mathrm{H^+}]$ 的三次方程

$$[\mathrm{H^+}]^3+K_a[\mathrm{H^+}]^2-(K_aC_{\mathrm{HAc}}+K_w)[\mathrm{H^+}]-K_aK_w=0 \tag{2-5}$$

把 K_a，K_w，C_{HAc} 代入可得出 $[\mathrm{H^+}]$ 的浓度。溶液的 $\mathrm{pH}=-\lg[\mathrm{H^+}]$。

例 2-3　量子化学计算共轭体系的π 电子能级。

根据 HMO 的基本假定，令

$$x=\frac{\alpha-E}{\beta}$$

其中：α 称为库仑积分，β 称为共振积分，E 为π 电子能级。

求解下面久期行列式的根得到丁二烯的π 电子能级。

$$\begin{vmatrix} x & 1 & 0 & 0 \\ 1 & x & 1 & 0 \\ 0 & 1 & x & 1 \\ 0 & 0 & 1 & x \end{vmatrix} = x^4 - 3x^2 + 1 = 0 \tag{2-6}$$

也就变为求解一个多项式的代数方程。

关于代数方程求根问题类型的分类，首先是根据方程的个数，其次是解的个数。代数方程求根的分类如图 2-1 所示。

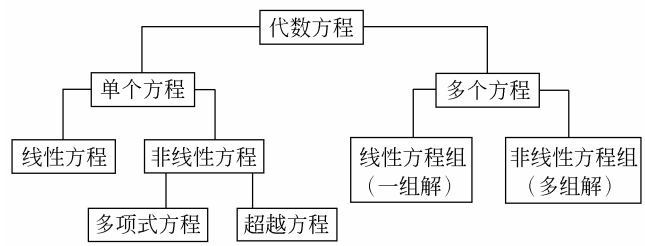

图 2-1

关于多个方程中线性方程组的解在第 6 章讨论，非线性方程组的解在第 7 章讨论，本章主要讨论单个方程的求解问题。

一元非线性方程又分为多项式方程和超越方程。含有 sin$x$，cos$x$，tan$x$ 等三角函数或者含有 lg$x$，e^x 等特殊函数的非线性方程统称为超越方程。

一元非线性方程求解分直接法和间接法。所谓的直接法是用一组公式得到解，如二次方程的求根。间接法是重复使用某种算法求解，得到的解是近似的，但可以达到任意要求的精度。迭代法是最适宜于计算机的间接解法。

2.2 多项式的求值

形如

$$f(x) = a_0 x^n + a_1 x^{n-1} + \Lambda + a_{n-1} x + a_n = 0 \tag{2-7}$$

称为一元 n 次代数方程，$f(x)$ 称为一元 n 次多项式。

如果我们直接计算多项式的值，使用乘法的次数为 $N + N - 1 + \Lambda + 1 = \frac{N(N-1)}{2}$，加法的次数为 N 次。如果把上式用代数结合律改写为

$$f(x) = ((\Lambda\ (a_0 \cdot x + a_1) \cdot x + a_2) \cdot x + \Lambda + a_{n-1}) \cdot x + a_n \tag{2-8}$$

计算上式的值仅用 N 次乘法和 N 次加法。若 $N = 10$，第一种方法用 45 次乘法

和 10 次加法，而第二种方法仅用 10 次乘法和 10 次加法。

多项式的这种有效算法是我国宋代数学家秦九韶最先提出的，外国文献中称霍纳(Horner)算法。霍纳工作较秦九韶工作晚五六个世纪。

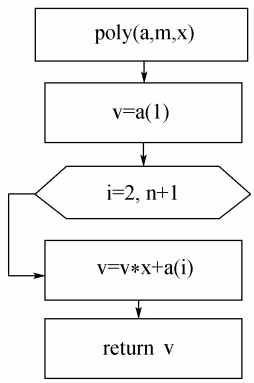

图 2-2

秦九韶算法特点是通过一次式反复计算逐步得到高次多项式的值。用一个递推公式表示

$$\left.\begin{array}{l}(1)\ v_0 = a_0\\(2)\ v_i = v_{i-1}\cdot x + a_i\\(3)\ \ f(x) = v_n\end{array}\right\}\quad (i=1,2,\Lambda\ ,n)\qquad (2\text{-}9)$$

n 次多项式的系数有 n+1 个，存放在 $a[n+1]$，x 为给定值，求 $f(x)$ 值的 poly(a, m, x) 函数框图如图 2-2。poly(a, m, x) 函数参数说明：$a[\ ]$存放多项式的 n+1 个系数，m：多项式系数的个数：x ：x 设定值。

求多项式及一阶导数值的程序如下：

```
/* po.c   */
main( )
{     int i,n;float x,p,a[20];
      float poly(float a[],int m,float x);
      printf( "\nInput n x : ");
      scanf( "%d%f ", &n, &x);
      printf( "\nPlease input (a[i]i=1,n+1: ");
      for(i=1;i <=n+1;i++)
        scanf( "%f ", &a[i]);
      p=poly(a,n+1,x);
      printf( "\n f(x)      =%12.6f\n ",p);
      for(i=1;i<=n;i++)
        a[i]*=(n-i+1);
      p=poly(a,n,x);
      printf( "df(x)/dx     =%12.6f\n ",p);
      for(i=1;i<=n-1;i++)
        a[i]*=(n-i);
      p=poly(a,n-1,x);
      printf( "df(x) ^ 2/dx ^ 2     =%12.6f\n ",p);
      for(i=1;i<=n-2;i++)
        a[i]*=(n-i-1);
      p=poly(a,n-2,x);
      printf( "df(x) ^ 3/dx ^ 3     =%12.6f\n ",p);
```

```
    }
float poly(float a[],int m,float x)
{   int  i;float v;
    v=a[1];
    for(i=2;i<=m;i++)
      v=v*x +a[i];
    return(v);
}
```

计算下面多项式在 x=2 **时的值及一阶、二阶、三阶导数值。**

y=5*x ^ 5+4*x ^ 4+3*x ^ 3+2*x ^ 2+1*x+2.0
y=？ y'=？ y"=？ y"'=？

运行结果如下：

```
Input n x：52.↙
Please input (a[i]i=1，n+1：5.4.3.2.1.2.↙
f(x)                    =   260.000000
df(x)/dx                =   573.000000
df(x) ^ 2/dx ^ 2        =   1032.000000
df(x) ^ 3/dx ^ 3        =   1410.000000
```

如果采用文件形式输入和输出，多项式求值以及求一阶、二阶、三阶导数值的程序和计算例子如下：

```
/* po1.c   */
#include <stdio.h>
main(int argc,char*argv[])
{   FILE*fp1,*fp2;int i,n;float a[20],x,p;
    float poly(float a[],int m,float x);
    if(argc <3)
    {puts("\nUsage:PO1 input_data_filename output_filename\n");
      exit(1);}
    if((fp1=fopen(argv[1],"r"))==0){
      printf("can't open data file\n");exit(1);}
    if((fp2=fopen(argv[2],"w"))==0){
      printf("can't open output file\n");exit(1);}
    fscanf(fp1,"%d%f\n",&n,&x);
    for (i=1;i<=n+1;i++)
      fscanf(fp1,"%f",&a[i]);
    fclose(fp1);
```

```
        fprintf(fp2, "The f(x)given x,f(x)= ? f '(x)= ? f ''(x)= ? f(x) " '= ? \n ");
        pr_line(fp2,80);fprintf(fp2, "f(x)= ");
        for(i=1;i<=n+1;i++)
            {char ch;ch=(i <=n) ? '+ ': '\n ';
              fprintf(fp2, "%.2f*X ^ %d %c ",a[i],n-i+1,ch);}
        fprintf(fp2, "\n ");
        p=poly(a,n+1,x);
        fprintf(fp2, "f(x)\t\t=%12.6f\n ",p);
        for(i=1;i<=n;i++)
           a[i]*=(n-i+1);
        p=poly(a,n,x);
        fprintf(fp2, "df(x)/dxi\t=%12.6f\n ",p);
        for(i=1;i<=n-1;i++)
           a[i]*=(n-i);
        p=poly(a,n-1,x);
        fprintf(fp2, "df(x) ^ 2/dx ^ 2=%12.6f\n ",p);
        for(i=1;i<=n-2;i++)
           a[i]*=(n-i-1);
        p=poly(a,n-2,x);
        fprintf(fp2, "df(x) ^ 3/dx ^ 3=%12.6f\n ",p);
        fclose(fp2);
}
float poly(float a[],int m,float x)
{       int i;float v;
        v=a[1];
        for(i=2;i<=m;i++)
           v=v*x +a[i];
        return(v);
}
pr_line(FILE*fp2,int n)
{       int i;
         for(i=1;i<=n;i++)
            fprintf(fp2, "%c ",0xc4);fprintf(fp2, "\n ");}
```

计算下面多项式在 x=2 **时的值及一阶、二阶、三阶导数值。**

y=x ^ 5+2*x ^ 4-3*x ^ 3+8*x ^ 2-7*x+11.0

x=2y= ? y '= ? y "= ? y " '= ?

Input data file：

5 2

```
1 2 - 3 8 - 7 11
output file：
The f(x) given x，f(x)=？ f '(x)=？ f ''(x)=？ f(x)" '=？
f(x)=1.00*X ^ 5+2.00*X ^ 4-3.00*X ^ 3+8.00*X ^ 2-7.00*X ^ 1+11.00*X ^ 0
f(x)                =    69.000000
df(x)/dx            =   133.000000
df(x) ^ 2/dx ^ 2    =   236.000000
df(x) ^ 3/dx ^ 3    =   318.000000
```

2.3 二 分 法

设函数 $f(x)$ 在某个区间$(a，b)$内仅有一根 x^*，我们取中点 $x_0=\frac{1}{2}(a+b)$，如果 $f(x_0)=0$，则 $\frac{1}{2}(a+b)$ 即为所求的根，否则将区间分为两半，进行根的扫描，即检查 $f(x_0)$ 与 $f(a)$ 是否同号，如果同号，表明所求的根 x^*在 x_0 的右侧区间 (x_0, b)，这时令 $a_1=x_0$，$b_1=b$，否则 x^*在 x_0 的左侧区间$(a，x_0)$，则取 $a_1=a$，$b_1=x_0$，这样得到一个新的根区间$(a_1，b_1)$，其长度仅为区间$(a，b)$长度的一半，见图 2-3。

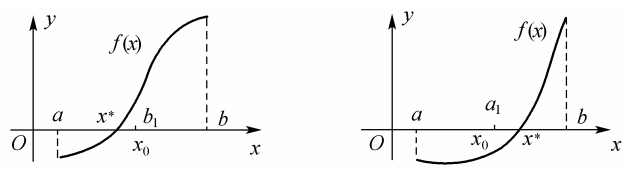

图 2-3

对压缩了有根区间$(a_1，b_1)$再二分 $x_1=\frac{1}{2}(a_1+b_1)$，检查 $f(x_1)$ 是否等于零，否则再检查 $f(x_1)$ 与 $f(a)$ 的符号，又可得到新的根区间 (a_2, b_2)，依此反复二分下去，直到有根区间 (a_k, b_k) 的长度小于给定的精度 ε，则 $x_k=\frac{1}{2}(a_k+b_k)$ 即为满足精度 ε 的 $f(x)=0$ 的根。值得注意的是每次二分后，$f(x_k)$ 只要和 $f(a)$ 比较符号即可，因为 $f(a_k)$ 和 $f(a)$ 永远是同符号。

二分法得到一个根的序列

$$x_0, x_1, x_2, \cdots, x_k, \cdots$$

该序列的极限必为根 x^*。二分法是一个逐步逼近的方法，一旦初始区间已知，经过 k 次二分后，区间长度缩小为原来的 $2^{-(k+1)}$。二分法误差估计式为

$$|x^* - x^k| \le \frac{1}{2^{k+1}}(b-a) \tag{2-10}$$

用误差估计式(2-10)可以估计二分的次数

$$k \le \ln\frac{b-a}{|x^* - x^k|} \Big/ \ln 2 - 1 \tag{2-11}$$

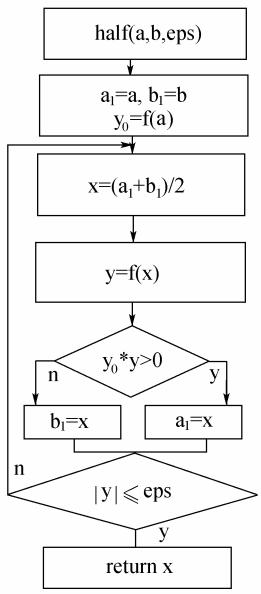

图 2-4

二分法函数的框图见图 2-4。二分法函数参数说明：f，求根的函数；a，根区间的左端点；b，根区间的右端点；eps，根的允许误差。

例 2-4 求方程 $x^6 - x^4 - x^3 - 1 = 0$ 在区间(1，2)内的实根，要求计算精度为 10^{-6}。

解 用二分法。这里 $a = 1.0$，$b = 2.0$，且 $f(a) < 0$，取区间中点 $x_0 = 1.5$，将区间二等分。$f(x_0) > 0$，即 $f(x_0)$ 与 $f(a)$ 异号，故所求的根必在 x_0 左侧区间 $(a,\ x_0)$，则 $a_1 = 1.0$，$b_1 = 1.5$，得到新的根区间 $(a_1,\ b_1)$。对新的根区间再二等分得 $x_1 = 1.25$，$f(x_1) < 0$，由于 $f(x_1)$ 和 $f(a)$ 同号，故所求的根必在 x_0 右侧区间 $(x_1,\ b_1)$，则 $a_2 = 1.25$，$b_2 = 1.5$，得到新的根区间 $(a_2,\ b_2)$。如此反复二分下去，直到满足计算精度。各步的 $(a_k,\ b_k)$，x_k 和 $f(x)$ 见表 2-1。由(2-11)式计算得到 $k \approx 19$。

表 2-1 例 2-4 的计算结果

IT	a_k	b_k	x_k	$y=f(x)$	IT	a_k	b_k	x_k	$y=f(x)$
0	1.000000	2.000000	1.500000	1.953	11	1.403320	1.403809	1.403564	–5.920E–04
1	1.000000	1.500000	1.250000	–1.580	12	1.403564	1.403809	1.403687	1.327E–03
2	1.250000	1.500000	1.375000	–4.161E–01	13	1.403564	1.403687	1.403625	3.672E–04
3	1.375000	1.500000	1.437500	5.831E–01	14	1.403564	1.403625	1.403595	–1.124E–04
4	1.375000	1.437500	1.406250	4.191E–02	15	1.403595	1.403625	1.403610	1.274E–04
5	1.375000	1.406250	1.390625	–1.969E–01	16	1.403595	1.403610	1.403603	7.468E–06
6	1.390625	1.406250	1.398438	–8.005E–02	17	1.403595	1.403603	1.403599	–5.248E–05
7	1.398438	1.406250	1.402344	–1.971E–02	18	1.403599	1.403603	1.403601	–2.251E–05
8	1.402344	1.406250	1.404297	1.094E–02	19	1.403601	1.403603	1.403602	–7.519E–06
9	1.402344	1.404297	1.403320	–4.425E–03	20	1.403602	1.403603	1.403602	–2.536E–08
10	1.403320	1.404297	1.403809	3.247E–03	x = 1.403602 $f(x)$= –2.536E–08				

一个完整的计算例 2-4 的程序如下：

```
/* binbis1.c  */
#include <math.h>
```

```
main()
{     int i;double f(),half();
       double a,b,x,eps=1e-06;
       printf("\nPlease input a b: ");
       scanf("%lf%lf",&a,&b,);
       printf("IT\t   A1\t   B1\t   X\t   f(x)\n");
       pr_line(58);
       x=half(f,a,b,eps);
       pr_line(58);
       printf("\nApproximate root x=%12.6f\n",x);
       printf("Function value of f(x)=%12.4f\n",x);
}
double x;
{   double x3;
    x3=x*x*x;
    return(x3*(x3-x-1.0)-1.0);
    }
double half(f,a,b,eps)
double (*f)();double a,b,eps;
{   int it=0;
    double x,a1,b1,y0,y;
    a1=a;b1=b;y0=(*f)(a);
    do {
         x=(a1+b1)/2.0;
         printf("%2d   %12.6f   %10.6f   %10.6f   %12.4E\n",
                it,a1,b1,x,f(x));
         y=(*f)(x);
         if(y*y0>0)a1=x;
         else b1=x;
         it++;
      }  while(fabs(y )>eps);
  return(x);
}
pr_line(int n)
{     int i;
       for(i=0;i<n;i++)
       printf("%c",0xc4);printf("\n");
}
       Please input a b: 1. 2.↙
```