



C 语言数值算法

张圣华编著

海洋出版社



151A

北京希望电脑公司计算机技术丛书

C 语言数值算法

张圣华 编著

海洋出版社

1993 年·北京

内 容 简 介

本书包括了162个工程上常用的数学数值算法的C语言程序。内容包括：复数运算，插值，拟合与平滑，数值积分，线性代数方程组的求解，行列式求值及矩阵求逆，特征值与特征向量的计算，多项式方程的求解，超越方程的求解，线性代数方程组的求解，常微分方程的数值积分，特殊函数，数学变换，概率统计等。

所有这些程序均在IBM-PC系列及其兼容机上用Microsoft C 6.0调试通过，
本书可供计算机工作者、科技人员及高校师生参考。

需要本书的用户，可直接与北京8721信箱联系，邮码：100080，电话：2562329。

(京)新登字087号

C 语言数值算法

张圣华 编著

*

海洋出版社出版(北京市复兴门外大街1号)

海洋出版社发行 北京四季青印刷厂印刷

开本：787×1092 1/16 印张：38 字数：840千字

1993年8月第一版 1993年8月第一次印刷

印数：1—5000册

*

ISBN 7-5027-3528-3/TP·195 定价：35.00元

前 言

数值计算在计算机的应用领域起着重要的作用。它不仅在科技和工程领域中扮演着重要角色,在当前微机迅猛发展的形势下,很多非科技与工程领域的应用软件中也要用到这其中的算法。如最小二乘法、三次样条函数插值、曲线拟合等算法应用得就非常广泛。

80年代C语言取得了瞩目的发展,C语言从一个小的系统程序设计语言成长为广为人知的通用程序设计语言。C语言由于它的紧凑、高效、易于移植、模块化等特色已被广泛地应用于系统程序设计、软件工具、图像处理、科学计算、数值分析、人工智能、数据库管理系统设计、文字处理等领域,而且越来越为广大程序员所喜爱。它的影响已达到了如此的地步:即能否使用C语言来编程是区分真正的计算机专业人员和一般计算机用户的分水岭。就各种编程语言来看,C语言的编程工具的发展是最快的。Microsoft公司和Borland公司不断地更新它们的C语言编程工具。这对广大程序员具有极大的吸引力。目前,C语言是唯一可以向具有统治地位的科学计算语言FORTRAN挑战的语言,C语言在很多方面都要强于FORTRAN语言。

由于C语言流行时间不长,比起流行了几十年的FORTRAN和BASIC语言来,有关C语言算法的书实在太少,尤其是在数值计算方面,这使C语言程序员颇感不便。因此特编此书,以供广大读者参考之用。

书中共给出了162个算法,其中每个算法中,都给出了算法概要、完整的C语言算法函数以及其应用示例。程序都是用Microsoft C6.0编写的,并都经过上机调试通过。这些程序绝大多数不需要修改就可适用于Turbo C。

目 录

前言

第一章 导 论

1.1 C 语言的数学库函数	1
1.2 动态数组	2
1.3 C 语言中的复数表示	3
1.4 使用双精度数据类型	3
1.5 堆栈溢出问题	4

第二章 复数运算

2.1 复数的模	5
2.2 复数除法	6
2.3 复数的根	7
2.4 复数的实幂指数	10
2.5 复数的复幂指数	11
2.6 复变量的自然对数	13
2.7 复变量的正弦	15
2.8 复变量的余弦	17
2.9 复变量的正切	19

第三章 插值

3.1 拉格朗日一元 n 点插值	22
3.2 线性插值	24
3.3 一元三点插值	26
3.4 牛顿插值	28
3.5 埃特金插值	30
3.6 厄米特插值	33
3.7 三次自然样条函数插值	35
3.8 第一种边界条件的样条函数插值、微商和积分	39
3.9 第二种边界条件的样条函数插值、微商和积分	45
3.10 第三种边界条件的样条函数插值、微商和积分	51
3.11 有理函数插值	57
3.12 分段有理函数插值	61
3.13 拉格朗日二元 n 点插值	65
3.14 二元拟线性插值	70
3.15 二元三点插值	73

第四章 拟合与平滑

4.1 五点三次平滑	77
4.2 样条函数平滑	79
4.3 切比雪夫曲线拟合	89
4.4 最小二乘法曲线拟合	94
4.5 最小二乘法曲面拟合	99

第五章 数值积分

5.1 定步长辛普生积分	106
5.2 变步长辛普生积分	107
5.3 自适应辛普生积分	110
5.4 变步长辛普生求二重积分	113
5.5 切比雪夫求积分	117
5.6 龙贝格求积分	120
5.7 高斯法求积分(I)	124
5.8 高斯法求积分(II)	126
5.9 高斯法求多重积分	129

第六章 线性代数方程组的求解

6.1 高斯消去法	135
6.2 列主元高斯消去法(I)	139
6.3 列主元高斯消去法(II)	142
6.4 全主元高斯消去法	144
6.5 克劳特分解法	148
6.6 求线性对称方程组的分解法	151
6.7 高斯-赛德尔迭代法	155
6.8 对称方程组的平方根法(I)	158
6.9 对称方程组的平方根法(II)	161
6.10 带型方程组的列主元消去法	165
6.11 对称带型方程组的解法	168
6.12 大型稀疏对称正定方程组的解法	172
6.13 大型稀疏方程组的求解	176
6.14 解三对角型方程组的追赶法	180
6.15 全选主元高斯-约当法	183
6.16 托布利兹方程组的解法	187
6.17 共轭斜量法	194
6.18 复系数方程组的列主元消去法	198

6.19 复系数方程组的全选主元高斯消去法	201
第七章 行列式求值及矩阵求逆	
7.1 全主元高斯消去法求行列式值	207
7.2 对称正定矩阵的乔里斯基分解法求行列式值(I)	209
7.3 对称正定矩阵的乔里斯基分解法求行列式值(II)	212
7.4 行主元消去法求逆矩阵及行列式值	214
7.5 全主元高斯-约当消去法求逆	219
7.6 对称正定矩阵的求逆(I)	222
7.7 对称正定矩阵的求逆(II)	226
7.8 高斯-约当消去法解线性方程组、求系数矩阵的逆及行列式的值	229
7.9 全主元高斯-约当消去法解线性方程组、求系数矩阵的逆及行列式的值 ..	231
7.10 托布利兹矩阵的求逆	236
7.11 叶尔绍夫法求逆矩阵	240
7.12 复矩阵求逆的全主元高斯-约当消去法	243
第八章 特征值与特征向量的计算	
8.1 求实对称矩阵的特征值及特征向量的雅可比方法	248
8.2 用豪斯荷尔德变换法化对称阵为三对角阵(I)	254
8.3 用豪斯荷尔德变换法化对称阵为三对角阵(II)	258
8.4 用豪斯荷尔德变换法化对称阵为三对角阵(III)	262
8.5 实对称的三对角矩阵的 QL 方法求特征值	265
8.6 实对称三对角矩阵的 QL 方法求特征值及特征向量	271
8.7 实对称三对角矩阵的二分法求特征值	275
8.8 对称带型矩阵的三角化	280
8.9 广义特征值($Ax = \lambda Bx, ABx = \lambda x$ 等)问题的简化	285
8.10 初等相似变换法化一般实矩阵为赫申伯格型矩阵	291
8.11 豪斯荷尔德法化一般实矩阵为赫申伯格型矩阵	294
8.12 化一般复数矩阵为赫申伯格型矩阵	297
8.13 求实赫申伯格矩阵特征值 QR 算法	301
8.14 求实矩阵的特征值及特征向量	308
8.15 用改进的 LR 算法求复赫申伯格矩阵的特征值	321
8.16 求复矩阵的特征值及特征向量	327
第九章 求解多项式方程	
9.1 霍纳法求多项式值及其导数值	338
9.2 直接公式法解二次方程	340
9.3 直接公式法解三次方程	342

9.4 直接公式法解四次方程	346
9.5 贝努利法	350
9.6 牛顿-麦劳奇法	353
9.7 林士谔-赵访熊法	356
9.8 牛顿-下山法	362
9.9 QR 方法	367
9.10 切线法	370
9.11 割线法	373
第十章 求解超越方程	
10.1 牛顿法	378
10.2 二分法	380
10.3 弦截法	383
第十一章 求解非线性方程组	
11.1 梯度法	386
11.2 拟牛顿法	389
11.3 改进的牛顿法	396
11.4 布罗伊登法	403
第十二章 常微分方程的数值积分	
12.1 定步长欧拉方法	409
12.2 变步长欧拉方法	412
12.3 定步长维梯方法	415
12.4 定步长龙格-库塔方法	419
12.5 全区间积分的定步长龙格-库塔方法	421
12.6 变步长龙格-库塔方法	424
12.7 积分一步的变步长基尔法	428
12.8 全区间积分的变步长基尔法	433
12.9 积分一步的变步长库塔-墨森法	437
12.10 全区间定步长阿当姆斯预报-校正方法	441
12.11 全区间定步长哈明方法	444
12.12 双边法	448
12.13 积分一步的特雷纳法	452
12.14 全区间积分的特雷纳法	458
12.15 吉尔法	462
第十四章 特殊函数	
14.1 Γ 函数	480

14.2	Γ 函数的自然对数	482
14.3	第一类和第二类完全椭圆积分	484
14.4	任意整数阶第一类贝塞尔函数	486
14.5	任意整数阶球贝塞尔函数	488
14.6	球诺伊曼函数	491
14.7	第一类连带勒让德函数	492
14.8	切比雪夫多项式	496
14.9	厄米特多项式	498
14.10	拉盖尔多项式	500
14.11	勒让德多项式	502
14.12	正态分布函数	505
14.13	指数积分 (I)	508
14.14	指数积分 (II)	510
14.15	定指数积分 (I)	512
14.16	定指数积分 (II)	513
14.17	正弦积分	516
14.18	余弦积分	518
14.19	实误差函数	520
14.20	复误差函数	522
14.21	高斯误差函数	526
第十四章 数学变换		
14.1	傅里叶级数的系数	529
14.2	一维快速傅里叶变换	532
14.3	二维快速傅里叶变换	537
14.4	快速沃什变换	540
第十六章 概率统计		
16.1	0~1 间均匀分布随机数的产生	544
16.2	0~1 间均匀分布随机数序列的产生	545
16.3	任意区间均匀分布随机整数序列的产生	547
16.4	满足正态分布随机数序列的产生	549
16.5	正态分布的上概率	551
16.6	正态分布的百分点	553
16.7	χ^2 分布的上概率	555
16.8	χ^2 分布的百分点	559
16.9	F 分布的上概率	563

16.10	F 分布的百分点	566
16.11	t 分布的上概率	568
16.12	t 分布的百分点	570
16.13	贝塔分布 ($\frac{1}{2}$ 整数倍) 的分布函数	573
16.14	贝塔分布 ($\frac{1}{2}$ 整数倍) 的百分点	576
16.15	T 函数	580
16.16	数学期望和自协方差函数值的估计	584
16.17	数学期望和互协方差函数值的估计	588
16.18	一元线性回归分析	591
16.19	多元线性回归分析	595

第一章 导 论

1.1 C 语言的数学库函数

下面给出了 Microsoft C 6.0 的数学库函数, 它们的大多数都是采用 double 数据类型, 其中以 *l* 结尾的函数表示支持 80 位数据类型。有关这些函数的详细情况可参考其使用手册。

<i>acos, acosl</i>	计算反余弦
<i>asin, asinl</i>	计算反正弦
<i>atan, atanl</i>	计算反正切
<i>atan2, atan2l</i>	计算反正切
<i>cabs, cabsl</i>	计算复数的绝对值
<i>ceil, ceil</i>	求出上限整数
<i>cos, cosl</i>	计算余弦
<i>cosh, coshl</i>	计算双曲余弦
<i>div, ldiv</i>	整数相除, 返回商和余数
<i>exp, expl</i>	计算指数函数
<i>fabs, fabsl</i>	求绝对值
<i>floor, floorl</i>	求小于等于变量的最大整数
<i>fmod, fmodl</i>	求浮点余数
<i>frexp, frexpl</i>	计算一指数值
<i>hypot, hypotl</i>	计算直角三角形的斜边
<i>j0, j0l</i>	计算 0 阶第一类 Bessel 函数
<i>j1, j1l</i>	计算 1 阶第一类 Bessel 函数
<i>jn, jnl</i>	计算 n 阶第一类 Bessel 函数
<i>ldexp, ldexpl</i>	计算变量与 2^{exp} 的积
<i>log, logl</i>	计算自然对数
<i>log10, log10l</i>	计算以 10 为底的对数
<i>max, min</i>	返回两个数值中较大或较小的值
<i>modf, modfl</i>	把变量分解为整数部分和小数部分
<i>pow, powl</i>	计算给定值的方幂
<i>sin, sinl</i>	计算正弦
<i>sinh, sinhl</i>	计算双曲正弦
<i>sqrt, sqrtl</i>	求平方根
<i>y0, y0l</i>	计算 0 阶第二类 Bessel 函数
<i>y1, y1l</i>	计算 1 阶第二类 Bessel 函数
<i>yn, ynl</i>	计算 n 阶第二类 Bessel 函数

1.2 动态数组

在编制一个通用的子程序时,若其中涉及到数组问题,最好采用动态数组,其优点是显而易见的。若采用传统的静态数组,我们不得不把数组设置得大些,以满足各种不同的需要。例如,在处理矩阵问题时,通常要用二维数组来存放矩阵元素,若在程序中设置了一个 50×50 的二维数组,则当所要处理的矩阵只是 2×2 阶矩阵时,就造成了内存资源的浪费,而遇到 51×51 阶矩阵时,程序又无能为力。解决这一问题的根本办法就是采用动态数组。

对于一维动态数组, C 语言中提供了一个内存分配函数`calloc`,可以用它来为一维数组动态分配内存空间, `calloc`为一维数组分配空间时,会把个元素初始化为0。当此数组不再需要时,可以用`free`函数来释放所分配的空间。例如,我们要为一 n 阶一维`double`型数组`x[n]`分配空间,则可表示如下:

```
double *x;

x=(double *)calloc(n, sizeof(double));
if( x == NULL )
    return( FALSE );
.
.
.
free( x );
```

在进行动态数组分配时,一定要检查分配是否成功。当`calloc`函数返回`NULL`时,则表示分配失败。

对于二维动态数组, C 语言中并没有给出相应的函数,下面这段程序可以实现二维($r \times c$)阶动态数组的分配和释放:

```
double **TwoArrayAlloc(int r, int c)
{
    double *x, **y;
    int n;

    x=(double *)calloc(r*c, sizeof(double));
    y=(double **)calloc(r, sizeof(double *));
    for(n=0; n<=r-1; ++n)
        y[n]=&x[c*n];
    return( y );
}

void TwoArrayFree(double **x)
{

```

```
    free(x[0]);  
    free(x);  
}
```

这两段程序在以后的各章节中经常要用到。

1.3 C 语言中的复数表示

在C语言中, 没有复数型数据类型, 但我们可以用C语言中的结构来定义一复数。在Microsoft C 6.0 中已给出了这样的结构

```
struct complex {  
    double x;  
    double y;  
}
```

我们在程序中也可自己来定义这样的结构。有了它, 我们就可以如下定义一个复变量 z :

```
struct complex z;
```

对其赋值可以有两种方法(如 $z = 3 + i4$):

```
struct complex z={3, 4};
```

或

```
z.x=3;   z.y=4;
```

类似地, 我们也可以定义一个复变量数组, 如

```
struct complex z[5];  
struct complex z[5][5];
```

动态复变量数组的分配方法如下:

```
struct complex *z;  
  
z=(struct complex *)calloc(5, sizeof(struct complex));
```

1.4 使用双精度数据类型

在Microsoft C 6.0 中, 所有涉及浮点运算的数学库函数都使用double型数据, 鉴于此, 在我们的所有程序中也使用了double型数据。

在Microsoft C 6.0 中, 若在程序中有类似下列的语句

```
float x;  
x=sin(1);
```

并且使用了/W3 或/W4 选择项进行编译, 则会显示下列警告信息

```
warning C4136: conversion between different floating types
```

这表示在等号的两边使用了不同的浮点类型数据, `sin(1)` 的返回值为 `double` 型, 而 `x` 为 `float` 型, 当把 `sin(1)` 的值赋与 `x` 时, C 会自动作类型转换, 首先要将 `sin(1)` 的值变成 `float` 型, 然后再赋给 `x`, 这样有可能会丢失数据的有效位数。

1.5 堆栈溢出问题

在一些程序中, 由于静态数据量太大, 超过了 Microsoft C 6.0 编译器的缺省栈设置值 2048 字节, 则在程序执行时会出现如下的信息

```
stack overflow
```

在这种情况下, 你可以在连接程序时, 用 Link 的 /ST 选择项来改变栈设置值, 如

```
link /ST:4096
```

但是, 最大值不能超过 64KB。

第二章 复数运算

2.1 复数的模

2.1.1 功能

本子程序用于求复数 $z = x + iy$ 的模 $|z|$ 。

2.1.2 方法简介

$$|z| = \sqrt{x^2 + y^2} = \begin{cases} |x| \sqrt{1 + \left(\frac{y}{x}\right)^2} & |x| > |y| \\ |y| \sqrt{1 + \left(\frac{x}{y}\right)^2} & |x| \leq |y| \end{cases}$$

后一等式是为了防止上溢或下溢。

2.1.3 子程序及其使用说明

参数表

z 结构型变量, 输入参数, 存放复数 z 。

子程序

```
double CABS(struct complex z)
{
    z.x = fabs( z.x );
    z.y = fabs( z.y );
    if( z.x == 0 )
        return( z.y );
    else if( z.y == 0 )
        return( z.x );
    else
    {
        if( z.x > z.y )
            return( z.x * sqrt(1+(z.y/z.x)*(z.y/z.x)) );
        else
            return( z.y * sqrt(1+(z.x/z.y)*(z.x/z.y)) );
    }
}
```

2.1.4 例题

求复数 $z = 3 + 4i$ 的模。

程序:

```
#include <stdio.h>
#include <math.h>

double CABS(struct complex z);

void main()
```

```

{
    struct complex z = {3, 4};

    printf("%.10e\n", CABS(z));
}

```

计算结果: $|z| = 5$.

2.2 复数除法

2.2.1 功能

本子程序用于求复数 z_1 除以 z_2 的商 z 。

2.2.2 方法简介

设 $z_1 = a + ib$, $z_2 = c + id$, $z = x + iy$, 则

$$z = \frac{z_1}{z_2} = \frac{(ac + bd) + i(bc - ad)}{c^2 + d^2} = \begin{cases} \frac{\left(a + b\frac{d}{c}\right) + i\left(b - a\frac{d}{c}\right)}{c + d\frac{d}{c}} & |c| \geq |d| \\ \frac{\left(a\frac{c}{d} + b\right) + i\left(b\frac{c}{d} - a\right)}{c\frac{c}{d} + d} & |c| < |d| \end{cases}$$

后一公式是为了防止在 a, b, c, d 很大时或很小时用前一公式计算可能出现的上溢或下溢。

2.2.3 子程序及其使用说明

参数表

- z_1 结构型变量, 输入参数, 存放被除数。
- z_2 结构型变量, 输入参数, 存放除数。
- z 结构型变量, 输出参数, 存放商。

子程序

```

CDIV(struct complex z1, struct complex z2, struct complex *z)
{
    double e, f;

    if( fabs( z2.x ) >= fabs( z2.y ) )
    {
        e = z2.y / z2.x;
        f = z2.x + e * z2.y;
        z->x = ( z1.x + z1.y * e ) / f;
        z->y = ( z1.y - z1.x * e ) / f;
    }
    else
    {
        e = z2.x / z2.y;
        f = z2.y + e * z2.x;

```

```

    z->x = ( z1.x * e + z1.y ) / f;
    z->y = ( z1.y * e - z1.x ) / f;
}
}

```

2.2.4 例题

求 $z_1 = 10^{10} + i2 \times 10^{10}$ 除以 $z_2 = 3 \times 10^{16} + i3 \times 10^{15}$ 的商。

程序:

```

#include <stdio.h>
#include <math.h>

int CDIV(struct complex, struct complex, struct complex *);

void main()
{
    struct complex z1 = {10e+10, 2*10e+10},
                  z2 = {3*10e+16, 3*10e+15},
                  z;

    CDIV(z1, z2, &z);
    printf("%.10e+i%.10e\n", z.x, z.y);
}

```

计算结果: $z = 3.9603960396e - 007 + i6.2706270627e - 007$

2.3 复数的根

2.3.1 功能

本子程序用于求复数的全部 n 次根 $z = x + iy = z_1^{1/n} = (a + ib)^{1/n}$ 。

2.3.2 方法简介

$$\begin{aligned}
 z &= (a + ib)^{1/n} = [r(\cos \phi + i \sin \phi)]^{1/n} \\
 &= r^{1/n} \left(\cos \frac{\phi + 2k\pi}{n} + i \sin \frac{\phi + 2k\pi}{n} \right) \\
 &= r^{1/n} (\cos(\theta + k\theta_1) + i \sin(\theta + k\theta_1)) \quad k = 0, 1, \dots, n-1.
 \end{aligned}$$

其中

$$\theta = \begin{cases} 0 & a > 0, b = 0 \\ \pi/n & a < 0, b = 0 \\ \pi/(2n) & a = 0, b > 0 \\ -\pi/(2n) & a = 0, b < 0 \\ \arctan(b/a)/n & a > 0, b \neq 0 \\ (\arctan(b/a) + \pi)/n & a < 0, b \neq 0 \end{cases} \quad \theta_1 = \frac{2\pi}{n}$$