

徐士良 编著
孙甲松



科学计算 通用 程序集

辽宁科学技术出版社

科学计算通用程序集

徐士良 孙甲松 编著

辽宁科学技术出版社

· 沈阳 ·

图书在版编目(CIP)数据

科学计算通用程序集/徐士良,孙甲松编著. —沈阳:
辽宁科学技术出版社,1997.6
ISBN 7-5381-2468-3

I. 科… I. ①徐… ②孙… II. 科学计算程序 N. TP
319

中国版本图书馆 CIP 数据核字(96)第 25156 号

JS163/26

辽宁科学技术出版社出版
(沈阳市和平区北一马路 108 号 邮政编码 110001)
地方国营新民印刷总厂印刷 新华书店北京发行所发行

开本:787×1092 1/16 印张:22 $\frac{3}{4}$ 字数:500,000
1997 年 6 月第 1 版 1997 年 6 月第 1 次印刷

责任编辑:马旭东 版式设计:于 浪
封面设计:邹君文 责任校对:东 戈

印数:1—5 000 定价:30.00 元

前 言

C 语言已为广大计算机应用工作者所喜爱。用 C 语言编制科学计算程序也已越来越普遍。但对于绝大多数的工程技术人员,特别是专门从事科学计算的计算工作者,其主要精力是放在设计计算方案、选择行之有效的算法上,然后是直接使用算法程序集中的有关程序。本书就是为了满足这种需要而编写的。

本书是整个丛书中的一本,虽然有些内容,如优化问题、数理统计、平滑滤波等,也属于科学计算的范畴,但由于已经归入与本书同时出版的其他选题之中,因此,这些内容也就不再收入本书。

本书共分十章。

第一章是实系数与复系数多项式的运算。主要包括多项式的求值、相乘、相除以及变量的线性代换。

第二章是复数的各种运算。主要包括复数的相乘、相除、 n 次方根、 n 次幂、正弦、指数、对数与幂指数。

第三章是矩阵运算。主要包括矩阵相乘、线性组合、求逆、求广义逆、各种分解以及求行列式值。

第四章是求解线性代数方程组以及线性最小二乘问题。

第五章是计算矩阵特征值与特征向量。主要包括 QR 方法与 Jacobi 方法。

第六章是求解非线性方程与方程组。其中求解非线性方程有对分法、连分式法、QR 方法、Newton 下山法;求解非线性方程组有梯度法、拟 Newton 法。最后还介绍了无约束条件下优化问题的广义逆法。

第七章是求解常微分方程组初值问题的各种数值解法。

第八章是各种等距与不等距插值法,还包括三种边界条件下的三次样条函数插值以及二元函数插值。

第九章是各种数值积分法,其中还包括了计算半无限与无限区间的积分方法以及用 Monte-Carlo 法计算积分。

第十章包括最小二乘曲线、曲面拟合以及 Chebyshev 曲线拟合,还包括最佳一致逼近的 Remez 方法。

书中许多算法是新算法,所有算法函数程序都是行之有效的,读者可以根据自己的需要和问题的性质从中选择最合适的算法程序。

由于水平所限,书中难免有不妥甚至错误之处,恳请读者批评指正。

作 者

1996.9

目 录

前 言

第一章 多项式	1
1.1 实系数多项式求值	1
1.2 复系数多项式求值	2
1.3 实系数多项式相乘	4
1.4 复系数多项式相乘	6
1.5 实系数多项式相除	8
1.6 复系数多项式相除	10
1.7 多项式的变量线性代换	13
1.8 最大公约多项式	15
第二章 复数运算	18
2.1 复数相乘	18
2.2 复数相除	19
2.3 复数的 n 次方根	21
2.4 复数的 n 次幂	23
2.5 复数的正弦	25
2.6 复数的指数	27
2.7 复数的对数	28
2.8 复数的幂指数	30
第三章 矩阵运算	33
3.1 实矩阵线性组合	33
3.2 复矩阵线性组合	34
3.3 实矩阵相乘	36
3.4 复矩阵相乘	38
3.5 行列式值	41
3.6 实矩阵求逆	43
3.7 复矩阵求逆	47
3.8 对称正定矩阵的求逆	52

3.9	Toeplitz 矩阵的求逆	55
3.10	矩阵的三角分解	60
3.11	对称正定矩阵的 Cholesky 分解与行列式值	62
3.12	矩阵的 QR 分解	65
3.13	矩阵的奇异值分解	69
3.14	矩阵的广义逆	84
第四章	线性代数方程组	88
4.1	实系数方程组的 Gauss 消去法	88
4.2	实系数方程组的 Gauss—Jordan 消去法	91
4.3	复系数方程组的 Gauss 消去法	95
4.4	复系数方程组的 Gauss—Jordan 消去法	99
4.5	三对角线方程组的追赶法	103
4.6	一般带型方程组	105
4.7	对称方程组	110
4.8	对称正定方程组	113
4.9	Toeplitz 方程组	116
4.10	线性最小二乘问题的 Householder 变换法	119
4.11	线性最小二乘问题的广义逆法	122
第五章	矩阵特征值与特征向量	125
5.1	求实对称矩阵全部特征值与特征向量的 Householder 变换法	125
5.2	求一般实矩阵全部特征值的 QR 方法	132
5.3	求实对称矩阵全部特征值与特征向量的 Jacobi 法	139
第六章	非线性方程与方程组	144
6.1	对分法求实根	144
6.2	连分式法求一个实根	147
6.3	求实系数多项式方程全部根的 QR 方法	150
6.4	求实系数多项式方程全部根的 Newton 下山法	152
6.5	求复系数多项式方程全部根的 Newton 下山法	158
6.6	求非线性方程组一组实根的梯度法	164
6.7	求非线性方程组一组实根的拟 Newton 法	167
6.8	求解无约束条件下优化问题的广义逆法	171
第七章	常微分方程 (组)	178
7.1	Euler 方法	178
7.2	Witty 方法	181

7.3	Runge—Kutta 法	184
7.4	Gill 方法	188
7.5	Merson 方法	192
7.6	连分式法	196
7.7	双边法	201
7.8	Hamming 方法	205
7.9	Adams 方法	209
7.10	Treanor 方法	213
7.11	Gear 方法	218
7.12	二阶微分方程边值问题	235
第八章 插值		240
8.1	Lagrange 等距插值	240
8.2	Lagrange 不等距插值	242
8.3	一元三点等距插值	244
8.4	一元三点不等距插值	247
8.5	连分式等距插值	249
8.6	连分式不等距插值	251
8.7	Hermite 等距插值	254
8.8	Hermite 不等距插值	256
8.9	Aitken 等距插值	258
8.10	Aitken 不等距插值	260
8.11	Akima 等距插值	263
8.12	Akima 不等距插值	268
8.13	第一种边界条件的三次样条插值	272
8.14	第二种边界条件的三次样条插值	278
8.15	第三种边界条件的三次样条插值	282
8.16	二元三点插值	289
8.17	二元全区间插值	292
第九章 数值积分		296
9.1	变步长梯形求积法	296
9.2	变步长 Simpson 求积法	298
9.3	Romberg 求积法	300
9.4	自适应梯形求积法	302
9.5	Gauss—Legendre 求积法	305
9.6	Gauss—Laguerre 求积法	307
9.7	Gauss—Hermite 求积法	309

9.8	高振荡函数求积法	311
9.9	连分式法	314
9.10	Chebyshev 求积法	316
9.11	Monte Carlo 法	319
9.12	变步长 Simpson 二重积分法	320
9.13	二重积分的连分式法	323
9.14	多重积分的 Gauss 方法	327
9.15	多重积分的 Monte Carlo 法	331
第十章	拟合与逼近	334
10.1	最小二乘曲线拟合	334
10.2	Chebyshev 曲线拟合	338
10.3	矩形域的最小二乘曲面拟合	342
10.4	最佳一致逼近的 Remez 方法	350
参考文献		356

第一章 多项式

1.1 实系数多项式求值

【功能】

计算实系数多项式

$$P(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0$$

在指定点 x 处的函数值。

【方法说明】

递推计算公式为

$$r = a_{n-1}$$

$$r = rx + a_k, k = n-2, \dots, 1, 0$$

最后得到的 r 即是所求的多项式值 $P(x)$ 。

【函数语句与形参说明】

double rpoly (a, r, x)

形参与函数名类型	参 数 意 义
double a [n]	存放多项式系数。
int n	多项式的项数。
double x	自变量值。
double rpoly	返回多项式值。

【函数程序】

```
/* ch01-01.c */
double rpoly (a, n, x)
double a [], x;
int n;
{ double r;
  int i;
  r = a [n-1];
  for (i = n-2; i >= 0; i--)
    r = r * x + a [i];
  return (r);
}
```

【例】

计算多项式

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

在 $x = \pm 0.9, \pm 1.1, \pm 1.3$ 处的函数值。

主函数程序

```
/* ch01-01e.c */
#include <stdio.h>
#include "ch01-01.c"
main ( )
{ double a [7] = {-20.0, 7.0, -7.0, 1.0, 3.0, -5.0, 2.0};
  double x [6] = {0.9, -0.9, 1.1, -1.1, 1.3, -1.3};
  int i;
  for (i=0; i<6; i++)
  { double y;
    y=rpoly (a, 7, x [i]);
    printf (" x [%d] = %5.2f x [%d] = %12.6e\n", i, x [i], i, y);
  }
}
```

运行结果

```
x [0] = 0.90 x [0] = -1.85623e+01
x [1] = -0.90 x [1] = -2.67154e+01
x [2] = 1.10 x [2] = -1.95561e+01
x [3] = -1.10 x [3] = -2.15130e+01
x [4] = 1.30 x [4] = -2.08757e+01
x [5] = -1.30 x [5] = -6.34043e+00
```

1.2 复系数多项式求值

【功能】

计算复系数多项式

$$P(z) = a_{n-1}z^{n-1} + a_{n-2}z^{n-2} + \cdots + a_1z + a_0$$

在指定点 z 处的函数值。

【方法说明】

同 1.1 节的方法说明，但改成复数运算。

【函数语句与形参说明】

void cpoly (ar, ai, n, x, y, u, v)

形参与函数名类型	参 数 意 义
double ar [n]	存放多项式复系数的实部。
double ai [n]	存放多项式复系数的虚部。
int n	多项式的项数。
double x	指定的复变量值的实部。
double y	指定的复变量值的虚部。
double *u	返回多项式值的实部。
double *v	返回多项式值的虚部。
void cpoly	过程。

本函数要调用复数相乘的函数 cmul (), 参看 2.1 节。

【函数程序】

```

/* ch01-02.c */
void cpoly (ar, ai, n, x, y, u, v)
double ar [], ai [], x, y, *u, *v;
int n;
{ double pu, pv;
  int i;
  pu=ar [n-1];
  pv=ai [n-1];
  for (i=n-2; i>=0; i--)
  { cmul (pu, pv, x, y, &pu, &pv);
    pu +=ar [i];
    pv +=ai [i];
  }
  *u = pu;
  *v = pv;
  return;
}

```

【例】

计算复系数多项式

$$P(z) = (2+j2)z^3 + (1+j)z^2 + (2+j)z + (2+j)$$

当 $z=1+j$ 和 $1-j$ 时的函数值。

主函数程序

```

/* ch01-02e.c */
#include <stdio.h>
#include <math.h>

```

```

#include          " ch02-01.c"
#include          " ch01-02.c"
#define          SB (a)    (a>=0.0)? '+': '-'
main ()
{ double ar [4] = {2.0, 2.0, 1.0, 2.0};
  double ai [4] = {1.0, 1.0, 1.0, 2.0};
  double x=1.0, y=1.0, u, v;
  cpoly (ar, ai, 4, x, y, &u, &v);
  printf ("%12.6e %cj%11.e\n", u, SB (v), fabs (v));
  x=1.0; y=-1.0;
  cpoly (ar, ai, 4, x, y, &u, &v);
  printf ("%12.6e %cj%11.e\n", u, SB (v), fabs (v));
}

```

运行结果

```

-7.00000e+00+j6.00000e+00
 7.00000e+00-j1.00000e+01

```

1.3 实系数多项式相乘

【功能】

求两个实系数多项式

$$P_m(x) = p_{m-1}x^{m-1} + p_{m-2}x^{m-2} + \cdots + p_1x + p_0$$

$$Q_n(x) = q_{n-1}x^{n-1} + q_{n-2}x^{n-2} + \cdots + q_1x + q_0$$

的乘积多项式

$$\begin{aligned}
 S_{m+n-1}(x) &= P_m(x) Q_n(x) \\
 &= S_{m+n-2}x^{m+n-2} + S_{m+n-3}x^{m+n-3} + \cdots + S_1x + S_0
 \end{aligned}$$

【方法说明】

乘积多项式的系数按如下公式计算：

$$s_k = 0, k = 0, 1, \cdots, m+n-2$$

$$s_{i+j} = s_{i+j} + p_i q_j, i = 0, 1, \cdots, m-1; j = 0, 1, \cdots, n-1$$

【函数语句与形参说明】

void rpoly_mul (p, m, q, n, s, l)

形参与函数名类型	参 数 意 义
double p [m]	存放 $P_m(x)$ 多项式的系数。
int m	$P_m(x)$ 多项式的项数。
double q [n]	存放 $Q_n(x)$ 多项式的系数。

续表

形参与函数名类型	参 数 意 义
int n	$Q_n(x)$ 多项式的项数。
double s [l]	返回乘积多项式的系数。
int l	乘积多项式的项数, $l=m+n-1$ 。
void rpoly_mul	过程。

【函数程序】

```

/* ch01-03.c */
void rpoly_mul (p, m, q, n, s, l)
int m, n, l;
double p [], q [], s [];
{
    int i, j, k;
    for (i=0; i<l; i++)
        s[i] = 0.0;
    for (i=0; i<m; i++)
        for (j=0; j<n; j++)
            {k=i+j;
              s[k] += p[i] * q[j];
            }
    return;
}

```

【例】

计算两个多项式

$$P_6(x) = 3x^5 - x^4 + 2x^3 + 5x^2 - 6x + 4$$

$$Q_4(x) = 2x^3 - 6x^2 + 3x + 2$$

的乘积多项式 $S_9(x) = P_6(x) Q_4(x)$ 。

主函数程序

```

/* ch01-03e.c */
#include <stdio.h>
#include "ch01-03.c"
main ()
{
    double p [6] = {4.0, -6.0, 5.0, 2.0, -1.0, 3.0};
    double q [4] = {2.0, 3.0, -6.0, 2.0};
    double s [9];
    int i;
    rpoly_mul (p, 6, q, 4, s, 9);
}

```

```

for (i=0; i<9; i++)
    printf (" s [%d] = %12.6e\n", i, s [i]);
}

```

运行结果

```

s [0] = 8.00000e+00
s [1] = 0.00000e+00
s [2] = -3.20000e+01
s [3] = 6.30000e+01
s [4] = -3.80000e+01
s [5] = 1.00000e+00
s [6] = 1.90000e+01
s [7] = -2.00000e+01
s [8] = 6.00000e+00

```

1.4 复系数多项式相乘

【功能】

求两个复系数多项式

$$P_m(z) = p_{m-1}z^{m-1} + p_{m-2}z^{m-2} + \cdots + p_1z + p_0$$

$$Q_n(z) = q_{n-1}z^{n-1} + q_{n-2}z^{n-2} + \cdots + q_1z + q_0$$

的乘积多项式

$$S_{m+n-1}(z) = P_m(z) Q_n(z) \\ = s_{m+n-2}z^{m+n-2} + s_{m+n-3}z^{m+n-3} + \cdots + s_1z + s_0$$

其中所有的系数均为复数。

【方法说明】

同 1.3 节的方法说明，但所有运算为复数运算。

【函数语句与形参说明】

```
void cpoly_mul (pr, pi, m, qr, qi, n, sr, si, l)
```

形参与函数名类型	参 数 意 义
double pr [m]	存放 $P_m(z)$ 多项式系数的实部。
double pi [m]	存放 $P_m(z)$ 多项式系数的虚部。
int m	$P_m(z)$ 多项式的项数。
double qr [n]	存放 $Q_n(z)$ 多项式系数的实部。
double qi [n]	存放 $Q_n(z)$ 多项式系数的虚部。
int n	$Q_n(z)$ 多项式的项数。

续表

形参与函数名类型	参 数 意 义
double sr [1]	返回乘积多项式系数的实部。
double si [1]	返回乘积多项式系数的虚部。
int l	乘积多项式的项数, $l=m+n-1$ 。
void cpoly _mul	过程。

本函数要调用复数相乘的函数 cmul (), 参看 2.1 节。

【函数程序】

```

/* ch01-04.c */
void cpoly _mul (pr, pi, m, qr, qi, n, sr, si, l)
double pr [], pi [], qr [], qi [], sr [], si [];
int m, n, l;
{ double u, v;
  int i, j, k;
  for (i=0; i<l; i++)
    sr [i] =si [i] =0.0;
  for (i=0; i<m; i++)
    for (j=0; j<n; j++)
      {k=i+j;
        cmul (pr [i], pi [i], qr [j], qi [j], &u, &v);
        sr [k] +=u;
        si [k] +=v;
      }
  return;
}

```

【例】

求两个复系数多项式

$$P_6(z) = (3+j2)z^5 + (-1-j)z^4 + (2+j)z^3 + (5-j4)z^2 + (-6+j3)z + (4+j2)$$

$$Q_4(z) = (2+j)z^3 + (-6-j4)z^2 + (3+j2)z + (2+j)$$

的乘积多项式 $S_9(z) = P_6(z) Q_4(z)$ 。

主函数程序

```

/* ch01-04e.c */
#include <stdio.h>
#include <math.h>
#include "ch02-01.c"

```

```

#include          " ch01-04.c"
#define          SB (a)    (a>=0.0)? '+': '-'
main ()
{ double pr [6] = {4.0, -6.0, 5.0, 2.0, -1.0, 3.0};
  double pi [6] = {2.0, 3.0, -4.0, 1.0, -1.0, 2.0};
  double qr [4] = {2.0, 3.0, -6.0, 2.0};
  double qi [4] = {1.0, 2.0, -4.0, 1.0};
  double sr [9], si [9];
  int i;
  cpoly_mul (pr, pi, 6, qr, qi, 4, sr, si, 9);
  for (i=0; i<9; i++)
    printf (" s [%d] =%12.6e %cj%11.e\n", i, sr [i], SB (si [i]), fabs (si
[i]));
}

```

运行结果

```

s [0] =6.00000e+00+j8.00000e+00
s [1] =-7.00000e+00+j1.40000e+01
s [2] =-2.60000e+01-j3.40000e+01
s [3] =8.00000e+01+j1.60000e+01
s [4] =-5.80000e+01+j8.00000e+00
s [5] =9.00000e+00-j1.50000e+01
s [6] =1.00000e+01+j2.60000e+01
s [7] =-1.10000e+01-j2.70000e+01
s [8] =4.00000e+00+j7.00000e+00

```

1.5 实系数多项式相除

【功能】

求两个实系数多项式

$$P_m(x) = p_{m-1}x^{m-1} + p_{m-2}x^{m-2} + \cdots + p_1x + p_0$$

$$Q_n(x) = q_{n-1}x^{n-1} + q_{n-2}x^{n-2} + \cdots + q_1x + q_0$$

相除后的商多项式 $S_{m-n+1}(x)$ 和余多项式 $R_{n-1}(x)$ 。

【方法说明】

设商多项式最高次为 $k=m-n$, 则 $S_{k+1}(x)$ 的系数由如下过程确定:

$$s_{k+1-i} = p_{m-i} / q_{n-1}$$

$$p_j = p_j - s_{k+1-i} q_{j+i-k}, \quad j = m-i, m-i-1, \cdots, k+1-i$$

$$i = 1, 2, \cdots, k+1$$

最后的 $p_0, p_1, \cdots, p_{n-2}$ 即为余多项式的系数 $r_0, r_1, \cdots, r_{n-2}$ 。

【函数语句与形参说明】

void rpoly_div (p, m, q, n, s, k, r, l)

形参与函数名类型	参 数 意 义
double p [m]	存放 $P_m(x)$ 多项式的系数。
int m	$P_m(x)$ 多项式的项数。
double q [n]	存放 $Q_n(x)$ 多项式的系数。
int n	$Q_n(x)$ 多项式的项数。
double s [k]	返回商多项式的系数。
int k	商多项式的项数, $k=m+n-1$ 。
double r [l]	返回余多项式的系数。
int l	余多项式的项数, $l=n-1$ 。
void rpoly_div	过程。

【函数程序】

```
/* ch01-05.c */
void rpoly_div (p, m, q, n, s, k, r, l)
int m, n, k, l;
double p [], q [], s [], r [];
{int i, j, ll, mm;
  for (i=0; i<k; i++)
    s[i] = 0.0;
  if (q[n-1] + 1.0 == 1.0)
    return;
  ll=m-2;
  for (i=k-1; i>=0; i--)
  { s[i] = p[ll+1] / q[n-1];
    mm=ll;
    for (j=0; j<n-1; j++)
      { p[mm] -= s[i] * q[n-2-j];
        mm--;
      }
    ll--;
  }
  for (i=0; i<l; i++)
    r[i] = p[i];
  return;
```