第一章 导 论

C语言科学与工程程序库是一个运行于IBM PC、XT、AT和大多数IBM兼容机的通用C函数集。这些函数解决了科学与工程应用领域中的常见的数据分析和图形问题，它们分别包含在17个不同的文件中，每个文件含若干相关的函数。其分类见下表：

分　　类	函数文件
一般统计	sumstats. C
多重回归	mulreg. C
曲线拟合	curvefit. C
数值积分与微分	integrat. C
Fourier 分析	fft. C
Lotus 文件转换	lotus. C
联立方程	gj. C
复数联立方程	cgj. C
矩阵运算	matmath. C
数据平滑	smooth. C
复数运算	complex. C
绘制图形	segraph. C
三维图形	plot3d. C
线性规划	lsimplex. C
非线性方程求解	roots. C
特殊函数	specfunc. C
异步通信	miscio. C

1. 数理统计

计算数据集合的概要统计，包括最小值、最大值、数据值范围、

方差、标准偏差、平均误差和众数。

示例程序：statsdem. C

2. 多重回归

对数据集进行最小二乘线性多重回归。函数结果并返回回归方程系数、y 的估值、残差、估值的标准误差、回归系数的标准误差、测定系数(R 平方值)和相关系数(R 值)。

示例程序：regdem. C

3. 曲线拟合

我们提供两种不同的拟合技术，将实验数据拟合成光滑曲线。第一种是多项式曲线拟合，第二种是三维样条曲线拟合。

示例程序：curvedem. C 和 splinede. C

4. 数值积分

根据实验数据计算出由实验数据曲线所包围的面积，或一个任意函数曲线所包围的面积。采用的算法是 Simpson 1/3 法则和 Simpson 3/8 法则。

示例程序：integrde. C

5. 微分方程求解

解决涉及一阶微分方程的初值问题。使用变步长的 Runge—Kutta—Fehlberg 方法。

示例程序：rkfdemo. C

6. Fourier 分析

计算数据集的快速富里叶正变换和快速富里叶反变换；使用快速富里叶技术(FFT)计算二维 Fourier 变换；并解决在调和分析中所使用的矩形、Parzen、Hanning、Welch、Hamming 和 Exact Blackman 窗口；计算波形的能谱周期图。

示例程序：fftdemo. C，fft2demo. C，fftwinde. C，fftpower. C，graphfft. C

7. Lotus 1—2—3 文件转换

*以某种格式把数值数组保存到磁盘上，这些数值数组可以使*用 Lotus 文件输入选择项被输入到一个 Lotus 1—2—3 电子表中。

表示列和行头的字母数字字符串也可被存放，使得这些字串能作为“标号”输入到 Lotus 1—2—3 中。

示例程序：lotusdem. C

8. 联立方程组的求解

用 Gauss—Jordan 法求解一个线性方程组。函数返回系统解矢量、X 变量系数矩阵的逆阵和 X 变量系数矩阵的行列式。书中给出了二种不同的函数，一种用于求解实型联立方程组，另一种用于求解复型联立方程组。

示例程序：gjdemo. C，cgjdemo. C

9. 矩阵运算

矩阵运算模块包含了计算矩阵乘积、矩阵求和、矩阵转置、矩阵行列式、逆矩阵和求实对称矩阵特征值的函数。函数可用于处理实数和复数。

示例程序：matmathd. C，cmmathde. C

10. 复数运算

计算复数的加、减、乘和除以及复数的指数、模、极角等。

示例程序：cgjdemo. C

11. 数据平滑和卷积

使用 Savitzky—Golay 函数来减少实验数据集中的噪声量。广义卷积函数还允许用户设置加权系数和正规因子。

示例程序：smoothdem. C

12. 绘制图形

通过使用本程序库中的特殊例程，可以产生直线图、条形图、散布图、等高线堆栈式直线图和栈式条形图、条形图组、Linear、Log 和半 Log 比例图。CRT 图形适配器支持包括 CGA、EGA、VGA、Hercules 单显和其它设备（阅读“编译程序手册”）。打印机支持包括 Epson MX、FX 和 LQ 在内的打印机、HP Laser Jet 系列、HP Think Jet 和 Toshiba P1351、P351。

示例程序：linedem. C，bardem. C，scatdem. C，groupdem. C，logdem. C，errordem. C，contour. C，curvedem. Cgraphfft. C，fftwinde.

C,smoothwe. C,smoothsg. C

13. 三维绘图例程

使用二维视图投影来处理三维物体的建模和视图的通用系统。物体在显示前可以在三维坐标系中进行比例变换、旋转变换、平移变换,此外,还包括平行和透视投影变换等。

示例程序:house3d. C, rotaxes. C, cross2d. fir, funcplot. c, hideline. C

14. 线性规划

求解线性规划方程的标准单纯形表系统。

示例程序:simplexd. C

15. 非线性方程求根

采用二分法、Brents 和 Newton 法求解非线性方程的根。

示例程序:rootsdem. C

16. 特殊函数

科学与工程领域中出现的许多数学方程不包括在标准数学函数库中。这些特殊函数包括 gamma、beta 和贝塞尔函数,还包括实数和虚数的错误函数、双曲三角函数及正交多项式的求解方法。

示例程序:specdemo. C

17. 异步通信

Microsoft C 对 RS－232 通信没有足够的支持。在 asyncxx 库中的函数包括基本的 RS－232 支持。程序员可设置所有由通信参数支持的 IBM DOS。

示例程序:plotterd. c

§1.1 用 C 科学与工程工具启动

在你的硬盘上创建一个新目录:MKDIR\MC006
并将主磁盘的内容拷到上面,改变工作目录到:C:\MC006

为了运行一个特殊的示例程序,用它调用的任何库文件(.obj 或.lib)编译链接主源文件。下面表示如何做:

cl gjdemo. c gj. c miscio. c

如果你正在使用 Quick C 编辑程序，你可设置程序表到正确的 *. mak 文件。在这个例子中，gjdemo. mak 是正确的 *. mak 文件。

主源文件与它直接和间隔调用的文件一起被编译并链接，产生一个可执行程序。

curvedem. c 示例程序包括下列文件：P4(A)

curvedem. c 程序调用在 curvefit 文件中的库函数 PolyClurveFit，而 PolyCurveFit 调用 mulreg 文件中的多重回归例程，多重回归例程再调用 gj 文件中的 Gauss－Jordar 例程，最后用科学与工程图形库绘制一个给定的 curvefit 结果。

curvedem. c 主程序有如下的程序头：P4(B)

在主程序中调用的所有函数需要通过 # include＜*. h＞或“*. h”文件被调用，这样对于从其他被单独编译的模块中输入的函数调用就产生了正确的代码。

文件 miscio. c 也包括在标准 Microsoft C 库中不包括的函数，这些函数包括文本窗口的屏幕控制指令如 GoToxy 和 ClrScr 函数，也包括常量 pi(π)和 pi/2 的定义。

有关在 Microsoft C 中编译链接程序的详细内容可参阅 Microsoft C 手册。

§1.2 Quick C 用户特别注意事项

如果你正使用任何图形并使用 Quick C 1. 5，则需要一个较大的堆栈。通过选择 Run Runtime 的选择可 ck 可调整堆栈。设置栈空间为 16K 左右。如果你正使用 Quick C 2. 0，则可通过选择 Options Make Compiler Large 使用大的内存模式。确认你已对每个编译程序均使用 Rebuild All，使用合适的堆栈大小/内存模式可编译全部源文件。若不能 Rebuild All 源文件则可能产生一个链接错。

在警告级上使用 Quick C 编辑程序可测试全部库函数和示例程序。如果你在大于 1 的警告级上编译和链接这个包中的任何函数，在代码中某些行上可能会看到信息数据转换。因为 Microsoft 将自动转换变量类型，例如从整型到实型，所以这个包不包括所有类型转换。

如果你在 miscio. c 库中使用 Quick C 编辑程序，你将注意到 ClrScr 和 ClrEol 函数不工作。这是因为 Quick C 编辑程序截断了由 ClrScr 和 ClrEol 函数产生的 bios 调用。

学习如何使用这些函数的最好方法是学习你的科学和工程磁盘提供的例子程序。编译这些示例程序的最简单方法是将有关的程序清单装载在 Quick C 编辑器中(Linedem. mak，bardem. mak，scatdem. mak，groupdem. mak 和 logdem. mak)然后进行编译 。

第二章　C语言的数据结构

§2.1　C语言数组

Microsoft C 语言科学与工程程序库中的所有函数均使用指向实数数组的指针作为函数的参数。与本程序库的 Pascal 和 Modula 模块不同，C 语言程序库中没有一个由所有库公用的预定义的向量和矩阵数据类型标准集，相反，所有的数组被作为数的线性开向量处理。

带一个一维数组作为参数的函数的编码形式如下：

```
FFTCalc(float * x,float * y,int n);
```

对该例程的调用形式类似于：

```
float a[100];
float b[100];
int n;
n=100;
FFTCalc(a,b,n);
```

传递数组名与传递数组的起始地址相同，不需要任何下标括号。

主调用程序可把任意大小（最大为 64K 字节）的数组传递给用这种方式编码的例程。参数 n 可由被调用的例程使用，以确定所传递的数组的大小。如果 n 的值超过了数组 x 和 y 的实际大小，则无法通知被调用的例程。与 Pascal 和 Modula 语言不同，C 不对数组进行下标检查。如果一个数组下标越界，则被调用的例程会覆盖程序中的其它部分，从而破坏数据并且可能使程序崩溃。

带一个二维数组作为参数的函数的编码形式与上例相似：

```
MatAddr(float * x,float * y,float * z,int r,int c);
```

参数 r 和 c 通知例程正在被传递的数组维数为 r×c。数组 x、

y 和 z 维数必须与这个值完全相同，否则过程会错误地计算数组下标。

如果一个二维数组名传递给了一个要求一维数组或指针的例程，则 Microsoft C 编译程序将产生警告信息。因为编译程序的这种反常现象，所有的二维数组都应通过传递二维数组的第一个元素的地址而被传递。

下例给出了设置二维数组，并把它们传递给 C 函数的错误和正确方法：

错误方法：

```
{
float x[2][2],y[2][2],z[2][2];
 ⋮
MatAdd(x,y,z,2,2);/ * 不能仅用名来传递二维数组 * /
}
```

正确方法：

```
{
float x[2][2],y[2][2],z[2][2];
 ⋮
MatAdd(&x[0][0],&y[0][0],&z[0][0],2,2);
}
```

2 维数组的另一个例子将产生一个不正确的结果：

错误方法：

```
{
float x[10][10],y[5][5],z[6][6];
 ⋮
MatAdd(&x[0][0],&y[0][0],&z[0][0],2,2);
  / * x,y,z 的维数不是 2×2 * /
}
```

这个例程不会产生预期的结果，因为三个数组的大小是不相同的，同样，被传递到函数 MatAdd(2,2)中的维数与源和目标数组

的维数不完全相同。正确调用 MatAdd 函数的的例子如下：

正确方法：

```
{
float x[2][2],y[2][2],z[2][2];
⋮
MatAdd(&x[0][0],&y[0][0],&z[0][0],2,2);
}
```

这个方法有优点，也有缺点。

优点是被调用的函数使用了尽可能少的数据空间，它仅利用了运行这个函数所需的最小内存空间，所有临时工作数组在堆上分配，以使堆栈空间最小。

缺点是要求二维数组必须与保存数据所需的空间一样大，许多程序员习惯于设置大的工作数组（如 10×10，100×100），这些大数组能处理他们可能遇到的最大数组。当分析较小数组（2×2，4×8）时，它们正好位于较大数组内部。当一个 2×2 矩阵的数据被放在一个 10×10 大小的数组中时，结果是产生了很多未被利用的内存。

程序员如何来写一个能处理各种大小的数组的程序呢？有二种方法。第一种是专为一维数组编码，一旦知道了数组的维数，就可在堆上分配保存数组所需的准确内存空间。例如，创建一个足够大的浮点数组，该数组可由用户输入维数。

```
float * x; / * x 是指向浮点数的指针 * /
int r,c;
scant("%d %d,&r,&c");
x=(float * )calloc(r * c,4);        / * 在堆中分配 r * c * 4
                                    个字节，并将此数组的起
                                    始地址赋给 x。 * /
```

如果矢量是双精度（8 字节/实型）而不是浮点型（4 字节/实型），则 calloc 函数将以如下方式被调用：

```
x=(float * )calloc(r * c,8);
```

对用二维坐标指定的任何元素，计算数组的相对偏移地址的方法非常简单。在上例中，如果数组大小是 3×5(r=3 和 c=5)，则以下面形式存取元素[2][3]：

```
x[2 * c+3]=3.2425;
```

其中 c=5。

用这种方式处理数组可以使用本程序包中的所有例程。

编一个能处理各种二维数组程序的另一种方法是使用一个大的二维工作数组，例如，50×50。通常，数组中只能被部分赋值，并且这些数据在数组的左上角。你可以写一个 C 函数，该函数能从工作数组中把感兴趣的数据(0～2 列，0～2 行)移到一个一维行向量中，工作数组中不用的行和列不拷贝进该一维向量，然后该一维向量被作为数据传递到分析例程中，一旦数据被返回，可以再将这些数据移回工作数组中。

例如，用户想转换一个 3×3 数组，使用下面的简单程序就可把值输入到工作数组中：

```
float a[100];          /* 用于向函数传递参数的线性向量 */
float b[10][10];       /* 二维工作数组 */
int i,j,r,c;
scanf("input # of rows and columns %d %d \n",r,c);
for(i=0;i<=r-1;i++)
  for(j=0;j<=c-1;j++){
    printf("Input Value for Row # %d Column # %d \n",i,
j);
    scanf("%f",b[i][j]);
}
```

在内存中，3×3 数组存贮在 10×10 工作数组中的实际情况如下：

0.0	1.0	2.0	×	×	×	×	×	×	×
3.0	4.0	5.0	×	×	×	×	×	×	×
6.0	7.0	8.0	×	×	×	×	×	×	×
×	×	×	×	×	×	×	×	×	×
×	×	×	×	×	×	×	×	×	×
×	×	×	×	×	×	×	×	×	×
×	×	×	×	×	×	×	×	×	×
×	×	×	×	×	×	×	×	×	×
×	×	×	×	×	×	×	×	×	×
×	×	×	×	×	×	×	×	×	×

下面的函数说明了如何把一个二维数组中的一部分拷贝到一个向量中：

```
void mat2vect(float * a,        /* 目标向量 */
              float * b,        /* 源二维数组 */
              int cs,           /* 源二维列的宽度 */
              int n,            /* 要拷贝的行数 */
              int m)            /* 要拷贝的列数 */
{
  int i,j;
  for(i=0;i<=n-1;i++)
    for(j=0;j<=m-1;j++)
      a[i * n+1]=b[i * cs+1];
}
```

以存贮在 10×10 工作数组中的 3×3 数组，函数 mat2vect 可使用下列参数调用：

```
mat2vect(a,&b[0][0],10,r,r)
```

其中，r=3。

当函数返回时，向量是：

0.0　1.0　2.0　3.0　4.0　5.0　6.0　7.0　8.0
9.0　×　×　×

该向量然后被传给一个矩阵求逆过程：

Matinvert(a,r)

C 科学与工程程序库中的所有例程都假设有效数据从源数组中的第一个元素开始。当把数据赋给数组时，总是从向量的第 0 个元素开始，从二维数组的元素[0][0]开始。

§2.2　C 中的复数表示

complex.c 库中包含了 complext 结构的定义，这个结构用于复数的计算。complext 的定义如下：

```
struct complext
{float x,y;
}
```

类型为 complext 的变量(名为 complexvar)可定义如下：

```
struct complext complexvar;
```

复数的赋值采用结构变量的标准 C 语法，例如：

```
complexvar.x=1.23;
complexvar.y=3.45;
```

complext 型的向量和数组的定义类似于简单数据类型的向量和数组的定义。例如：

```
struct complext cvector[10];
struct complext cmat[10][10];
```

上例定义了类型为 complext 的一个带 10 个元素的向量和一个 10×10 矩阵。

复数能象上例中的复数一样被赋给一个复数数组。

```
struct complext Complexarray[10];
Complexarray[1].x=3.1;
Complexarray[1].y=4.9;
```

接收一个二维复数数组作为参数的所有函数应编码为：

CMatAdd(struct complext * a,struct complext * b,int r,int c,
struct complext * c,);

复数数组 a,b,c 被认为是指向类型 complext 的指针且没有明确的结构。参数 r 和 c 通知 CMatAdd 例程正在被传递的复数数组是 r×c 维的。数组 a,b 和 c 必须是 r×c 的，否则过程将错误地计算正确的数组下标。在本章开头讨论的有关把二维数组转换成一维向量的方法也适用于类型 complext。下面的一个函数把存贮在一个临时工作空间中的二维数组转换到一个一维向量中：

```
void mat2vect(struct complext * a   /* 目标向量 */
        struct complext * b,        /* 源二维数组 */
        int cs,                     /* 源二维列的宽度 */
        int n,                      /* 要拷贝的行数 */
        int m)                      /* 要拷贝的列数 */
{
  int i,j;
  for(i=0;i<=n-1;i++){
    for(j=0;j<=m-1;j++){
      a[i*n+1].x=b[i*cs+j].x;
    a[i*n+j].y=b[i*cs+j].y;
  }
}
```

§2.3 使用双精度浮点值

本程序库中的所有例程都使用浮点(4 字节)实数数据类型编码。如果你想使用双精度(8 字节)实数类型，可把所有调用中的关键字 float 用关键字 double 代替，这包括特殊的 C 语言科学与工程库程序库和调用程序。你还需把所有引用中的参数改变为 calloc 内存分配函数。本程序库中的许多例程都在堆上分配暂时数据存

贮区，而 calloc 命令就是用来做这个工作的。calloc 命令的格式如下：

void * calloc(unsigned NumElements,unsigned ElementSize);

其中：NumElements 是在堆上分配的结构所包含的数据项个数。

ElementSize 是在堆上分配的数据结构中一个元素的大小。

如果把数据类型从 float 改为 double，则 NumElements 参数不变。因为 double 在内存中占用 8 个字节而不是 4 个字节，所以 ElementSize 参数需从 4 变为 8。例如：

float	double
calloc(numdat,4)→	calloc(numdat,8)

Segraph.c 库和 plot3d.c 库不应改为使用双精度浮点数，相反，你应使用如下函数把 double 型数组改为 float 型。

函数 void double2float(double * destination, float * source, int n);

和

函数 void float2double(float * destination, double * source, int n);

可在 miscio.c(miscio.h)库文件中找到。

第三章　概要统计

函数 SummaryStats

包含文件：sumstats. c

原型文件：sumstats. h

函数功能：

对数据集的每一列计算概要统计——最大值、最小值、范围、总和、平均值、方差、标准偏差、众数和平均值的标准误差。

函数原型：

```
void SummaryStats (float * dataset, int numobs, int numcol, float *
                   minima, float * maxima,
                   float * range, float * sumxx, float * mean, float
                   * variance, float * stddev,
                   float * semean, float * mode);
```

调用示例：

```
float dataset[50][5];   {或 float * dset;或 float dset[250];}
int numobs;
int numcol;
float minima[5];        {或 float * minima;}
float maxima[5];        {或 float * maxima;}
float range[5];         {或 float * range;}
float sumxx[5];         {或 float * sumxx;}
float mean[5];          {或 float * mean;}
float variance[5];      {或 float * variance;}
float stddev[5];        {或 float * stddev;}
```

folat semęan[5]; {或 float * semean;}

float mode[5]; {或 float * mode;}

numobs=50;numcol=5;

SummaryStats (&dataset[0][0], numobs, numcol, minima, maxima,
range,sumxx,mean,variance,stddev,semean,mode);

其中:

dataset:是一个二维浮点型数据,其维数必须是(numobs * numcol)。Dataset 保存被分析的原始数据,有效数据从 dataset[0][0]开始。

numobs:是 dataset 中观察值的个数。

numcol:是 dataset 的列数。

minima:维数为 numcol 的浮点型向量,它返回 dataset 中每一列数据的最小值。

maxima:维数为 numcol 的浮点型向量,它返回 dataset 中每一列数据的最大值。

range:维数为 numcol 的浮点型向量,它返回 dataset 中每一列数据的范围值(maxima-minima)。

sumxx:维数为 numcol 的浮点型向量,它返回 dataset 中每一列数据的总和。

mean:维数为 numcol 的浮点型向量,它返回 dataset 中每一列数据的平均值。

variance:维数为 numcol 的浮点型向量,它返回 dataset 中每一列数据的方差值。

stdder:维数为 numcol 的浮点型向量,它返回 dataset 中每一列数据的标准偏差值。

semean:维数为 numcol 的浮点型向量,它返回 dataset 每一列数据平均值的标准误差。

mode:维数为 numcol 的浮点型向量,它返回 dataset 中每一列数据的众数值。

特别说明:

1. 每一列的最大值、最小值，范围等都应分别在每个输出向量中返回(即第 0 列的最大值在 maxima[0]中，第 1 列的最大值在 maxima[1]中，依此类推)。

示例程序：statsdem. c

示例程序 statsdem. c 可用下面的命令与它的库文件一起被编译和链接：

cl statsdem. c sumstats. c miscio. c

编译后生成 statsdem. exe，该程序的运行结果如下：

```
summary statistics FOR storedata

        variable #0

number OF observations =      10
minima =                      101.000
maxima =                      421.000
range =                       320.000
mean =                        227.000
sum =                         2270.000
variance =                    8838.000
standard deviation =          94.011
standard error OF mean =      29.729
mode =                        236.000
summary statistics FOR storedata

        variable #1

number OF observations =      10
minima =                      31.000
maxima =                      98.000
range =                       67.000
mean =                        68.600
sum =                         686.000
variance =                    624.711
standard deviation =          24.994
standard error OF mean =      7.904
mode =                        80.000
press carriage return TO END PROGRAM.
```

sumstats. c 的源程序清单如下：