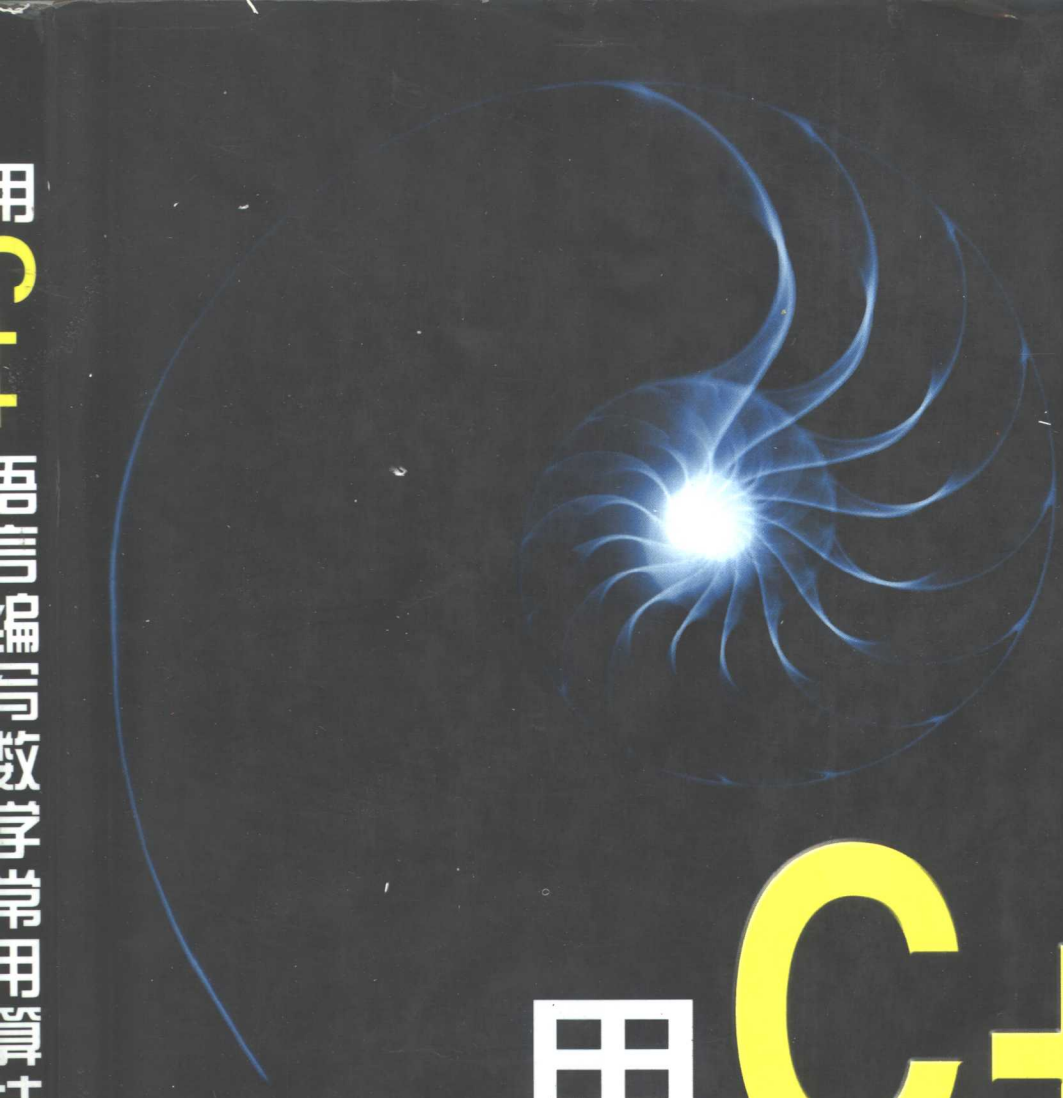

用C++语言编写数学常用算法

陈必红 著

人民邮电出版社
www.pptph.com.cn

附光盘

用C++语言编写数学常用算法
版社

用 C++ 语言编写数学常用算法

陈必红　著

人民邮电出版社

图书在版编目(CIP)数据

用C++语言编写数学常用算法/陈必红著. -北京:人民邮电出版社,2000.11
ISBN 7-115-08904-3

Ⅰ.用… Ⅱ.陈… Ⅲ.计算机辅助计算-C语言-程序设计 Ⅳ.TP391.75

中国版本图书馆CIP数据核字(2000)第76316号

用C++语言编写数学常用算法

◆ 著 陈必红
责任编辑 邹文波

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@pptph.com.cn
网址 http://www.pptph.com.cn
北京鸿佳印刷厂印刷
新华书店总店北京发行所经销

◆ 开本:787×1092 1/16
印张:16
字数:398千字 2000年12月第1版
印数:1-4 000册 2000年12月北京第1次印刷
ISBN 7-115-08904-3/TP·1911

定价:34.00元

内容提要

本书主要研究利用C++语言编写各种与实数和复数有关的数学算法，如线性代数、矩阵运算、实数方程求解、插值、拟合、数值积分、微分方程求解、特殊函数、函数变换回归分析等等。书中所有程序均调试通过。本书提供的类库为作者的独创，具有编程容易、效率高的特点。

本书可供科研人员、工程技术人员和程序员阅读，也可作为高等院校学生学习、研究和软件开发的参考书。

前　言

从事科技研究的人员经常需要编写一些数学常用算法的程序，如矩阵求逆、解线性方程组、求解微分方程、作线性回归等。当然可以用 MATLAB 这样的软件工具来进行各种数学演算，但这样就无法编写自己的软件产品和自己的程序。比如说一个数控机床的控制程序，或者一个自动驾驶仪程序，就需要拿出自己的软件产品。C++语言是一种好的编程语言，可是搞科研的人往往来不及进行大量的C++语言的研究，常常在初通 C 语言的语法之后就开始编程。

当他们在编程需要解一个数学问题时，通常要参考一些 C 语言实现数学算法的参考书。而现有的一些用 C 语言或者 C++语言实现算法的参考书很多没有考虑到应用上的困难。比如说有一个 C 语言的求矩阵的逆的范例，如果矩阵特别大，内存放不下，此时需要使用磁盘作缓存数组；可这样一来他们就不能直接使用范例程序，而不得不将范例程序进行更动以适应具体的需要；改完之后再进行调试，程序的可靠性就大受影响。

一些C++语言的书过分地大谈深奥的类编程知识，并做出使用起来极为复杂的类库，这样的书又导致阅读者不得不先掌握很深的C++的面向对象编程的知识，在一些继承性等概念上搞得昏头涨脑，而且这些类库也过分地依赖于 BORLAND 公司提供的容器类库。虽然 BORLAND 公司提供了此类库的全部源程序，但这些程序的可读性很差，因此更改也很不容易。

本书是专门针对一些不想学习很多C++知识，却又能够很快地编写出数学常用算法程序的技术人员编写的，为他们提供一个可用的数学软件工具，以便能够节省他们编写数学计算程序的时间。阅读本书甚至不需要特别懂C++面向对象的编程知识，只需初通C++语言就行。当然，这是本书第 1 章的任务。在大多数情况下是无需看后面几章的，除非有更高的学习C++语言的愿望，或者想对本书提供的类库有更深的了解，或者想对本书的程序进行更动或者改进。本书后面几章则给出了所提供的数学计算类的全部技术说明。本书附带的光盘也提供了全部的源代码，并以中文加以详细地注释。

在本书的编写过程中，受到了来自各方朋友的支持和帮助，在此表示深切的谢意。

陈必红　博士
深圳大学数学系

常用算法索引

目　　录

引　言

下面介绍本书所附光盘中程序的安装方法和本书的结构以及如何使用本书和所附光盘中的程序。

光盘安装

本书的C++源代码都在光盘的 math 子目录下，并且都是后缀为 cpp 的C++源程序文件和后缀为 h 的头文件。因此只需将光盘中的 math 子目录下的文件都拷贝到要工作的、存放源程序的目录就可以了。

文件的清单如下：

buffer.cpp, cbuffer.cpp, matrix.cpp, cmatrix.cpp, func.cpp, funcl.cpp, cfunc.cpp, vfunc.cpp, specfunc.cpp

buffer.h, cbuffer.h, matrix.h, cmatrix.h, func.h, cfunc.h, vfunc.h

在编写自己的程序时，你必须有C++编译器(Borland C++ 或 V C++，或者其他C++编译器)。但有一条，本书中的程序用了 Borland C++ 的复数类，也就是使用了 complex.h 这个头文件。如果编译系统没有这种复数类的支持，那么只好自己编写一个复数类的程序了，否则代码没有办法运行。

把这些源程序并到程序产品中去有多种办法。最简单的办法就是把这些 cpp 程序编译后同源程序连接成一个执行程序，或者也可以把程序编译后形成一个库文件，和应用程序相连接。如果是编写 Window 或者 Win95、Win98、Win2000 程序，也可以选择将它们都编译成动态链接库 dll 文件。

本书不打算讲述把几个C++程序连结到一起的方法，因为不同的编译系统情况也不同，请参考相应的参考书。阅读本书需要懂一些C++语言编程技术，至少需要会把几个C++语言编写的源程序文件连到一起编译成可执行文件。

本书的结构和使用方法

本书内容采用由浅入深的方式讲述。

第 1 章介绍使用本书提供的程序进行各种基本的数学运算的编程方法，如矩阵求逆、解线性方程组、求积分、求卡尔曼滤波、求

傅里叶变换等各种算法，并给出样例程序。此外还对各种算法的原理进行了初略的介绍。

在正常情况下，读者只需阅读本书的第1章就足够用了。这里说的正常情况，是指程序中的一些数据不是怪怪的样子。比如说如果要求逆，就必须保证这矩阵一定有逆；如果求对称正定阵的所有特征向量和特征值，就必须保证提供的确实是对称正定阵。还有就是矩阵并不是大得吓人，有个几阶十几阶或者几十阶都没有关系，甚至几百阶的矩阵也能凑合着对付。

如果要使用得更好一些，比如说想对特别大的矩阵进行操作，或者想进行出错处理，就是说当接收到的数据不合要求时，本书的程序不会导致整个运行程序结束，而是发出出错信息，并让程序继续下去，或者想知道一个数组最多能够容许多少个实数或者复数，或者希望知道在几种可选的办法中哪一种效率比较高，此时就需要阅读第2章。第2章将透露编写的类的更多信息，它有助于读者掌握这些类的性能指标，并对一些参数进行控制。对于类变量的情况，比如说占多少内存，数据存放在哪，怎样编程效率比较高等。

学习并掌握了第2章的内容，基本上就能够利用本书的程序进一步编写非常合格的软件产品，但如果若想对本书的类库进行比较深入的了解，并进行扩充，增加一些功能或作一些简单的修改，比如说嫌计算精度不够，想用长双精度实数来作所有的数学运算，或者将一些新的数学算法扩充到本书的类库当中，那么就需要阅读第3章。第3章详细介绍了设计数学的两大类重要类库(矩阵类和函数类)的基本思路。

第4章将各个类库的作用等技术细节，每个类的定义，各个数据成员的定义，每个成员函数和全局函数的定义，并将它们的基本结构全部列出。在提供的源程序中包含有详细的中文注释，结合着这些内容可以说掌握了本书的数学计算类库的全部知识。

作者

第 1 章

基本用法

1.1 矩阵类基本用法

在数学运算中经常使用的是矩阵和向量，而向量其实也是多行一列的矩阵。因此，本书定义了两个矩阵类，实数矩阵类 matrix 和复数矩阵类 cmatrix 来完成有关的数学运算。利用矩阵类可以很方便地做矩阵的加、减、乘、求逆、求行列式、解线性方程组、求特征值和特征向量等运算。本节介绍实矩阵类的用法。实矩阵类在文件 matrix.h 中定义，所有算法的实现程序都在源文件 matrix.cpp 中。复数矩阵类在文件 cmatrix.h 中定义，其算法的实现在源文件 cmatrix.cpp 中。它们都受到缓存器类的支持，因此实矩阵类的使用需要包括描述实缓存器的文件 buffer.h 和 buffer.cpp，而复矩阵类的使用则更需要包括描述复缓存器的文件 cbuffer.h 和 cbuffer.cpp。

本节仅介绍实矩阵类 matrix 的用法，matrix 类的设计构思参见第 3 章 3.3 节“矩阵类的基本结构”，其详细的技术说明，包括C++语句的定义代码参见第 4 章第 4.3 节。

1.1.1 一个例子

先通过一个例子来说明实矩阵类 matrix 的用法。

假设有三个矩阵 A 和 B 和 C，$A=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$，$B=\begin{pmatrix}2 & 2\\ 4 & 6\end{pmatrix}$，$C=\begin{pmatrix}3 & 3\\ 3 & 3\end{pmatrix}$，我们要计算 $2A^{-1}B+C$，并将结果输出到屏幕上。下面的程序可以完成这个功能：

```
1: # include <iostream.h>
2: # include "matrix.h"
3:void main()
4:{
5:static double aa[2][2] = {{1.0,2.0},{3.0,4.0}};
6:static double bb[2][2] = {{2.0,2.0},{4.0,6.0}};
7:matrix a(aa,2,2),b(bb,2,2);
8:cout≪2.0 * ~a * b+3.0;//此程序的核心是这一句，它实现运算并输出到屏幕
```

9:}

程序的第1行包含了一个C++的流处理文件 iostream.h。

第2行包含了文件 matrix.h,凡是要用到本章中提到的类的功能和函数在程序的开头必须包含它。

第3行为主程序的入口函数 main,所有C++语言都从这个函数开始运行,被第4行和第9行的大括号{}括起来的部分就是函数体。

第5行和第6行说明了两个实数数组 aa 和 bb,它存放矩阵 A 和 B 的内容。

第7行则将这两个数组 aa 和 bb 包装到设计的 matrix 类型的类库变量 a 和 b 中,指明这两个矩阵变量 a 和 b 的维数都是2行2列。

核心的计算在第8行,就是语句 cout≪2.0 * ~a * b+3.0。这一句完成了矩阵的计算并将结果打印到屏幕。在C++语言中,cout 代表了一个标准的输出设备,或称为输出流。如果有什么变量的内容想要输出到屏幕,就将这个内容用双小于号≪送到 cout 就成。不管是一段字符,还是一个整数或者实数变量,或者是像这样的矩阵变量都可以这样输出。为了使用这个流的功能,所以第1行要包含 iostream.h 文件。

表达式 2.0 * ~a * b+3.0 完成所要求的计算 $2A^{-1}B+C$。因为矩阵 C 是一个常数为3的矩阵,所以表达式后面只要简单地加上3就可以了,这种"数加"的算法是参照矩阵的数乘而(自作主张)定义的。~a 利用运算符 ~ 对 a 进行求逆,乘法使用 * 这一点同其他变量类型的乘法使用在形式上一致。

程序执行完毕后屏幕上显示:

matrix 2 2

1:3 7

2:5 3

这是一个矩阵变量输出的"标准"格式,这"标准"当然是作者自己想当然地定的。其中第一行以关键字 matrix 开头,后面紧跟着行数和列数,后面紧跟着一行接一行地显示数据,每一行以行号加冒号开头,后面跟着用空白字符隔开的这行各列的数据。算出正确的结果是 $\begin{pmatrix}3 & 7\\5 & 3\end{pmatrix}$。

其实,上面程序的重点是第7行的对矩阵类变量怎样做初始化以及第8行的表达式来说明矩阵怎样计算。而 cout 作为C++流在编写实用的软件时基本上是不用的,因为现在要么倾向于用很好看的界面来显示数据,要么就是一个自动控制系统,算出的数据又去控制其他接口。而第7行的初始化只是其中的不常用的一种。因为,实际的数据也经常不是存放在数据文件中,就是存放在数据库中,或者要通过一个输入口来转换,很少直接写成内存数组的形式(后面会介绍其他初始化方法)。

这个例子为了简单只用了2行2列的数组,就是更大的数组,如几千个元素的数组,使用方法也一样。想像一下,如果不用本书的矩阵类,而是自己完成矩阵求逆到矩阵相乘的各种算法,包括申请内存中转,各种的循环套循环,将是多么麻烦的事!而现在有了的矩阵类,编写的效率大大提高,程序的质量也相当可靠。

1.1.2 矩阵类变量的初始化

在进行矩阵运算之前,首先要将各种数据包装到矩阵变量中去。总共有三类初始化矩阵

的办法：一是利用内存数组或者指针，二是利用数据文件，三是分别对每个元素进行赋值。

利用内存数组或指针初始化矩阵变量

matrix 是作者定义的一个矩阵类，使用起来就像是一般的 C 语言数据类型，如 int，double 那样。比如说，语句：

```
matrix a;
```

就初始化了一个叫做 a 的矩阵变量，但此矩阵变量在被赋值之前是个空矩阵。C++ 语言可以对每一个类编写各种构造函数以适应各种不同的初始化需要，也就是在上面说明的矩阵变量右边加一对圆括号，并在括号中间按照要求填入相应的数据。下面的程序

```
static double aa[2][2] = {{1.0,2.0},{3.0,4.0}};
static double bb[2][2] = {{2.0,2.0},{4.0,6.0}};
matrix a(aa,2,2),b(bb,2,2);
```

就初始化了两个矩阵变量 a 和 b。这种构造函数需要三个变量：第一个是实数数组变量或者实数指针，后跟两个数代表矩阵的行数和列数。注意行数和列数的乘积不可超过相应的实数数组的元素总数。其实，上面的数组变量即使是一维的数组也是可以的，将上面的程序改成：

```
static double aa[4] = {1.0,2.0,3.0,4.0};
static double bb[4] = {2.0,2.0,4.0,6.0};
matrix a(aa,2,2),b(bb,2,2);
```

也一样可工作。一维数组的顺序是先排完第一行的各列，然后再排第二行的各列数字，以此类推。

此外，还可用一个指向实数的指针，申请一段内存，装入数据之后，再包装到 matrix 类的变量中。

利用数据文件进行矩阵变量的输入输出

这里定义的矩阵数据文件格式是一种文本文件，可以用文本编辑器进行编写或者由其他程序产生。文件的开始以关键字 matrix 打头，注意必须用小写，后面是空白字符，就是空格或者制表符回车换行符什么的，再后面是矩阵的行数，再接空白字符，再接矩阵的列数，再接空白字符后，就是各行数据。每一行以行号开头后面紧跟冒号“:”，再后面就是用空白字符隔开的这一行的数据。假设数据文件 test.dat 有如下的内容：

```
matrix 2 3
1:1.2 3.3 -33
2:0.2 -5 2.5
```

则语句

```
matrix a("test.dat");
```

将矩阵变量 a 初始化成一内容为$\begin{pmatrix}1.2 & 3.3 & -33\\0.2 & -5 & 2.5\end{pmatrix}$的二行三列矩阵。

用C++ 流也可以做到这一点，但首先要在程序源文件的开始加上一行：

```
# include < fstream.h >
```

下面的一段程序打开和文件 test.dat 连接的流，并建立一个空的矩阵变量 a，然后通过流将数据读入 a：

```
ifstream if("test.dat");
matrix a;
if≫a;
```

这样的语句比较啰嗦,但如果在经常要使用一个矩阵变量来装载多个矩阵数据时,可能有用处。

矩阵变量中的内容也可以通过流送到相应的文件中去。下面的程序行将矩阵变量 a 中的内容送到 test1.dat 的文件中去:

```
ofstream of("test1.dat");
matrix a;
...(中间的运算使 a 中存放着某个运算结果)
of≪a;
```

这送出的数据可能是一个运算的结果,而且可以通过其他程序或者另外的文件流将数据再读回来。

其实,其他语言,如 VB 或者 PASCAL,也可以通过这种数据文件格式与本书的C ++ 程序进行数据的双向传送。

直接对矩阵变量中的元素进行存取

语句

```
matrix a;
```

初始化一个空的矩阵,而语句

```
matrix a(2,3);
```

则初化化了一个 2 行 3 列的矩阵,而语句

```
matrix a(5);
```

则初始化了一个 5 行 1 列的矩阵,也就是一个列向量。

如果一个矩阵变量 a 原来是空的或者是 2 行 3 列的,怎样把它变成 4 行 5 列的呢? 可用下面的语句做到这一点:

```
a = matrix(4,5);
```

a 原来如果有内容的话,则原来的内容被冲掉了。

下面的语句将矩阵变量 a 的第 2 行第 3 列的值设为 4.7:

```
a.set(2,3,4.7);
```

其中 a.set 调用了矩阵类型的变量的一个类函数,所以格式就是变量名加一点后面加函数名,而调用方法则和一般的 C 语言没有什么不同。set 函数的头两个变量代表行号和列号,请注意在程序内部,行号和列号都是从 0 开始起算的。

为了方便一维列向量的值的设置,在取第某行 0 列的元素时,列号可以省略,比如下面的语句设置 a 的第 3 行 0 列的数值为 4:

```
a.set(3,4.0);
```

下面的程序段将矩阵 a 的所有元素都设成 5:

```
for(size_t i=0;i<a.rownum;i ++)
for(size_t j=0;j<a.colnum;j ++)
a.set(i,j,5.0);
```

其中 rownum 和 colnum 是矩阵类的变量，存放变量的行数和列数。

当然上面只是示范怎样设置一个矩阵的所有元素。如果真的要将矩阵的每个元素都设成一个常数 5，那么应当使用下面的语句：

```
a=5.0;
```

这是因为程序重载了等号 = 或者叫赋值运算符的缘故。这种重载技术是C++ 语言的重要技术。

也可以通过等号将一个矩阵变量的内容赋给另一个矩阵变量，如下面的语句：

```
matrix a,b;
...
a=b;
```

将 b 的内容都传送到 a 中。而 a 原来的内容如果有也全都释放了。

后面可以看到，正是大量地重载了C++ 语言的运算符，才使数学演算变得简单。

下面的程序行将矩阵变量 a 的第 2 行第 3 列的元素同 2 相乘后赋给一实数变量 x 中：

```
x=a(2,3)*2;
```

这里有趣的是矩阵变量似乎又像一个函数一样被调用，这是因为重载了函数运算符的缘故。笔者认为这种作法比重载两个取数组元素操作符 a[2][3]的方案要好。为了方便列向量的操作，后面的列号如果不写，就缺省为 0。例如，下面的语句将 a 的第 5 行第 0 列的元素取到实数变量 x 中：

```
x=a(5);
```

1.1.3　矩阵常用算法

本节介绍如何使用矩阵类完成各种常用的数学算法。在本节中，许多函数都会返回一个矩阵类型的引用。那么，引用是什么意思呢？其实引用的实质是指向一个变量的指针，但为了方便程序员编程，使你用起来就和使用一个变量而非指针一样容易。

赋值运算

矩阵类变量因为重载了赋值运算符而使矩阵的复制变得简单，假如有矩阵变量 a 和 b 则语句

```
a=b;
```

将 b 的内容赋给 a；

当然，在实际应用中更经常的是将一个能够返回矩阵或矩阵的引用的表达式赋给一个矩阵。如语句

```
a=b+c;
```

就将矩阵 b 和 c 相加后赋给矩阵变量 a。

还可以临时构造一个新的矩阵变量赋给矩阵变量 a。例如，下面的语句

```
a=matrix(30,20);
```

不管 a 原来的内容如何，将 a 重新设为 30 行 20 列的矩阵。而语句

```
b=matrix(10);
```

则将 b 重新设为 10 行一列的矩阵，或者说 10 阶的列向量。