

目 录

第一章 MATLAB 系统与语言简介	(1)
1.1 MATLAB 系统	(1)
1.1.1 什么是 MATLAB	(1)
1.1.2 MATLAB 系统的常用概念	(2)
1.1.3 MATLAB 文件类型	(5)
1.2 MATLAB 语言语法要素	(6)
1.2.1 MATLAB 的矩阵、变量与表达式	(6)
1.2.2 MATLAB 的基本管理命令	(12)
1.2.3 MATLAB 的基本运算符	(13)
1.2.4 MATLAB 的常用数学函数	(17)
1.3 简单程序设计	(18)
1.3.1 控制语句	(18)
1.3.2 M 文件、MATLAB 函数与函数型函数	(22)
1.3.3 全程变量	(26)
1.3.4 程序设计中应注意的几个问题	(27)
1.4 矩阵运算与数组运算	(28)
1.4.1 矩阵的创建函数	(28)
1.4.2 矩阵的角标	(31)
1.4.3 矩阵与数组运算	(34)
1.4.4 线性代数与稀疏矩阵	(36)
1.5 信号处理	(47)
1.5.1 信号处理函数	(47)
1.5.2 数据滤波	(47)
1.5.3 快速 FOURIER(FFT) 算法	(48)
第二章 图形功能	(49)
2.1 平面图形与坐标系	(49)
2.1.1 图形窗口与坐标系	(49)
2.1.2 基本绘图函数	(52)
2.1.3 线型、顶点标记和颜色	(55)
2.1.4 其他 2 维绘图函数	(57)
2.2 3 维图形	(59)
2.2.1 3 维图形函数简介	(59)
2.2.2 3 维线型图形	(60)

2.2.3 3维曲面	(61)
2.2.4 等高线图形	(65)
2.2.5 3维坐标系及图形元的控制	(66)
2.3 MATLAB 的色谱与着色原理	(69)
2.3.1 色谱	(69)
2.3.2 着色原理	(71)
2.3.3 色谱矩阵的分析	(72)
2.4 图像处理	(74)
2.4.1 伪色图像	(74)
2.4.2 图像显示技术	(75)
2.4.3 动画	(79)
2.4.4 图形像素位置动态输入	(79)
第三章 图形对象控制	(82)
3.1 MATLAB 图形对象简介	(82)
3.1.1 图形对象类型与结构	(82)
3.1.2 图形对象句柄及其访问	(83)
3.1.3 图形对象属性	(84)
3.2 图形窗口对象	(93)
3.2.1 图形窗口对象创建函数	(93)
3.2.2 图形窗口对象的属性	(94)
3.2.3 属性应用技巧	(101)
3.3 坐标系对象	(105)
3.3.1 坐标系对象生成函数	(105)
3.3.2 坐标系对象属性	(106)
3.3.3 属性应用技巧	(114)
3.4 线段对象	(117)
3.4.1 线段对象创建函数	(118)
3.4.2 线段对象属性	(118)
3.4.3 属性应用技巧	(120)
3.5 曲面对象	(122)
3.5.1 曲面对象创建函数	(122)
3.5.2 曲面对象属性	(123)
3.6 区域片对象	(127)
3.6.1 区域片对象创建函数	(127)
3.6.2 区域片对象属性	(128)
3.7 图像对象	(131)
3.7.1 图像对象创建函数	(131)

3.7.2 图像对象属性	(132)
3.8 文字对象	(133)
3.8.1 文字对象创建函数	(133)
3.8.2 文字对象属性	(134)
3.9 光源对象	(137)
3.9.1 光源对象创建函数	(137)
3.9.2 光源对象属性	(137)
3.10 缺省属性及其设置	(138)
3.10.1 缺省属性值	(138)
3.10.2 设置缺省属性值	(138)
3.10.3 例子	(140)
 第四章 MATLAB 的接口	(142)
4.1 MATLAB 的数据接口	(142)
4.1.1 数据结构	(142)
4.1.2 MATLAB 数据输入	(143)
4.1.3 MATLAB 数据输出	(144)
4.1.4 MAT 数据格式	(145)
4.2 文件 I/O 操作	(147)
4.2.1 文件的打开与关闭	(147)
4.2.2 二进制数据文件的读/写操作	(148)
4.2.3 文件内的位置控制	(150)
4.2.4 格式文件输入和输出	(150)
4.3 MEX 动态连接函数接口	(152)
4.3.1 MEX 文件的使用	(153)
4.3.2 C 语言 MEX 文件	(153)
4.3.3 FORTRAN 语言 MEX 文件	(167)
4.4 M 文件 Debugger	(181)
4.4.1 Debugger 主要功能	(181)
4.4.2 Debug 主要命令	(182)
4.4.3 Debugger 的使用	(182)
4.4.4 例子	(183)
 第五章 MATLAB GUI 程序设计	(189)
5.1 控制元对象及属性	(189)
5.1.1 控制元对象类型	(189)
5.1.2 控制元创建函数	(193)
5.1.3 控制元对象的属性	(193)

5.1.4 例子	(198)
5.2 菜单对象	(202)
5.2.1 菜单对象创建函数	(203)
5.2.2 菜单对象属性	(205)
5.3 应用例子	(207)
5.3.1 按钮的设计	(208)
5.3.2 收音机按钮的设计	(210)
5.3.3 滑标条的设计	(211)
5.3.4 弹出式菜单的设计	(211)
5.3.5 编辑框的设计	(212)
5.3.6 菜单的设计	(212)
5.4 MATLAB GUI 高级特性	(213)
5.4.1 择一选择的收音机按钮组的设计	(214)
5.4.2 GUI 设计方法	(215)
5.4.3 鼠标操作处理技术	(217)
5.5 中断 callback 操作	(225)
5.5.1 事件及事件队列	(226)
5.5.2 MATLAB 处理 callback 的过程	(226)
5.5.3 事件的处理	(228)
5.6 GUI 工具集 Guide	(228)
5.6.1 Guide 控制板	(228)
5.6.2 属性编辑器	(230)
5.6.3 Callback 编辑器	(231)
5.6.4 菜单编辑器	(233)
5.6.5 位置调整器	(235)
第六章 小波(Wavelet)分析工具包	(236)
6.1 主程序	(236)
6.2 小波变换计算函数	(247)
6.3 Daubechies 小波函数的生成函数	(256)
6.4 辅助函数	(259)

第一章 MATLAB 系统 与语言简介

本章主要介绍 MATLAB 的一些基本知识和概念,使读者对 MATLAB 系统有一个整体的认识。内容包括: MATLAB 系统要素, MATLAB 语言的变量与语句, MATLAB 的矩阵与矩阵元素, 数值输入与输出格式, MATLAB 系统工作空间信息, 以及 MATLAB 的在线帮助功能等。

1.1 MATLAB 系统

1.1.1 什么是 MATLAB

MATLAB 是由美国 MathWorks 公司推出的用于数值计算和图形处理的科学计算系统环境。MATLAB 是英文 MATrix LABoratory(矩阵实验室)的缩写。它的第 1 版(DOS 版本 1.0)发行于 1984 年, 经过 10 余年的不断改进, 现今已推出它的 Windows 95 版本(5.0 版)。新的版本集中了日常数学处理中的各种功能, 包括高效的数值计算、矩阵运算、信号处理和图形生成等功能。在 MATLAB 环境下, 用户可以集成地进行程序设计、数值计算、图形绘制、输入输出、文件管理等各项操作。

MATLAB 提供了一个人机交互的数学系统环境, 该系统的基本数据结构是矩阵, 在生成矩阵对象时, 不要求作明确的维数说明。与利用 C 语言或 FORTRAN 语言作数值计算的程序设计相比, 利用 MATLAB 可以节省大量的编程时间。在美国的一些大学里, MATLAB 正在成为对数值线性代数以及其他一些高等应用数学课程进行辅助教学的有益工具。在工程技术界, MATLAB 也被用来解决一些实际课题和数学模型问题。典型的应用包括数值计算、算法预设计与验证, 以及一些特殊的矩阵计算应用, 如自动控制理论、统计、数字信号处理(时间序列分析)等。

MATLAB 系统最初是由 Cleve Moler 用 FORTRAN 语言设计的, 有关矩阵的算法来自 LINPACK 和 EISPACK 课题的研究成果; 现在的 MATLAB 程序是 MathWorks 公司用 C 语言开发的, 第一版由 Steve Bangert 主持开发编译解释程序, Steve Kleiman 完成图形功能的设计, John Little 和 Cleve Moler 主持开发了各类数学分析的子模块, 撰写用户指南和大部分的 M 文件。自从第 1 版发行以来, 已有众多的科技工作者加入到 MATLAB 的开发队伍中, 并为形成今天的 MATLAB 系统做出了巨大的贡献。

MATLAB 系统由五个主要部分组成,下面分别加以介绍。

(1) MATLAB 语言体系

MATLAB 是高层次的矩阵/数组语言,具有条件控制、函数调用、数据结构、输入输出、面向对象等程序语言特性。利用它既可以进行小规模编程,完成算法设计和算法实验的基本任务,也可以进行大规模编程,开发复杂的应用程序。

(2) MATLAB 工作环境

这是对 MATLAB 提供给用户使用的管理功能的总称,包括管理工作空间中的变量,数据输入输出的方式和方法,以及开发、调试、管理 M 文件的各种工具。

(3) 图形句柄系统

这是 MATLAB 图形系统的基础,包括完成 2D 和 3D 数据图示、图像处理、动画生成、图形显示等功能的高层 MATLAB 命令,也包括用户对图形图像等对象进行特性控制的低层 MATLAB 命令,以及开发 GUI 应用程序的各种工具。

(4) MATLAB 数学函数库

这是对 MATLAB 使用的各种数学算法的总称,包括各种初等函数的算法,也包括矩阵运算、矩阵分析等高层次数学算法。

(5) MATLAB 应用程序接口 (API)

这是 MATLAB 为用户提供的一个函数库,使得用户能够在 MATLAB 环境中使用 C 程序或 FORTRAN 程序,包括从 MATLAB 中调用子程序(动态链接),读写 MAT 文件的功能。

综上所述,可以看出 MATLAB 是一个功能十分强大的系统,是集数值计算、图形管理、程序开发为一体的环境。除此之外,MATLAB 还具有很强的功能扩展能力,与它的主系统一起,可以配备各种各样的工具箱,以完成一些特定的任务。目前,MathWorks 公司推出了 18 种工具箱。用户可以根据自己的工作任务,开发自己的工具箱。

1.1.2 MATLAB 系统的常用概念

1. 命令窗口

在 Windows 95 下启动 MATLAB 系统后,Windows 95 的工作平台上会弹出一个窗口,如图 1-1 所示,这个窗口称为 MATLAB 的命令窗口(Command Window)。MATLAB 的

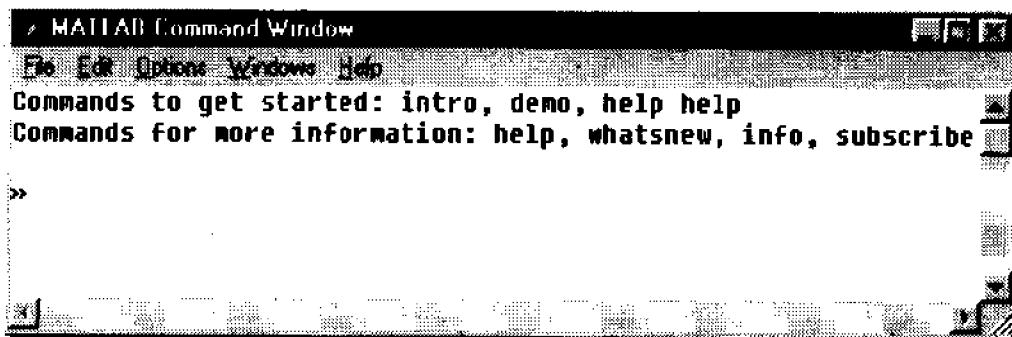


图 1-1 MATLAB 的命令窗口

命令窗口是用户与 MATLAB 解释器进行通信的工作环境,提示符“>”表示 MATLAB 解释器正等待用户输入命令。所有的 MATLAB 命令、MATLAB 函数,以及 MATLAB 程序都要在这个窗口下运行。

在命令窗口中,用户可以发出 MATLAB 命令。例如,为了生成一个 3×3 阶的矩阵,可在提示符下,键入如下的命令:

```
A=[1 2 3; 4 5 6; 7 8 9]
```

方括号命令表示矩阵,空格或逗号将每行的元素分开,而分号将矩阵的各行数值分开。再键入 Enter(回车)后,MATLAB 将回显如下的矩阵:

```
A =  
1 2 3  
4 5 6  
7 8 9
```

为了求该矩阵的逆矩阵,则只要键入命令

```
>>B= inv(A);
```

MATLAB 就将计算出相应的结果。如果不希望在命令窗口中显示计算结果,只要如上所示,在该命令后多键入一个分号即可。此时,MATLAB 系统只完成该命令所要求的计算任务,其计算结果不回显。这项功能在程序设计中是非常必要的。

MATLAB 系统也可以说是一种新的语言,该语言十分容易掌握,其结构非常类似于数学式子的书写格式,用户花上几个小时的时间即可掌握 MATLAB 语言的大部分命令。

MATLAB 系统提供了交互式的解释程序环境。为了简化命令的输入,MATLAB 提供了几种行命令编辑功能,可以使用方向键修改输入错了的命令,也可以复制先前使用过的命令。例如,假设将函数名 sqrt 错写为 srt,而键入了如下的命令:

```
>>log(srt(atan2(3,4)))
```

MATLAB 将回应一条错误信息:

```
Undefined function or variable srt.
```

为了重新输入这条命令,只要先用↑键将刚才错误的命令复制下来,然后用←键或鼠标将光标移到 s 与 r 之间,键入 q,再回车即可。在回车时,光标可以在该命令行的任何位置,没有必要将光标移到该命令的末尾。在 MATLAB 的环境下,最后执行的几条命令都存储在内存 buffer 中,所以可以复制使用过的命令。特别是可以只键入少量的几个字母,使用↑键即可复制最后一条以这些字母开始的命令。例如,键入 plot,当键入↑后,即可得到最后一条以 plot 开始的命令,大多数情况是 plot 命令。

2. 图形窗口

MATLAB 系统的强大功能之一是其优秀的图形功能。对于任何作图命令,MATLAB 将打开另一个窗口来绘制与输出图形,这样的窗口在 MATLAB 系统中被称为图形窗口 (Figure Window)。

在 MATLAB 环境中调用任何绘图函数绘图时,MATLAB 将自动生成一个图形输出窗口,并在其中绘出图形。在缺省情况下,图形窗口的标题栏标题为“Figure No.: 号码”,其中“号码”为图形窗口的序号,也称为图形窗口的句柄值,参见本书第二章。在标题栏下面是图

形窗口的主菜单栏,通常情况下,MATLAB 图形窗口的主菜单有 File、Edit、Windows 和 Help。用户可以在菜单条上加入自己的主菜单,具体方法参见本书第五章。

在同一个图形窗口中,可以绘制多个图形,也可以生成多个图形窗口,并选择其中一个图形窗口,在其中绘制图形。生成图形窗口的方法比较多,在没有图形窗口存在时,每个绘图函数都能自动生成一个图形窗口;也可以用 figure 命令生成一个新的图形窗口;还可以用命令窗口 File 菜单的 New 子菜单的 Figure 项来打开一个新的图形窗口。有关图形的绘制和管理,参见本书第二章。

3. 搜索路径

MATLAB 管理着一条搜索路径,它在搜索路径下寻找与命令相关的函数文件。例如,如果在 MATLAB 提示符下输入 example, MATLAB 解释器将按照下面的步骤来处理这条字符串:

- (a) 检查 example 是不是一个变量;
- (b) 如果不是,检查 example 是不是一个内部函数;
- (c) 如果不是,检查在当前文件夹下是否存在名为 example.mex, example.dll, 或 example.m 的文件。MEX 文件是 MATLAB 的执行文件,将优先执行;
- (d) 如果不存在,检查在 MATLAB 的搜索路径的目录下是否存在名为 example.mex, example.dll, 或 example.m 的文件。MEX 文件优先执行。

使用 MATLAB 的 path 函数,可以查看 MATLAB 系统的当前搜索路径,例如:

```
>>path  
MATLAB PATH  
C:\MATLAB\TOOLBOX\MATLAB  
C:\MATLAB\TOOLBOX\DEMO  
C:\MATLAB\TOOLBOX\SIGNAL  
C:\MATLAB\TOOLBOX\CONTROL
```

用户可以用 path 命令在 MATLAB 的搜索路径中添加新的搜索路径。例如,下面的命令

```
>>path('C:\MYFILES',path);
```

将搜索顺序改为在搜索完当前目录之后,先搜索目录 C:\MYFILES,再在当前的 MATLAB 搜索路径的目录中搜索。这是由于不带参数的命令 path 会返回至 MATLAB 的当前搜索路径中。

初始的 MATLAB 搜索路径是在文件 matlabrc.m 中描述的,该文件在 MATLAB 目录下,是在安装 MATLAB 系统和 MATLAB 的各项工具箱的时候生成的。用户可以根据自己的需要在 matlabrc.m 文件中添加或删除部分目录。

4. 外部系统命令

在 MATLAB 环境中,可以发出 Windows 或 DOS 系统命令。感叹号字符“!”命令能起到这种作用,它使得 MATLAB 将感叹号后面的命令传到相应的操作系统,这个过程通常称为使用外部系统命令。MATLAB for Windows95 系统具有 4 种形式的外部系统命令,可根据命令形式的尾部参数来区分。

(1) 同步实时处理命令

如果在外部系统命令(感叹号命令)之后没有其他附加的参数,那么 MATLAB 会打开一个新的窗口作为该命令的运行窗口。MATLAB 系统要等到该命令完成之后,才开始接受新的命令。例如:

```
! dir
```

将生成一个新的窗口,在其中列出当前目录中的内容。虽然 MATLAB 执行完上述命令后,便回到了 MATLAB 的提示符状态,等待新的命令,但是列出目录内容的窗口要由用户来关闭。

(2) 后台处理命令

如果外部命令行以字符“&”结束,MATLAB 则将此命令作为后台命令处理,不必等到该命令完成之后,才接受和执行新的 MATLAB 命令。在需要打开新的 Windows 应用程序时,可以使用该命令。例如,发出下列命令:

```
! notepad&
```

后,将启动 Notepad 作为一个新的 Windows 任务。类似地,也可以发出 DOS 命令,这时将打开一个 DOS 窗口,但作为后台处理命令,此时可以在 MATLAB 命令窗口中执行其他的命令。

(3) 图标后台处理命令

第三种外部命令形式是以字符“|”结尾,这个命令的作用与后台处理命令相同,只是用一个图标作为后台命令打开的窗口。这个命令可以用在对命令的运行结果不感兴趣的情况下,例如,DOS 的批处理命令等。

(4) MATLAB 的 DOS 命令

另一种使用操作系统命令的方法是使用 MATLAB 的 DOS 命令。MATLAB 系统将部分的 DOS 命令作为自己的命令,执行时将这些命令直接传递给 DOS 操作系统,运行的结果在 MATLAB 的命令窗口中显示。有时,也打开一个 DOS 窗口,但该窗口在执行下一条新的 MATLAB 命令时自动关闭。也可以在命令后面加上“&”和“|”来改变窗口的形态。

1.1.3 MATLAB 文件类型

在 MATLAB 系统中,根据功能可将 MATLAB 系统所使用的外部文件分成几类,并用不同的扩展名作为其标识,如下所述。

1. M 文件

M 文件以字母 m 为其扩展名,例如 startup.m。一般说来,M 文件是 ASCII 码文本文件,可以用任何文本编辑器进行编辑。在 MATLAB 系统中,有两类 M 文件。一类称为程序 M 文件,简称 M 文件;另一类称为函数 M 文件,或简称为函数,统称为 M 文件。M 文件的内容是由符合 MATLAB 语法的语句构成的,函数 M 文件的第一行必须是以关键字 function 开始的函数说明语句。两类 M 文件的共同特征是:在 MATLAB 命令窗口中的命令提示符下键入文件名,来执行 M 文件中的所有语句规定的计算任务或完成一定的功能。它们的区别在于以下两方面:第一,程序 M 文件中创建的变量都是 MATLAB 工作空间中的变量,工

作空间中的其他程序或函数可以共享,而函数 M 文件中创建的所有变量除了全局变量外,均为局限于函数运行空间内的局部变量;第二,函数 M 文件可以使用传递参数,所以函数 M 文件的调用式中可以有输入参数和输出参数,而程序 M 文件则没有这种功能。

2. MAT 文件

MAT 文件是 MATLAB 系统的二进制数据文件,用于保存 MATLAB 系统所使用的数据。MATLAB 除了可以读写 ASCII 码形式的数据文件外,也定义了它自己的数据存储格式,这就是 MAT 文件。MAT 文件按照 MATLAB 的基本数据结构——矩阵的方式来管理和记录数据。对于每一个矩阵对象,MAT 文件记录了该矩阵对象的所有特性和各元素值。例如,矩阵对象的变量名、维数、矩阵的存储方式(即 MATLAB 的满矩阵或稀疏矩阵)等信息。如果用户要按照自己的方式读/写 MAT 文件,必须遵照 MATLAB 的规则进行读/写,或利用 MATLAB 系统提供的 MAT 数据接口函数来提取 MAT 文件中的数据,或向 MAT 文件写入自己的数据。

3. MEX 文件

MEX 文件是经过 MATLAB 编译系统编译的函数二进制文件。MEX 文件可以被直接调入 MATLAB 系统中运行。由于 MATLAB 是按边解释边运行的方式工作的,因此,M 文件的执行速度要比 MEX 文件慢得多。所以,用户通常把已经调试好,且比较大的 M 文件编译成 MEX 文件,供以后使用。

1.2 MATLAB 语言语法要素

MATLAB 语言的规则十分简单,它是一种表达式语言,其语句类似于数学式子的格式,十分容易掌握,下面分几个小节来介绍。

1.2.1 MATLAB 的矩阵、变量与表达式

在 MATLAB 系统中,只管理着一种对象(Object)——矩阵(包括复矩阵)。 1×1 的矩阵称为数量,或者说任何数量在 MATLAB 系统中是作为 1×1 的矩阵来处理的。与数学术语一样,仅有一行或一列的矩阵称为向量。MATLAB 的大部分运算或命令是在矩阵运算的意义下执行的。

1. 矩阵的创建

可以使用下列任何一种方法在 MATLAB 环境下创建或输入一个矩阵:

- (1) 显示地输入一个元素序列;
- (2) 用 MATLAB 的内部函数创建一个矩阵;
- (3) 在 M 文件中用 MATLAB 语句创建一个矩阵;
- (4) 从一个外部数据文件中装载并创建一个矩阵。

在 MATLAB 环境中,不需要对创建的变量对象给出类型说明和维数说明,所有的变量都作为双精度的矩阵来分配内存空间和存储空间。MATLAB 将自动地为每一个变量分配内存。最简单的创建矩阵的方法是显示地输入矩阵的元素序列。具体方法如下:将矩阵的元

素用方括号括起来,按矩阵行的顺序输入各元素,元素与元素之间用空格或逗号分开,用分号将每行的元素分开。例如,在键入下列的 MATLAB 语句:

```
>> A = [1 2 3; 4 5 6; 7 8 9]
```

后,MATLAB 执行该语句的输出结果是:

```
A =
1 2 3
4 5 6
7 8 9
```

这样,在 MATLAB 的工作空间中就创建了一个新的矩阵对象 A,以后就可以使用矩阵 A。也可以用回车键代替分号,按下列的方式输入:

```
>> A = [1 2 3
        4 5 6
        7 8 9]
```

对于大的矩阵,可以按矩阵的输入方式编辑一个 M 文件。例如,如果一个名为 mydata.m 的文本文件的内容如下:

```
A = [1 2 3
      4 5 6
      7 8 9]
```

那么,语句

```
>> mydata
```

将读入 M 文件 mydata.m,并执行其语句,生成同样的矩阵 A。另外,load 命令和 fread 函数都可以用来输入矩阵。

2. MATLAB 的变量和表达式

在 MATLAB 中有两个基本概念:变量和表达式。变量由变量名表示,函数名作为特殊的变量名看待,每个变量名由一个字母后面跟随任意个字母或数字(包括下划线)组成,但 MATLAB 只能分辨前 19 个字符。MATLAB 能区分组成变量名的大小写字母,这样,变量名 a 和 A 表示不同的变量,但所有的函数名要求是小写字母。例如,求矩阵 A 的逆用 inv(A) 时,而若使用 Inv(A) 则会遇到“未定义函数”的警告。表达式则是由运算符、函数调用、变量名以及特殊字符组成的类似于数学表达式的式子。

MATLAB 的语句则是下列两种形式之一:

```
>> 变量名 = 表达式
```

或者

```
>> 表达式
```

在 MATLAB 的矩阵管理方式中,MATLAB 每执行一条输入的语句,表达式运算求值的结果都是一个矩阵。在前一种语句形式下,MATLAB 将运算的结果赋给“变量名”;而在第二种语句形式下,将运算的结果赋给 MATLAB 的永久变量 ans,每条语句以回车符结束。一般地,运算的结果在命令窗口中显示出来。如果语句的最后一个字符是分号“;”,那么,MATLAB 仅仅执行赋值运算,不再显示运算的结果;如果运算的结果是一个很大的矩阵或

是对运算结果不关心，则可以在语句的最后加上分号。M 文件的语句常以分号结尾。例如：

```
>> p = conv(r, r);  
只计算 r 与它自己的卷积，不显示卷积的结果。
```

在一条语句中，如果表达式太复杂，一行写不下，可以加上三连点“...”并按下回车键，然后接下去再写。例如：

```
>> s = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 ...  
- 1/8 + 1/9 - 1/10 + 1/11 - 1/12;
```

这条语句计算级数的部分和，并将计算的结果赋给变量 s，但是不显示任何结果。

3. 复数的表示

MATLAB 提供对复数的操作与管理功能。在 MATLAB 中，复数的虚根单位用 i 或 j 表示。例如， $z=3+4*i$ 与 $z=3+4*j$ 表示的是同一个复数。又如，下面的两条语句输入的是相同的复数矩阵：

```
>> A = [ 1 2; 3 4 ] + i * [ 5 6; 7 8 ]  
>> A = [ 1+5i 2+6i; 3+7i 4+8i ]
```

第一种输入形式是将 i 看作数量，与矩阵作数量乘积；第二种形式是通常的矩阵创建方式，但要注意的是，此时作为矩阵元素的复数在输入时中间不能有任何空格。表达式 $1 + 5i$ （在加号的两边有空格）表示两个数相加，此时， $5i$ 就不是一个 MATLAB 的合法数据，因而表达式是错误的。例如， $1.23 \text{ e-}4$ ，(1.23 与 e 之间有空格) 也不是合法的 MATLAB 数据。

按照 MATLAB 的语法规则，MATLAB 内部函数的名字能够作为变量的名字。当内部函数名作为变量名时，该函数在当前的工作层中不能再被调用，直到该变量被清除为止。如果用 i 和 j 作为变量的名字，并且赋给了新的值，那么，i 和 j 不能再作为虚根单位使用。此时，可以用类似于下面的语句生成新的虚根单位：

```
>> ii = sqrt(-1)  
因此，建议读者将 i 和 j 作为 MATLAB 的保留字。
```

4. 数据的输入输出格式

MATLAB 用通常的十进制数表示常数、小数和负数。与通常的数学表示一样，还可以使用以 10 为幂的常数以及虚数。MATLAB 接受各种合法的数据输入，下面是一些合法的 MATLAB 型数据：

4	-99	0.00001
9.6397238	1.60210E-20	6.02252e23
20	-3.14159i	3e5i

在 MATLAB 内部，每一个数据元素都是用双精度数来表示和存储的。常数的相对精度是 eps，eps 是 MATLAB 的保留字，其值为 $2.220446049250313e-016$ 。按照 IEEE 浮点算术标准，大约有 16 位有效数字。MATLAB 能够表达的数值范围大致是 $10^{-308} \sim 10^{308}$ 。

除了在语句的后面有分号的情况外，MATLAB 将回显任何赋值语句的运算结果。MATLAB 按照一定的数据输出格式在 MATLAB 命令窗口中显示运算的结果，用户可以用 format 命令设置或改变数据输出格式。format 命令只影响数据输出格式，对 MATLAB 的内部计算和数据存储(MAT 文件)数值精度不产生任何影响。

如果输出矩阵的每个元素都是纯整数,MATLAB 就用不加小数点的纯整数格式显示结果。只要矩阵中有一个元素不是纯整数,MATLAB 将按当前的输出格式显示计算结果。缺省的输出格式是 short 格式,显示至 5 位有效数字。其他的输出格式可以给出更多的有效数字。作为一个例子,假设输入为

```
>> x = [ 4/3 1.2345e-6 ]
```

那么,在各种不同的输出格式下的输出为

```
>> format short  
1.3333 0.0000  
>> format short e  
1.3333e+00 1.2345e-06  
>> format long  
1.3333333333333 0.00000123450000  
>> format long e  
1.3333333333333e+00 1.234500000000e-06  
>> format bank  
1.33 0.00  
>> format hex  
3ff555555555 3eb4b6231abfd271  
>> format +  
++
```

5. 字符串与字符串变量

与 C 语言一样,MATLAB 将字符串当作数组或矩阵处理。在 MATLAB 语言中,字符串用单引号括起来(英文单引号字符用“'”表示)。例如:

```
>> s = 'Use MATLAB'
```

的输出结果是

```
s =  
Use MATLAB
```

字符串存储在行向量中,每个元素对应一个字符,其值为字符的 ASCII 码值。于是,字符串变量 s 是 1×10 的矩阵,它包括 Use 与 MATLAB 之间的一个空格字符。一些数学函数也可以应用在字符串变量上。例如,对字符串变量求绝对值 abs(s),其结果是字符串中各字符的 ASCII 码值组成的向量,尽管这个向量在维数与数值上与 s 的内部表示一样,但是它们的变量属性是不同的。事实上,MATLAB 对每个变量都定义了一个属性来说明该变量是否是一个字符串变量。

MATLAB 提供了几个与字符串操作有关的函数:函数 setstr 的作用是将 ASCII 码值转换成可显示的字符;disp 是将字符串变量的字符直接显示出来;isstr 用于检查一个变量是否为字符串变量;strcmp 用于字符串的比较;printf 将数值按指定的格式转换成数值字符串;num2str 和 int2str 也是将数值转换成字符串,读者可以看一看下列语句的运行结果:

```
>> f=70; c=(f-32)/1.8;
```

```
>> disp(['Room temperature is ', num2str(c), 'degrees C.'])
```

与字符串有关的另一个函数是 eval, 该函数的调用形式为 eval(t), 其中 t 是字符串变量, 它的作用是把该变量的内容作为表达式或语句进行求值。例如, 设 $t='100/20'$, 则

```
>> t
ans =
100/20
>> eval(t)
ans =
5
```

这个函数在程序设计中将起到很大的作用。

6. 其他数据结构

MATLAB 5.0 版提供了几种新的数据结构, 下面分别作简单的介绍。

(1) 多维数组 (Multidimensional Arrays)

通常 MATLAB 的基本数据结构是矩阵, 其维数是 2, 也就是说, 矩阵的每一个元素可以用给出两个角标的方法来访问。多维数组的数据结构在很多程序设计语言中都有定义, MATLAB 5.0 版提供了这种数据结构。如果数组的角标数多于 2, 则这样的数组就称为多维数组。在 5.0 版本中, 一些用于创建特殊矩阵的标准 MATLAB 函数, 如 zeros, ones, rand, randn 等都可扩展为创建多维数组的函数。例如:

```
>> R = randn(3,4,5);
```

表示创建一个阶为 $3 \times 4 \times 5$ 的 3 维随机数组 R, 其元素总数为 60。

3 维数组可以表示 3 维物理数据。在 MATLAB 中, 3 维数组最重要的应用是表示矩阵序列 $\{A^{(k)}\}$, 此时, 序列中第 k 个矩阵的第(i,j)个元素就可以用 $A(i,j,k)$ 表示。

如果 A 是一个幻方矩阵, 交换 A 的任何两列, 得到的仍是一个幻方矩阵。于是, 可以用一个 3 维数组将所有按这种方法创建的幻方矩阵记录下来。例如:

```
>> A = magic(4);
>> p = perms(1:4);
>> M = zeros(4,4,24);
>> for k = 1:24
    M(:,:,k) = A(:,p(k,:));
end
```

假设第二条语句创建一个 24×4 阶矩阵, 每个行向量都是向量 $1:4$ 的某个排列(共有 24 个不同的排列)。于是, 对任何 $1 \leq k \leq 24$, $M(:,:,k)$ 是一个 4 阶幻方矩阵。

很多应用于矩阵的 MATLAB 函数被扩展后都可应用于多维数组, 例如:

```
>> sum(M,d)
```

表示 n 维数组 M 按第 d 角标方向进行求和, 其结果应该是一个 $n-1$ 维数组。其他许多面向列运算的 MATLAB 函数都可以按这种方式使用, 如 min, max 等, 参见本书 1.4.3 节。

(2) 块数组 (Cell Arrays)

在 MATLAB 5.0 版中, 另一种新的数据结构是所谓的块数组。块数组的元素是某些其

他数组的拷贝值。通常，块数组对象可以用 `cell` 函数创建，也可以用花括号运算符 {} 创建。访问块数组的元素的方法与访问多维数组的方法是一样的，只不过是用花括号将角标值括起来，再加上块数组变量名而已。例如：

```
>> C = {A sum(A) prod(prod(A))}
```

创建一个 1×3 的块数组。在上述语句中，如果 A 是 4 阶幻方矩阵，则它的运算结果如下：

```
C =
```

```
[4×4 double] [1×4 double] [20922789888000]
```

即 C(1) 是一个 4×4 阶矩阵，C(2) 是 1×4 的行向量，而 C(3) 是一个数量。

一般来说，3 维数组表达同阶的矩阵序列，而块数组可以表达不同阶的矩阵序列。例如：

```
>> M = cells(8,1);
```

```
>> for n = 1 : 8
```

```
    M{n} = magic(n);
```

```
end
```

```
>> M
```

```
M =
```

```
[ 1]
[ 2×2 double]
[ 3×3 double]
[ 4×4 double]
[ 5×5 double]
[ 6×6 double]
[ 7×7 double]
[ 8×8 double]
```

(3) 结构(Structures)

结构是 MATLAB 5.0 的一种新型数组，它的元素既不像多维数组那样是数值，也不像块数组那样是多维数组。它的元素可以认为是一种记录，本身具有不同的域，元素要用域操作符访问。例如：

```
>> S.name = 'Jun Gao';
```

```
>> S.score = 83;
```

```
>> S.grade = 'B+';
```

表示创建一个数量结构对象，具有 3 个域，分别是 `name`, `score`, `grade`。很明显，这样的结构适于记录一个班级学生的成绩。

结构可以是数组，所以可以用增加数组元素的方法来增加结构型数组的元素。在这种情况下，结构型数组的每一个元素必须有相同的域。MATLAB 5.0 提供的函数 `struct` 可用于对整个结构的元素赋值。例如：

```
>> S(2) = struct('name','Qian Chen', ...
```

```
    'score', 70, 'grade', 'C')
```

```
S =
```

```
1×2 struct array with fields:
  name
  score
  grade
```

用结构名加上域名可以提出结构数组中所有元素相应域的值。例如：

```
>> [S.score]
```

```
ans =
```

```
83    70
```

这样就将 S 结构中的 score 域的值提出来创建一个向量。

1.2.2 MATLAB 的基本管理命令

为了方便用户使用 MATLAB，在 MATLAB 5.0 版中，系统提供了强大的在线帮助功能。根据 HTML 技术编写的帮助文件使用起来十分方便。另外，在 MATLAB 命令窗口的工作环境下，也可以使用 help 命令，直接向系统寻求帮助。这里不打算作详细介绍，读者只要启动 MATLAB，就会发现 HELP 功能使用起来十分得心应手。这一节仅扼要地介绍几个常用的系统管理命令。

1. 查询变量信息的命令 who 和 whos

有时用户需要知道在 MATLAB 的工作环境中有哪些变量，这时可以使用 MATLAB 命令 who 和 whos 来查看。这两个命令的区别在于，前一个命令只是简单地列出在工作环境中的变量名字；而后一个命令除了列出变量的名字外，还报告每个变量更详细的信息，通过一张表格向用户报告系统工作环境中存在的每个变量，它们的维数(size)，变量所对应矩阵的元素个数，在内存中占用的空间及其矩阵属性(如是否为稀疏矩阵，是否为复矩阵，等等)。

2. 装入和存储变量与数据的命令 load 和 save

MATLAB 提供了两个命令，用于装入和保存数据与变量之值，下面分别进行介绍。

(1) load 命令

该命令有两种调用形式，其一用于装入 MATLAB 格式的数据文件，即 MAT 文件，一般以 mat 为扩展名，装入的数据变量的名字等信息都包含在该文件中。例如：

```
>> load clown
```

表示在当前目录或搜索路径下寻找 MATLAB 数据文件 clown.mat，并把它装入 MATLAB 的工作环境中，创建的变量名是根据该文件的内容确定的(注：clown.mat 是随图像处理工具箱一起安装的一个图像数据文件)；另一种形式用于装入 ASCII 码文件形式的矩阵数据，这种数据文件可以用普通的文本编辑器编辑，其中每一行的数据恰为矩阵的每一行数据，数据元素之间用空格隔开，每行以回车符结尾，且每行的数据个数必须相同。装入 MATLAB 的工作环境后，这个矩阵以该文件的名字为变量名。例如：

```
>> load mydata.dat
```

表示在当前目录下寻找 MATLAB 数据文件 mydata.dat，并将其装入到 MATLAB 的工作

环境空间中, 创建的变量是 mydata。

(2) save 命令

与 load 命令相对应的是 save 命令, 它也有两种调用形式。例如, 为了将当前变量或某些指定的变量存入 MATLAB 数据文件 temp.mat, 可以用下列的调用方式:

» save temp

或

» save temp X Y Z

另一种调用方式可以根据要求将当前变量或指定变量的数据按 ASCII 格式存储, 例如:

» A = rand(3, 4);

» save temp.dat A -ascii

表示生成一个名字为 temp.dat 的 ASCII 文件, 其内容是由第一条语句创建的 3 行 4 列的随机矩阵

```
0.2113 0.8096 0.4842 0.0824
0.8217 0.8234 0.2354 0.3212
0.2345 0.6714 0.6528 0.2985
```

这两种命令都可以不带任何参数地调用, 此时, save 的作用是将当前工作环境中的所有变量存入到文件 matlab.mat 中, 而 load 则是将 matlab.mat 的内容重新装入, 恢复到用 save 存入 matlab.mat 时的状态。

3. 清除变量与数据的命令 clear

该命令用于清除工作环境中的全部或部分指定的变量。具体的调用方式如下所列:

» clear 清除所有的变量

» clear X Y Z 清除名为 X, Y 和 Z 的变量

» clear functions 从内存中清除所有已编译的函数

» clear variables 清除所有的变量

» clear mex 从内存中清除所有的 MEX 文件

» clear global 清除所有的全程变量

» clear all 清除所有的变量、函数、MEX 文件等

4. 退出 MATLAB 系统的命令 quit 和 exit

为了退出 MATLAB 系统, 既可以用命令窗口中 FILE 菜单下的 exit 命令退出, 也可以在命令行下键入命令 quit 或 exit 退出。

1.2.3 MATLAB 的基本运算符

1. 矩阵(算术)运算符

MATLAB 的主要矩阵(算术)运算符如下所列:

A' 矩阵 A 的转置, 如果 A 是复矩阵, 则其运算的结果是共轭转置

A±B 矩阵 A 和 B 的和与差, 其中的一个矩阵可以是数量, 表示另一个矩阵的元素加减该数量

$A * B$	矩阵 A 和 B 的乘法, A 与 B 均可以是向量或数量, 只要符合矩阵乘法的定义
A/B	方程 $X * A = B$ 的解 X
$A \setminus B$	方程 $A * X = B$ 的解 X
A^B	A 和 B 都为数量时, 表示数量 A 的 B 次方幂; A 为方阵, B 为正整数时, 表示矩阵 A 的 B 次乘积; B 为负整数时, 表示 A 的逆矩阵的 B 次乘积; 而当 B 为非整数时, $A^B = V * \begin{bmatrix} \lambda_1^B & & \\ & \ddots & \\ & & \lambda_n^B \end{bmatrix} / V$, 其中 $\lambda_1, \dots, \lambda_n$ 为矩阵 A 的特征值, V 为对应的特征向量矩阵。当 A 为数量, B 为方阵时, $A^B = V * \begin{bmatrix} A^{\lambda_1} & & \\ & \ddots & \\ & & A^{\lambda_n} \end{bmatrix} / V$, 其中 $\lambda_1, \dots, \lambda_n$ 为矩阵 B 的特征值, V 为对应的特征向量矩阵。当 A 与 B 都为矩阵时, 无定义
$A.*B$	矩阵 A 和 B 的对应元素相乘, A 和 B 为同维数的矩阵, 除非其中之一为数量
$A./B$	矩阵 A 的元素除以矩阵 B 的对应元素, 即等于 $[A(i,j)/B(i,j)]$, A 和 B 为同维数的矩阵, 除非其中之一为数量
$A.\setminus B$	矩阵 B 的元素除以矩阵 A 的对应元素, 即等于 $[B(i,j)/A(i,j)]$, A 和 B 为同维数的矩阵, 除非其中之一为数量
$A.^B$	等于 $[A(i,j)^B]$, A 和 B 为同维数的矩阵, 除非其中之一为数量
$A.'$	也表示矩阵 A 的转置, 但当 A 为复矩阵时, 不求共轭

2. 关系运算符

MATLAB 提供了 6 种关系运算符, 用于比较两个同维数的矩阵, 即

<	小于
\leq	小于或等于
>	大于
\geq	大于或等于
$=$	等于
\neq	不等于

MATLAB 比较两矩阵的对应元素, 例如 $C = A < B$ 的结果 C 是与 A 或 B 同维数的 0-1 矩阵, 1 表示比较的结果为真, 0 表示比较的结果为假。特别地, A 与 B 之一可以是数量。例如:

```
>> 2 + 2 ~== 4
ans =
    0
```

关系运算符可以用于检查矩阵的元素是否满足某些条件。例如, 设矩阵 A 是一个 6 阶幻方矩阵

```
A = magic(6)
A =
    35    1    6   26   19   24
      3   32    7   21   23   25
    31    9    2   22   27   20
      8   28   33   17   10   15
    30    5   34   12   14   16
      4   36   29   13   18   11
```

n 阶幻方矩阵是一个 $n \times n$ 的矩阵, 其元素由 1 到 n^2 的 n^2 个整数组成, 每行、每列元素的和相同。下面的语句可以用来检验矩阵 A 的元素是否可以被 3 整除:

```
P = (rem(A, 3) == 0)
```

在上面的语句中, `rem(A, 3)` 是矩阵 A 的每个元素除以 3 的余数矩阵, 此时, 0 被扩展为与 A 同维数的零矩阵, P 是进行 == 比较的结果矩阵:

```
P =
    0  0  1  0  0  1
    1  0  0  1  0  0
    0  1  0  0  1  0
    0  0  1  0  0  1
    1  0  0  1  0  0
    0  1  0  0  1  0
```

常与关系运算符一起使用的 MATLAB 函数是 `find`, 该函数能在 0-1 矩阵中找出非零元素, 并返回非零元素在矩阵中的位置指标向量。例如, 如果 Y 是一个向量, 那么 `find(Y < 3.0)` 的结果是一个向量, 其值为向量 Y 中小于 3.0 的元素的位置指标。

3. 逻辑运算符

MATLAB 提供了 3 种逻辑运算符, 即“与”运算符 `&`、“或”运算符 `|` 和“非”运算符 `~`。它们的定义如下:

`A&B` 结果是 0-1 矩阵, 其元素为 1 时, 表示矩阵 A 和 B 的对应元素都为非零值; 否则该元素为 0。矩阵 A 和 B 是同维数的, 除非其中之一为数量

`A|B` 结果是 0-1 矩阵, 其元素为 1 时, 表示矩阵 A 和 B 的对应元素中至少有一个为非零值; 其元素为 0 时, 表示矩阵 A 和 B 的对应元素都为 0。矩阵 A 和 B 是同维数的, 除非其中之一为数量

`~A` 结果是 0-1 矩阵, 矩阵 A 的元素为零, 则该矩阵对应的元素为 1; 矩阵 A 的元素非零时, 对应的元素为 0

常与逻辑运算符一起使用的 MATLAB 函数有 `any` 和 `all`。如果向量 x 的某一个元素为非零, 则 `any(x)` 的返回值为 1(真), 否则返回值为 0(假)。如果向量 x 的每一个元素都为非零, 则 `all(x)` 的返回值为 1(真), 否则返回值为 0(假)。这些函数都将返回一个条件值, 因此, 在条件语句中特别有用。例如:

```
if all(A < .5)
```

```
do something
end
```

对于矩阵变量,这些函数的返回值是一个行向量,其值为矩阵的每列进行 any 和 all 运算的结果。这样,两次迭代使用这些函数可以得到一个数量的条件值,例如: any(any(A))。

MATLAB 提供的关系运算和逻辑运算函数主要有:

any	逻辑条件
all	逻辑条件
find	提取逻辑值的行列指标
exist	检测某个变量的存在性
isnan	检测 NaN
isinf	检测无穷大
finite	检测有限值
isempty	检测空矩阵
isstr	检测字符串
isglobal	检测全程变量
issparse	检测稀疏矩阵

例如, exist(x) 可检测变量 x 在工作环境中是否存在,如果存在该变量,函数的返回值为 1,否则返回值为 0。

4. 特殊运算符

除了常用的数学运算符外,MATLAB 还提供了几种特殊运算符,下面分别加以介绍。

在 MATLAB 的 M 文件中,可以加入解释行。解释行的标识符为“%”,该标识符后的内容将被作为注解内容。程序执行时,注解被忽略。

方括号“[]”运算符用于生成矩阵。特别地,语句 A = []生成空矩阵 A。空矩阵是 MATLAB 提供的一种理想元,其维数为零,无任何元素,但其行列式为 1。空矩阵的作用在本书后续章节还要作介绍。

行分隔符“;”用在 MATLAB 语句后时,表示该语句的执行结果不被回显出来,这可避免显示一些不感兴趣的结果。

冒号运算符“:”最主要的作用是生成向量,从下面的例子中可以看出其使用方法:

j : k	生成向量 [j, j+1, j+2, …, k]
j : i : k	生成向量 [j, j+i, j+2i, …, k] 如果 j > k,则生成空矩阵
A(:, j)	矩阵 A 的第 j 列
A(i, :)	矩阵 A 的第 i 行
A(j : k)	向量 A(j), A(j+1), …, A(k)
A(:, j : k)	从第 j 列到第 k 列的矩阵子块

换行连接符“…”,有时一条 MATLAB 语句会很长,在命令窗口的一行内很可能写不下,此时只要在该行语句中加入三连点,再回车即可在下一行接着写该语句。例如:

```
>> A = a ^ 2 + b ^ 2 + ...
```

```
sin(0.6 * pi);
```

与下列语句

```
>> A = a^2 + b^2 + sin(0.6 * pi);
```

的作用是相同的。

1.2.4 MATLAB 的常用数学函数

1. 初等函数

MATLAB 提供了几乎所有的常用的初等函数, 函数的变量在 MATLAB 中被规定为矩阵变量, 运算法则是将函数逐项作用于矩阵的元素上, 因而运算的结果是一个与自变量同维数的矩阵。例如:

```
>> A = [ 1 2 3; 4 5 6 ]
>> B = fix(pi * A)
>> C = cos(pi * B)
```

这三条语句运算的结果是:

```
A =
    1   2   3
    4   5   6
B =
    3   6   9
   12  15  18
C =
   -1   1   -1
    1  -1   1
```

MATLAB 提供的三角函数主要包括有:

sin	正弦函数
cos	余弦函数
tan	正切函数
asin	反正弦函数
acos	反余弦函数
atan	反正切函数
sinh	双曲正弦函数
cosh	双曲余弦函数
tanh	双曲正切函数
asinh	反双曲正弦函数
acosh	反双曲余弦函数
atanh	反双曲正切函数

MATLAB 提供的初等函数包括有:

abs	实数的绝对值、复数的模、字符串的 ASCII 码值
angle	复数的幅角
sqrt	方根函数
real	复数的实部
imag	复数的虚部
conj	复共轭运算
round	最邻近整数截断(四舍五入)
fix	向零方向截断为整数
floor	不大于自变量的最大整数
ceil	不小于自变量的最小整数
sign	符号函数
rem	求余数或模运算
gcd	最大公因子
lcm	最小公倍数
exp	自然指数函数(以 e 为底)
log	自然对数函数(以 e 为底)
log10	以 10 为底的对数函数

2. 特殊函数

为了方便应用,MATLAB 还提供了一些特殊的数学函数,主要包括有:

bessel	Bessel 函数
beta	完全与不完全 Beta 函数
gamma	完全与不完全 Γ 函数
rat	有理逼近
erf	误差函数
erfinv	逆误差函数
ellipk	第一类型、第二类型完全椭圆函数
ellipj	Jacobi 椭圆函数

上述函数基本上按矩阵的元素进行运算。

1.3 简单程序设计

1.3.1 控制语句

大部分的求解问题都要受到一定条件的控制,与其他的程序设计语言一样,MATLAB 语言也提供了条件语句。下面分别加以介绍。

1. for 循环语句

MATLAB 也有自己的 for 循环语句。如果要反复执行的一组语句的循环次数是已知或

预定义的,就可以使用 for 循环语句。例如:

```
>> for i=1:n, x(i) = 0, end
```

这条循环语句将向量 x 的前 n 个元素赋零值,这里变量 n 的值必须预先给定。如果变量 n 的值小于 1,在这种情况下,语句仍是合法的,但是 MATLAB 不执行循环内的语句,即 $x(i) = 0$ 。如果 x 事先不存在或者其元素的个数少于 n,那么 MATLAB 将自动分配所需要的空间。

for 语句可以嵌套使用,例如:

```
>> for i = 1:m
    for j = 1:n
        A(i, j) = 1/(i+j-1);
    end
end
>> A
```

其中分号的作用是避免逐个显示矩阵元素的值,而最后一行表示显示循环语句创建的矩阵 A。

最重要的是,每一个 for 必须与 end 配对使用。例如,输入下列语句

```
>> for i = 1:n, x(i) = 0
```

后,MATLAB 将继续等待循环体的输入,直至遇到 end 结束循环体时,才开始执行 for 语句。

再看一个例子,设 t 是一个列向量:

```
t =
```

```
-1
0
1
3
5
```

如果要创建一个 Vandermonde 矩阵 A,其各列元素正好是 t 的各次幂,即

```
A =
```

```
1 -1 1 -1 1
0 0 0 0 1
1 1 1 1 1
81 27 9 3 1
625 125 25 5 1
```

那么,最常用的方法是使用下面的循环语句:

```
>> n = length(t);
>> for j = 1:n
    for i = 1:n
        A(i,j) = t(i)^(n-j);
    end
```

```
end
```

但是,如果要使用 MATLAB 提供的矩阵向量运算功能,则用下面基于向量运算的单个循环语句,其执行速度会更快:

```
>> A(:,n) = ones(n,1);
>> for j = n-1:-1:1
    A(:,j) = t.*A(:,j+1);
end
```

其中, `ones(n,1)` 的作用是生成一个 n 行 1 列的元素全为 1 的向量。

`for` 循环语句的一般形式如下:

```
for v = 表达式
    语句组
end
```

按照 MATLAB 的定义,表达式是一个矩阵,上述 `for` 循环语句的执行过程是:依次地将表达式矩阵的各列之值赋给变量 v ,然后执行语句组中的语句。换句话说,MATLAB 的 `for` 循环语句的循环变量 v 可以是一个列向量。在这种情况下,该循环语句等价于下面的语句组:

```
E = 表达式;
[m,n] = size(E);
for j = 1:n
    v = E(:,j);
    语句组
end
```

通常,表达式的形式是 $m:n$ 或 $m:i:n$,即仅为一行的矩阵,因而列向量为单个数量。在这种特殊形式下,MATLAB 的 `for` 循环语句与其他计算机语言的 FOR 或 DO 循环语句等价。

2. while 循环语句

MATLAB 提供有 `while` 循环语句,它的作用是在一定的逻辑条件控制下,不断地循环执行一条或一组语句,直到逻辑条件不再满足时为止。下面给出一个简单的例子来说明 `while` 语句的作用:

```
>> n = 1;
>> while prod(1:n) < 1.e100, n = n+1; end
>> n
```

这组语句的语义是找出使阶乘 $n!$ 小于 10 的 100 次方的最小整数 n ,其中 `prod(1:n)` 是计算向量 $1:n$ 各元素之积。

`while` 语句的一般形式是

```
while 表达式
    语句组
end
```

其语义是当表达式矩阵的各个元素都是非零值时,执行语句组并再循环。一般来说,这里的表达式是 1×1 的矩阵形式。如果问题涉及到某个非 1×1 维数的矩阵,那么,可以用函数any或all将这个矩阵条件转换成 1×1 的矩阵表达式。例如,对向量v,当它的所有元素为非零值时,all(v)取值为1,否则为0。

3. if 条件语句和 break 语句

这里首先通过例子来说明if条件语句的使用方法。第一个例子是根据一个整数的符号和奇偶性,分成三种情况作不同的计算:

```
if n < 0
    A = negative(n)
elseif rem(n,2) == 0
    A = even(n)
else
    A = odd(n)
end
```

第二个例子是一个有趣的数论问题,任取一个正整数,如果它是偶数,就除以2;如果是奇数,就乘3再加1,重复这个过程直至这个整数变成1时为止。问题是:对于任何一个整数,该过程是否能够终止?结合while和if语句的MATLAB语句如下:

```
while 1
    n = input('Enter n, negative quits : ');
    if n <= 0, break, end
    while n > 1
        if rem(n,2) == 0
            n = n/2
        else
            n = 3 * n + 1
        end
    end
end
```

其中,input函数的作用是等待用户的键盘输入,并将输入的值存入变量n中,而break语句则用于退出循环体。if条件语句有两种形式,分别是

if 表达式

语句组 1

else

语句组 2

end

和

if 表达式 1

语句组 1

```
elseif 表达式 2
```

```
语句组 2
```

```
else
```

```
语句组 3
```

```
end
```

它们的语义与通常的程序设计语句的语义是相同的。

1.3.2 M 文件、MATLAB 函数与函数型函数

MATLAB 采用的是行命令模式, 用户每输入一行命令, 回车后 MATLAB 就解释并执行这条命令, 并根据要求显示运算结果。此外, MATLAB 也可以执行某个文件中的 MATLAB 语句序列(类似于 DOS 的批处理文件)。这两种模式一起, 构成了 MATLAB 的解释环境。

包含 MATLAB 语句序列的文本文件称为 M 文件, 其文件名以 m 为扩展名。例如, 文件 bessel.m 包含的 MATLAB 语句用于计算 Bessel 函数。在 M 文件的语句中可以引用另外的 M 文件, 也可以递归地调用其自身。M 文件可以用任何文本编辑器生成, 例如用 Windows 95 的 Notepad 编辑器生成。MATLAB 定义了两种类型的 M 文件: 程序 M 文件和函数 M 文件。程序 M 文件的内容是 MATLAB 语句流(也可称为批处理), 而函数 M 文件总是提供某种特定的处理功能, 作为 MATLAB 函数的扩展。程序 M 文件和函数 M 文件都是 ASCII 码文本文件。

1. 程序 M 文件

当一个程序 M 文件启动后, MATLAB 就解释执行它所遇到的每一条 MATLAB 语句。每条语句都是在 MATLAB 的工作环境中执行的。为了设计、修改、分析一段较长的 MATLAB 语句序列, 可以将这些语句编写成一个程序 M 文件, 以便反复运行与修改。下面是一个程序 M 文件的例子, 假设它的文件名字是 fibno.m, 其语句序列如下:

```
%An M-file to calculate Fibonacci numbers
f = [ 1 1 ]; i = 1;
while f(i) + f(i+1) < 1000,
    f(i+2) = f(i) + f(i+1);
    i=i+1;
end
plot(f)
```

在 MATLAB 命令窗口中键入 fibno 命令后, MATLAB 将计算前 16 个 Fibonacci 数, 并画出一幅图。该文件执行完毕时, 变量 f 和 i 保留在 MATLAB 工作环境中。

MATLAB 提供的演示程序说明了如何用程序 M 文件求解复杂问题。每次启动 MATLAB 系统时, 系统自动执行 startup.m 文件。用户可以根据自己的需要将可能用到的有关常数等写入该文件中, 这样, 这些常用量就被装入到 MATLAB 的工作环境中。该文件的作用有点类似于 DOS 系统的 autoexec.bat 文件。另一个 M 文件是 matlabrc.m, 它的作

用是设置一些必要的系统参数,类似于 DOS 系统的 config.sys 文件。

2. 函数 M 文件

如果 M 文件的第一行以关键字 function 开始,则称该行为函数说明语句,这个文件称为函数 M 文件。函数 M 文件与程序 M 文件的主要区别是函数 M 文件可以用来传递参数或变量,而且函数 M 文件中定义和管理的变量都是局部变量。当函数 M 文件运行完毕时,这些变量被清除。函数 M 文件主要用于编写新的 MATLAB 函数。下面是一个简单的函数 M 文件的例子,其文件名为 mean.m:

```
function y = meanvalue(x)
MEAN Average or mean value
% For vectors, MEAN(x) returns the mean value
% For matrices, MEAN(x) is a row vector
% containing the mean value of each column
%
[m, n] = size(x);
if m == 1
    m = n;
end
y = sum(x)/m;
```

该文件定义一个新的函数 mean。在命令窗口的提示符下,用函数 M 文件的文件名和附加的调用参数(如果需要的话)可以调用函数。在上述例子中,设 z 是从 1 到 99 的向量,即 z=1 : 99,则下列函数调用

```
>> mean(z)
```

的结果是

```
ans =
```

```
50
```

3. 函数型函数

在 MATLAB 中,有一类函数是以某些数学函数为处理对象的(或它的输入参数为另一个函数),这种函数在 MATLAB 中被称为函数型函数。MATLAB 的基本系统中已定义的函数型函数有:数值积分;非线性方程求解及其优化;常微分方程数值求解。

MATLAB 用函数 M 文件定义数学函数。例如,函数

$$f(x) = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04} - 6$$

可以用名为 humps.m 的函数 M 文件来实现:

```
function y=humps(x)
y=1./((x-0.3).^2+0.01)+1./((x-0.9).^2+0.04)-6;
```

该函数在区间 [-1, 2] 上的图形可以用下列语句得到:

```
>> x = -1 : .01 : 2;
>> plot(x, humps(x));
```

(1) 数值积分(求积公式)

函数 $f(x)$ 与坐标横轴所围区域的代数面积可以由它的定积分求得。MATLAB 函数 quad 可以用来求定积分的近似值。该函数以被积函数为第一输入参数, 定积分的下限与上限分别为第二和第三输入参数。前面定义的函数 humps.m 在 $[0,1]$ 上的定积分为

```
>> q = quad('humps', 0, 1)
q =
29.8583
```

函数 quad 使用自适应的 Simpson 求积规则求解。另一个数值求积的函数型函数是 quad8, 其算法为 8 节点自适应 Newton-Cotes 求积规则。注意: 函数 quad 和 quad8 的第一个输入参数是单引号括起来的函数名, 它们是施加在另一个函数上的运算, 在这种意义下, 把它们称为函数型函数。

(2) 非线性方程求解与函数优化

下面几个 MATLAB 函数型函数用于求解非线性方程和函数优化问题:

fmin	单变量函数的极小值
fmins	多变量函数的极小值(无约束非线性优化)
fzero	单变量非线性方程求根

仍以函数 humps.m 为例, 该函数在 $[0.5,1]$ 区间上的最小值位置可由 fmin 函数求出:

```
>> xm = fmin('humps', 0.5, 1);
xm =
0.6378
```

最小值的近似值为

```
>> y = humps(xm)
y =
11.2528
```

从 humps 的图形中可以看出, 该函数有两个零点, 在 $x=0$ 附近的零点是

```
>> xz1 = fzero('humps', 0)
xz1 =
-0.1316
```

而 $x=1$ 附近的零点是

```
>> xz2 = fzero('humps', 1)
xz2 =
1.2995
```

在 MATLAB 的最优化工具箱中, 还增加了 7 个用于求解非线性方程与函数优化的函数型函数, 它们分别是:

attgoal	多目标优化函数
constr	约束最优化函数
fminn	无约束最优化函数
fsolve	非线性方程组求解

leastsq 非线性最小二乘解

minimax 极小极大解

seminf 半无限优化

(3) 常微分方程数值求解

用于常微分方程数值求解的 MATLAB 函数型函数有：

ode23 2 阶/3 阶 Runge-Kutta 方法

ode45 4 阶/5 阶 Runge-Kutta-Fehlberg 方法

例如, 考查二阶 Van der Pol 方程

$$\ddot{x} + (x^2 - 1)\dot{x} + x = 0$$

的数值解。这个方程可以改写成等价的一阶常微分方程组

$$\dot{x}_1 = x_2(1 - x_2^2) - x_1$$

$$\dot{x}_2 = x_1$$

第一步工作是编写描述这组方程的 M 文件, 该文件的名字设为 vdpol.m:

```
function xdot = vdpol(t, x)
xdot = zeros(2, 1);
xdot(1) = x(1). * (1-x(2). ^ 2)-x(2);
xdot(2) = x(1);
```

为了在区间 $0 \leq t \leq 20$ 上求解由 vdpol.m 定义的微分方程, 启动函数 ode23 如下:

```
>> t0 = 0; tf = 20;
>> x0 = [0, 0.25]';
>> [t, x] = ode23('vdpol', t0, tf, x0)
>> plot(t, x)
```

有关系统模拟的内容, 读者可以参考 MATLAB 附带的有关 SIMULINK 使用的帮助文件。

4. 变量 nargin 和 nargout

在前面的函数 M 文件 mean.m 中, 有一个输入形式参数 x 和一个输出形式参数 y。定义函数 M 文件时, 形式参数的个数可以是任意多个。在调用函数时, MATLAB 用两个永久变量 nargin 和 nargout 分别记录调用语句的输入和输出实参数的个数。这样, MATLAB 允许调用语句中的实参数个数可以与函数 M 文件定义中的形式参数个数不同。

5. 编写函数的在线帮助

用户可以为自己的函数编写在线的帮助信息。在 MATLAB 环境中, 下面的命令

`>> help 函数名`

将显示出该函数 M 文件的注释信息, 即函数说明语句后几条连续的注释行。例如, 本书 1.3.2 节中的 M 文件 mean.m 的 2~5 行。因此, 用户可以将有关的对函数功能进行说明的信息作为注释行写在函数说明语句的后面。help 命令只是显示跟随在函数说明语句后的连续的注释行块, 而忽略其他的注释行。故应该将最主要的说明放在第一行注释行内。在有些情况下, MATLAB 只访问第一行的内容。例如, MATLAB 命令 lookfor 就只在第一行中查

找用户指定的关键字。

6. 函数的编译

当用户第一次调用一个函数时, MATLAB 会对其进行编译并将其装入内存。如果再次调用该函数, 则不用再编译, 直到退出 MATLAB 系统或用户将该函数从内存中清除。what 命令可以用来显示当前目录下可用的函数 M 文件名, type 命令可用于打印指定的 M 文件内容。当用户输入一个名字, 例如 whoopie 时, MATLAB 将按照下列的步骤作解释:

- (1) 在工作空间中寻找变量 whoopie;
- (2) 检查 whoopie 是否为 MATLAB 的内部函数;
- (3) 在当前目录中查找文件 whoopie.m;
- (4) 在 MATLAB 搜索路径的目录中查找文件 whoopie.m。

这样, 对每个名字, MATLAB 首先将其作为变量在工作环境中查找, 然后再作为函数解释。如果不是以上这些情况, MATLAB 系统就报告错误信息。

1.3.3 全程变量

通常每一个由 M 文件定义的函数, 都有自己的局部变量, 这些变量独立于其他函数, 独立于工作环境, 独立于任何非函数的普通 M 文件。函数的输出量可以赋给工作环境中的任何变量, 达到修改该变量的目的。而函数的局部变量在退出函数体时被系统自动清除。但是, 在一些函数内将某个变量说明为全程变量, 则这些函数之间甚至与工作环境之间可以共享全程变量的同一个拷贝值。在任何一个这样的函数中, 对全程变量的任何赋值, 其他的函数再使用该全程变量时, 将按新的值用于计算。换句话说, 若在任何一个地方改变了全程变量的值, 则其后所有用到全程变量的地方都以新的值作计算。为了便于区别, 在程序中习惯于用大写字母表示全程变量。例如, 假设为研究 Lotka-Volterra 捕食模型

$$\begin{aligned}\dot{y}_1 &= y_1 - \alpha y_1 y_2 \\ \dot{y}_2 &= -y_2 + \beta y_1 y_2\end{aligned}$$

中的系数 α 和 β 对方程解的影响, 首先编写一个名为 lotka.m 的函数 M 文件, 其内容如下:

```
function yp = lotka(t, y)
% LOTKA Lotka-Volterra predator-prey model
global ALPHA BETA
yp = [ y(1)-ALPHA * y(1) * y(2); -y(2)+BETA * y(1) * y(2) ];
```

然后交互地输入下列语句(第一句输入一次即可):

```
>> global ALPHA BETA
>> ALPHA = 0.01
>> BETA = 0.02
>> [t, y] = ode23('lotka', 0, 10, [1; 1]);
>> plot(t, y)
```

上述两条全程变量说明语句 global 说明了变量 ALPHA 和 BETA 是全程变量, 对 ALPHA 和 BETA 的任何赋值, 函数 lotka 可以立即使用。输入不同的 ALPHA 和 BETA 即可得到

方程的不同的解。在上面的语句中,函数 `ode23` 用于对由函数 M 文件 `lotka.m` 定义的常微分方程进行数值求解。

1.3.4 程序设计中应注意的几个问题

MATLAB 的程序设计是很简单的,普通的 M 文件可以作为主程序,在其中可以调用任何 MATLAB 系统提供的函数和用户自己定义的函数 M 文件。为了使用户能够对程序的执行进行控制,MATLAB 提供了几个特殊的函数。

通常在执行一个 M 文件时,MATLAB 不显示 M 文件中的语句,使用 MATLAB 的 `echo` 命令可以改变这种状态。这在进行程序的设计和调试时是很有用的。在 M 文件中可以通过加入 `echo` 命令,使其后的语句在执行时在命令窗口中显示出来,而 `echo off` 命令则是关掉显示状态。

函数 `input` 用于获得用户的键盘输入,例如:

```
>> n = input('How many apples')
```

以字符串'How many apples'为提示符,等待用户的输入,并将用户从键盘上输入的数据或表达式赋给变量 `n`。函数 `input` 的一个用处是设计菜单式程序,详见 MATLAB 的 `demo` 程序中的例子。

另一个比 `input` 更有效的输入函数是 `keyboard`,这个函数使得键盘就像一个普通 M 文件一样,将其置入 M 文件中,对程序或函数的设计和调试很有帮助。

MATLAB 的 `pause` 命令的作用是使程序暂时停下,直至用户按下任何键后再运行。而 `pause(n)` 则是使程序暂停 `n` 秒钟后再继续执行。

MATLAB 的内部向量与矩阵运算的速度远比 MATLAB 的编译解释的速度要快,这意味着,为了加速 MATLAB 的运行速度,在程序设计中要尽可能地使用向量或矩阵化的算法,特别是要将 `for` 和 `while` 等循环语句尽可能地转换成向量或矩阵的运算。例如,为了计算从 1 到 10 之间的 1001 个点的正弦函数值,通常的方法是采用下列语句:

```
>> i = 0;  
>> for t = 0 : .01 : 10  
    i = i+1;  
    y(i) = sin(t);  
end
```

如果按照向量化的算法,相同的结果可以由下列语句组得出:

```
>> t = 0 : .01 : 10;  
>> y = sin(t)
```

在一台速度较慢的 PC386 计算机上,前一组语句所需要的时间大约是 15 秒,而第二组语句仅需要大约 0.6 秒。如何将一段复杂的程序转换成向量或矩阵化的运算是一个复杂的问题,但是在程序设计中要尽可能地使用向量或矩阵化的运算。

另一种加快程序执行速度的方法是为将要使用的向量或矩阵变量预先分配内存空间。例如:

```

>> y = zeros(1, 100);
>> for i = 1:100
    y(i) = i * i;
end

```

在这组语句中,第一条语句首先创建一个有 100 个元素的行向量,各元素的值都为零,然后,for 循环语句对向量 y 的分量重新赋值。如果在这个例子中,取消语句组中的第一条语句 $y = \text{ones}(1, 100)$,那么,MATLAB 将要使用较长的时间来生成向量 y。因为在进入 for 循环体之前,变量 y 不存在,执行第一次循环时, $i = 1$,首先创建变量 $y(1)$,即 1×1 的矩阵(向量)。以后,每执行一次循环,MATLAB 就要重新为向量 y 增加一个新的元素,如果预先定义了向量 y,这个步骤就会省略,从而加快了程序的执行速度。

对于要在内存资源有限的计算机上使用大型数据矩阵的用户来说,预先创建矩阵变量的方法还有第二个优点,即可以更有效地使用内存空间。这是因为在 MATLAB 系统的管理下,内存空间在逐步趋于碎片化,经过一段时间的运行后,虽然还剩下许多内存空间,但是可能没有足够的连续空间用于存放大型的矩阵变量。预先分配内存空间的方法可以减少内存碎片化的可能性。

前面已经介绍过,使用 who 命令可以用来报告所剩的内存空间。如果再从工作环境中清除一个变量,who 命令指出的可用内存数可能没有改变,除非这个变量是工作环境中最大的变量。这是因为指示的内存量是内存中尚未使用的连续内存片的数量,清除最大的变量可以增加这个内存指示量,而清除其他的变量不一定能使这个内存量增加,这意味着可能有更多的内存可以使用。

在编写 MATLAB 程序时,知道一点 MATLAB 的优化技术是很有帮助的。MATLAB 函数在传递输入参数时,如果在函数体内没有语句对该参数重新赋值,那么 MATLAB 就不在函数层的工作环境中生成这个参数的拷贝值。这就意味着只要不在函数内部改变输入参数的值,则在传递一个大型的数据给函数时,就不会存在内存上的障碍。

1.4 矩阵运算与数组运算

MATLAB 可以对矩阵的行、列、单个的元素以及矩阵的子块进行控制和管理。一个数值向量可以用作矩阵的角标,达到访问某些矩阵元素的目的。利用向量和角标的灵活表达方式,可以对矩阵的元素进行有效的访问与管理。

1.4.1 矩阵的创建函数

1. 创建向量对象

冒号“:”在 MATLAB 中是一个很重要的运算符,它可以用来创建向量对象。例如,语句

```
>> x = 1:5
```

表示创建一个行向量对象 x,其元素是从 1 到 5 的五个整数,在 MATLAB 命令窗口中,它的输出是

```
x =
1 2 3 4 5
```

在上述调用形式中,增量为1。也可以用任意大小的数为增量,甚至可以是负数。例如

```
>> y = 0 : pi/4 : pi;
>> z = 6 : -1 : 1
```

生成的结果是

```
y =
0.0000 0.7854 1.5708 2.3562 3.1416
z =
6 5 4 3 2 1
```

冒号运算符使得数据向量的创建变得十分容易。为了得到一个竖型的函数数据表,只需先用冒号运算符生成行向量,再转置,计算出函数值列,就可用两列数据拼成一个函数数据表。例如:

```
>> x = (0.0 : 0.2 : 3.0)';
>> y = exp(-x). * sin(x);
>> [x y]
```

计算的结果是

```
ans =
0.0000 0.0000
0.2000 0.1627
0.4000 0.2610
0.6000 0.3099
0.8000 0.3223
1.0000 0.3096
1.2000 0.2807
1.4000 0.2430
1.6000 0.2018
1.8000 0.1610
2.0000 0.1231
2.2000 0.0896
2.4000 0.0613
2.6000 0.0383
2.8000 0.0204
3.0000 0.0070
```

能够创建向量的其他函数还有:logspace 函数,用于创建对数等距的向量;linspace 函数,用于创建指定长度的等距向量。例如:

```
>> k = linspace(-pi, pi, 4)
```

计算的结果为

```
k =
-3.1416 -1.0472 1.0472 3.1416
```

2. 创建特殊矩阵

MATLAB 提供了一类函数, 用于创建某些特殊类型矩阵, 这些函数主要包括有:

compan	多项式的伴随矩阵
diag	向量的对角矩阵
gallery	检验矩阵
hadamard	Hadamard 矩阵
hankel	Hankel 矩阵
hilb	Hilbert 矩阵
invhilb	逆 Hilbert 矩阵
kron	Kronecker 张量积
magic	幻方矩阵
pascal	杨辉三角形矩阵
toeplitz	Toeplitz 矩阵
vander	Vandermonde 矩阵

例如, 为了求得多项式 $x^3 - 7x + 6$ 的伴随矩阵, 首先, 应根据 MATLAB 中多项式的表达方式, 把多项式表示成一个向量, 其元素为多项式的系数, 按最高次系数到最低次系数的顺序排列, 然后, 将函数 `compan` 作用在这个向量上:

```
>> p = [ 1 0 -7 6 ]
>> A = compan(p)
A =
    0   7  -6
    1   0   0
    0   1   0
```

矩阵 A 的特征值是这个多项式的根, 即

```
>> eig(A)
ans =
    -3.0000
     2.0000
     1.0000
```

又如, 由两个向量决定的 Toeplitz 矩阵可以按下列方式得到:

```
>> c = [ 1 2 3 4 5 ];
>> r = [ 1.5 2.5 3.5 4.5 5.6 ];
>> t = toeplitz(c, r)
t =
    1.000   2.500   3.500   4.500   5.500
    2.000   1.000   2.500   3.500   4.500
```

```

3.000 2.000 1.000 2.500 3.500
4.000 3.000 2.000 1.000 2.500
5.000 4.000 3.000 2.000 1.000

```

另一些创建特殊矩阵的MATLAB 函数有：

<code>zeros</code>	元素全为零的矩阵
<code>ones</code>	元素全为 1 的矩阵
<code>rand</code>	元素为一致分布的随机矩阵
<code>randn</code>	元素为正态分布的随机矩阵
<code>eye</code>	单位矩阵
<code>linspace</code>	线性等距向量
<code>logspace</code>	对数等距向量
<code>meshgrid</code>	用于二元函数, 参见图形功能一章

例如, 为了创建一个 4×3 的随机矩阵, 只需输入下列语句:

```
>> A = rand(4, 3)
```

A =

```

0.2113 0.8096 0.4832
0.0824 0.8474 0.6135
0.7599 0.4524 0.2749
0.0087 0.8075 0.8807

```

1.4.2 矩阵的角标

1. 角标的定义与矩阵子块的表示

矩阵是二维的, 为了标记矩阵的每个元素的位置, 数学上用该元素在矩阵中的行、列标号来表示。MATLAB 也使用类似的方法来表示, 并将行列标号称为角标。例如, $A(3,4)$ 是矩阵 A 的第三行第四列的元素, 这里 3 和 4 被称为矩阵 A 的角标。矩阵的角标可以是表达式, MATLAB 将表达式之值截断成最邻近的整数, 矩阵变量名与角标一起就决定了矩阵的元素及其位置。例如, 给定矩阵 A 如下:

A =

```

1 2 3
4 5 6
7 8 9

```

那么, 下面的语句

```
>> A(3, 3) = A(1, 3)+A(3, 1)
```

将更新矩阵 A 的第三行第三列元素的值, 运算的结果为

A =

```

1 2 3
4 5 6

```

7 8 10

MATLAB 约定矩阵的角标可以是一个向量。如果 x 和 v 都是长度为 n 的向量，那么 $x(v)$ 表示新的向量 $[x(v(1)), x(v(2)), \dots, x(v(n))]$ 。对矩阵来说，可以利用向量角标访问矩阵的子矩阵。例如，设 A 是一个 10×10 的矩阵，那么语句

$\gg A(1:5, 3)$

表示创建一个 5×1 的列向量，由矩阵 A 的第三列前 5 个元素组成，又如语句

$\gg A(1:5, 7:10)$

表示创建一个新的 5×4 阶矩阵，由矩阵 A 的前五行后四列的元素组成，即为 A 的一个子块。在这种表示方法中，单用冒号作角标时，表示该矩阵的全部行或全部列。例如，语句

$\gg A(:, 3)$

表示矩阵 A 的第三列，而

$\gg A(1:5, :)$

表示矩阵 A 的前五行。

矩阵的角标表示可以用在赋值语句中，这是矩阵运算的一种技巧。例如，语句

$\gg A(:, [3 5 10]) = B(:, 1:3)$

表示将矩阵 A 的第三列、第五列和第十列分别用矩阵 B 的前三列代替。

一般地，如果 v 和 w 都是整数元素向量，那么 $A(v, w)$ 表示的是一个取自 A 的矩阵子块，其行由向量 v 确定，而列由向量 w 确定，称 v 为行角标， w 为列角标。这样，

$\gg A(:, n:-1:1)$

表示矩阵 A 的列反序的矩阵。另一种十分有效的表示是 $A(:)$ ，如果 $A(:)$ 在赋值语句的右端，表示由矩阵 A 的元素按列的顺序排成的列向量。例如，语句

$\gg A = [1 2; 3 4; 5 6]$

$\gg b = A(:)$

的输出结果是

$A =$

1	2
3	4
5	6

$b =$

1
2
3
4
5
6

如果 $A(:)$ 出现在赋值语句的左端，表示用一个向量对矩阵 A 进行赋值，此时矩阵 A 必须事先存在。例如，设 A 是上述的 3×2 阶的矩阵，那么语句

$\gg A(:) = 11:16$

行向量(11,12,13,14,15,16)的 6 个元素依照矩阵 A 的列顺序给 A 的元素赋值,但保持矩阵 A 的维数 3×2 ,因此,

```
A =
11    14
12    15
13    16
```

MATLAB 的函数 `reshape` 正是使用这种方法来改变一个矩阵的维数的。一般说来,矩阵的角标出现在语句等号的右端时,表示创建一个新的矩阵对象;在左端时,则表示对原矩阵中的部分或全部元素重新赋值。

2. 矩阵的 0-1 角标

在 MATLAB 中,可以使用 0-1 向量作为矩阵的角标向量。这样就可以用关系运算的结果来引用矩阵的元素或子块,这在一些数学计算中是十分有意义的。假设 A 是一个 $m \times n$ 阶矩阵, L 是一个长度为 m 的 0-1 向量,那么 $A(L, :)$ 表示由向量 L 的非零元素对应的矩阵 A 的行组成的子块。下面的语句可以用来消去向量 x 中大于 0.03 的元素:

```
>> x = x(x<=0.03);
```

类似地,

```
>> L = X(:, 3)>100
>> X = X(L, :)
```

表示用原矩阵 X 的第三列中值大于 100 的元素所在行组成的子块矩阵去替换矩阵 X,即赋值给变量 X。

3. 空矩阵及其应用

MATLAB 定义了一个很特殊的矩阵,即空矩阵。空矩阵可以由下列的语句创建:

```
>> x = []
```

即将一个 0×0 阶的矩阵赋给变量 x,随后即可以应用这个空矩阵变量,而不会导致错误。空矩阵语句的作用是传播空矩阵,它与 MATLAB 的清除变量的命令

```
>> clear x
```

是不相同的, `clear x` 是从工作环境中清除变量 x,清除之后,再引用变量 x 是非法的。而空矩阵变量是一个存在于工作环境中的特殊变量,它的维数为 0×0 ,无任何元素,可以用 `exist` 函数检测该变量的存在性。函数 `isempty` 则可以用来检测某个变量是否为空矩阵。

除了空矩阵外,还有空向量。如果 n 比 1 小,那么向量 $1:n$ 就是无任何元素的空向量。空向量也是空矩阵。

最为重要的一点是,利用空矩阵的特性,可以进行从一个矩阵中消去部分行和部分列的运算。例如,语句

```
>> A(:, [2, 4]) = []
```

的运算结果是从矩阵 A 中消去第 2 列和第 4 列的矩阵子块。

在 MATLAB 中,某些 MATLAB 函数用空矩阵作变量时,被赋予了特定的值。这些函数包括 `det`(矩阵行列式的值)、`cond`(矩阵的条件数)、`prod`(各列元素之积)和 `sum`(各列元素之和)。例如,如果 x 是一个空矩阵,则分别规定 $\det(x)=1$, $\text{prod}(x)=1$ 和 $\text{sum}(x)=0$

等。

1.4.3 矩阵与数组运算

1. 矩阵的算术运算与关系运算

在本书 1.2.3 节中,已经介绍了矩阵的算术运算和关系运算。这里简单介绍大矩阵的构造方法。与数学运算一样,MATLAB 可以由小矩阵作为矩阵子块来构造大矩阵。例如,如果 A 是 n 阶方阵,那么,语句

```
>> C = [ A A'; ones(size(A)) A.^ 2];
```

表示创建一个 $2n$ 阶的方阵。在这样的运算中,要保持矩阵阶数的协调性,否则 MATLAB 系统会报告运算出错。

MATLAB 提供了一个特殊函数 NaN (Not-a-Number 的缩写),它的作用是创建一个对象。该对象表示的不是数学意义上的任何数值对象。通常 NaN 可以由未定义的表达式 $0/0$ 按照 IEEE 浮点运算的标准建立。用户可以用 NaN 表示一些不可使用的数据(Not Available Data)。

正确地掌握不可使用的数据是一个较困难的问题,在不同的场合其意义不同。MATLAB 为此提供了统一管理 NaN 的方法,使它自然地在运算过程中传播。因此,在任何计算过程中,只要使用了 NaN,那么最终的计算结果就是 NaN,除非计算结果不依赖于 NaN 的值。

MATLAB 函数 isnan 用于检验某些变量是否是 NaN。例如,在向量 x 中,其值为 NaN 的元素在向量中的位置是

```
>> ii = find(isnan(x));
```

其中,函数 find 的作用是返回向量 x 中非零元素的角标值。又如,由下列语句

```
>> x = x(find(~isnan(x)))
```

可使返回的数据中不再包含有以 NaN 为值的元素。另外,下列语句可以达到同样的目的:

```
>> x = x(~isnan(x))
```

或

```
>> x(isnan(x)) = [ ];
```

上述的第二种方法是最为直接的,但是,下面的语句却不能达到这样的目的:

```
>> x(x==NaN) = [ ];
```

因为,关系运算 $x == NaN$ 的结果仍是 NaN,所以角标 $x(NaN)$ 是没有意义的。如果要从一个矩阵中去掉含有 NaN 值的行,可以由下列语句来实现:

```
>> x(any(isnan(x)', :)) = [ ];
```

2. 矩阵的管理

为了实现出一个矩阵创建新的矩阵,以及对矩阵的运算进行管理,MATLAB 提供了一些功能函数,主要包括有:

rot90	将矩阵旋转 90 度
fliplr	将矩阵的列反序(左右反序)

flipud	将矩阵的行反序(上下反序)
diag	提取矩阵的对角元素形成对角矩阵或将一个向量转换成对角矩阵
tril	提取矩阵的下三角部分
triu	提取矩阵的上三角部分
reshape	改变矩阵阶数

例如,当要将一个 3×4 阶矩阵按照列的顺序重排成 2×6 的矩阵时,可以用下列的语句来实现:

```
>> A = [1 4 7 10; 2 5 8 11; 3 6 9 12];
>> B = reshape(A, 2, 6)
B =
    1   3   5   7   9   11
    2   4   6   8   10  12
```

又如,A 的下三角矩阵可以由下列语句得到:

```
>> tril(A)
ans =
    1   0   0   0
    2   5   0   0
    3   6   9   0
```

与矩阵运算相关的另外两个 MATLAB 函数是 size 和 length。函数 size(A) 的返回值是一个二元素的向量,第一元素为矩阵 A 的行数,第二元素为矩阵 A 的列数。在 MATLAB 5.0 中,size 的定义已经扩展到任何多维数组,它的返回值是一个维数向量。如果已经知道 v 是一个向量,那么 length(v) 的返回值是该向量的长度,或 max(size(v))。

3. 面向矩阵列的运算

为了方便数据分析,MATLAB 提供了一部分面向矩阵的列运算的数据分析函数,这些函数主要包括有:

max	矩阵各列的最大值向量
min	矩阵各列的最小值向量
mean	矩阵各列的平均值向量
median	矩阵各列的中值向量
std	矩阵各列的标准偏差向量
sum	矩阵各列的元素之和向量
prod	矩阵各列的元素之积向量
cumsum	矩阵各列的累加和向量
cumprod	矩阵各列的累积向量
diff	矩阵各列的差分向量
hist	矩阵各列的直方图向量
cov	列向量卷积
sort	排序

corrcoef 矩阵相关系数

对于向量变量,这些函数会把行向量和列向量都作为数组来处理。而对于矩阵变量,这些函数则面向矩阵的列进行运算,其结果是行向量。例如,函数 max 应用于矩阵变量时,返回的行向量的各元素的值恰为矩阵各列的最大值。用户可以自己编写面向列运算的函数 M 文件,只需加入一些辅助的条件语句判别出输入变量是向量还是矩阵。可以参考 MATLAB 的 M 文件 mean.m 和 diff.m。

在 MATLAB5.0 版中,以上这些函数(除最后一个函数外)都可以应用于多维数组。例如,函数 sum 的一般调用命令为

```
>> sum(M,d)
```

其中,M 是一个 n 维数组,d 可以是任何表达式,其值介于 1 与 n 之间。它的作用是在 n 维数组 M 的第 d 维方向计算 M 的元素之和,其结果应该是一个 n-1 维数组。例如:

```
>> M = zeros(3,3,4);
>> for k = 1:4
    M(:,:,k) = k * ones(3,3);
end
>> sum(M,1)
ans =
    3   6   9   12
    3   6   9   12
    3   6   9   12
>> sum(M,2)
ans =
    3   3   3
    6   6   6
    9   9   9
    12  12  12
>> sum(M,3)
ans =
    10  10  10
    10  10  10
    10  10  10
```

1.4.4 线性代数与稀疏矩阵

MATLAB 通过其矩阵运算,提供了强有力的数学运算功能。有些常用的算法被设计成 MATLAB 的核心程序,运算速度很快。另外一些功能由 MATLAB 提供的外部 M 文件库来实现。本节主要介绍 MATLAB 提供的线性代数功能。

1. 矩阵分解与线性方程求解

首先介绍 MATLAB 提供的矩阵分解算法和线性方程组的求解方法,读者可以参考 MATLAB 的用户手册,了解更详细的内容。

(1) 三角分解

矩阵的三角分解是最基本的一种矩阵分解方法,它将一个矩阵分解成上三角矩阵与下三角矩阵的乘积,称为 LU 分解或 LR 分解。实现 LU 分解的算法大都采用 Gauss 消去法,在 MATLAB 中,实现 LU 分解的函数是 lu,它的返回值是分解后的矩阵因子,调用命令为

```
>> [L, U] = lu(A)
```

其中,U 是上三角矩阵,而 L 可以重新排列成一个下三角矩阵,使其主对角元素全为 1。这些矩阵因子可以用来计算矩阵的逆和矩阵行列式的值。LU 分解也是求解线性方程组的基本算法,另外,矩阵的除法以 LU 分解算法为基础。例如,设有矩阵

```
>> A = [1 2 3; 4 5 6; 7 8 0]
```

为了得到 LU 分解,调用 lu 函数,即

```
>> [L, U] = lu(A)
```

其结果将是

L =

0.1429	1.0000	0.0000
0.5714	0.5000	1.0000
1.0000	0.0000	0.0000

U =

7.0000	8.0000	0.0000
0.0000	0.8571	3.0000
0.0000	0.0000	5.0000

为了检验分解的正确性,计算两个矩阵因子之积的语句如下:

```
>> L * U
```

计算结果为

ans =

1	2	3
4	5	6
7	8	0

而矩阵 A 的逆矩阵为

```
>> X = inv(A)
```

在 MATLAB 中,函数 inv 的算法之一是基于 LU 分解,也即 $\text{inv}(A) = U^{-1} * L^{-1}$ 。

矩阵 A 的行列式为

```
>> d = det(A)
```

d =

27

也可以由 LU 分解的矩阵因子的行列式得到:

```
>> d = det(L) * det(U)
```

```
d =
27.000
```

为什么两种方法的结果以不同的方式显示呢？这是因为当 MATLAB 计算 $\det(A)$ 时，首先检查到 A 的元素全为整数，这样，MATLAB 将计算的结果按纯整数的方式显示出来；而在第二种方式中， U 的元素包含有非整数，这样 MATLAB 就以实数的方式显示整数。

设有列向量 $b = [1 \ 3 \ 5]'$ ，为了求解方程 $Ax = b$ ，利用 MATLAB 的矩阵除法，得到

```
>> x = A\b
x =
0.3333
0.3333
0.0000
```

这个解也可以由 LU 分解得到：

```
>> y = L\b, x = U\y
```

另一个使用三角分解的 MATLAB 函数是 `rcond`，这个函数可以用于估计方阵的 L_1 模条件数倒数之值。还有两个与 LU 分解相关的 MATLAB 函数，即 `chol` 和 `rref`。函数 `chol` 用于获得对称正定矩阵的 Cholesky 分解，而函数 `rref` 用于则得到长方矩阵的约化行梯形矩阵。

(2) 正交分解

正交分解也称为 QR 分解，它可以将一个方阵或长方矩阵表示成一个正交矩阵和一个上三角矩阵的乘积。例如，设

```
>> A = [1 2 3; 4 5 6; 7 8 9; 10 11 12];
```

这个矩阵是秩退化的，中间一列是其他两列的平均。那么，下面的 `qr` 命令

```
>> [Q, R] = qr(A)
```

将计算 A 的 QR 分解，其结果如下：

```
Q =
-0.0776 -0.8331 0.5444 0.0605
-0.3105 -0.4512 -0.7709 0.3251
-0.5433 -0.0694 -0.0913 -0.8317
-0.7762 0.3124 0.3178 0.4461
```

```
R =
-12.8841 -14.5916 -16.2992
0.0000 -1.0413 -2.0826
0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
```

矩阵 R 是上三角形的，主对角线下的元素为零，在这里 $R(3,3)=0$ 表示矩阵 R 或 A 不是满秩的。QR 分解还被应用于方程个数比未知数个数多的方程组的求解中。例如，设

$b = [1 \ 3 \ 5 \ 7]'$ ；

则线性方程组 $Ax=b$ 的未知数个数是 3，而方程的个数是 4。在最小二乘的意义下，该方程

组的最佳近似解为

```
>> x = A\b;
```

而 MATLAB 的输出结果将是

```
Warning: Rank deficient, rank = 2, tol = 1.4593E-014
```

```
x =
```

```
0.5000  
0.0000  
0.1667
```

警告信息只是说明矩阵是秩退化的,而 tol 是用来判别主对角线元素是否为零的阈值。方程组的解是由下列两个步骤得到的:

```
>> y = Q'*b;
```

```
>> x = R\y;
```

如果检查方程组解的正确性,就会发现 $A * x$ 与 b 之间存在有一定的误差。

QR 分解还能为 MATLAB 函数 null 和 orth 提供基本算法。对一个矩阵而言,这两个函数分别给出矩阵变换的零空间和值域空间的正交基底。

(3) 奇异值分解

在这里不打算介绍奇异值分解的理论,只是说明奇异值分解是矩阵分析的一种有效的工具。有兴趣的读者可以参考 Golub 和 Van Loan 的著作《矩阵计算》。在 MATLAB 中,下面的语句

```
>> [U, S, V] = svd(A)
```

计算矩阵奇异值分解中的三个矩阵因子,它们满足条件

```
A = U * S * V'
```

其中,矩阵 U 和 V 是正交矩阵,而 S 是对角矩阵。在函数 svd 的调用式中,如果只有一个输出变量,则该函数返回的仅是矩阵 A 的奇异值。

使用矩阵奇异值分解算法的 MATLAB 函数主要有:矩阵的广义逆 pinv(A)、矩阵的秩 rank(A)、矩阵的 Euclidean 范数 norm(A, 2) 和矩阵的条件数 cond(A)。

(4) 特征值分解

设 A 是一个 $n \times n$ 阶矩阵,满足矩阵方程 $Ax = \lambda x$ 的 λ 称为矩阵 A 的特征值。特征值可以由下列命令求得:

```
>> eig(A)
```

它的返回值是列向量。如果矩阵 A 是一个实对称矩阵,那么所有的特征值都是实数。如果矩阵 A 是非对称的,大多数情况下,特征值是复数。例如,设

```
>> A = [ 0 1
```

```
          -1 0];
```

那么,eig(A)的结果是

```
ans =
```

```
0.0000 + 1.0000i  
0.0000 - 1.0000i
```

而下列形式的调用则可以同时得到特征值和特征向量：

```
>> [X, D] = eig(A)
```

其中，矩阵 D 是由特征值组成的对角矩阵，矩阵 X 的列由对应的特征向量组成，使得方程 $A * X = X * D$ 成立。

如果 A 和 B 都是方阵，那么 $\text{eig}(A, B)$ 的结果是使得方程 $Ax = \lambda Bx$ 成立的广义特征值组成的向量。为了同时得到相应的特征向量，可以用下列的命令形式：

```
>> [X, D] = eig(A, B)
```

其中，D 是由特征值构成的对角矩阵，而矩阵 X 的列是由对应的特征向量组成的，满足方程 $A * X = B * X * D$ 。

2. 矩阵函数与矩阵分析

在矩阵分析方面，MATLAB 也提供了许多函数，如多项式的表示方法、矩阵函数以及矩阵范数和条件数等。

(5) 多项式表示法

MATLAB 将算术多项式表示成一个由多项式系数组成的行向量，其向量元素按照从高次项系数到低次项系数的顺序排列。例如，数学上的多项式

$$a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

就被表示成长度为 $n+1$ 的行向量

$$p = [a_n \ a_{n-1} \ \dots \ a_1 \ a_0]$$

或者说一个行向量在与多项式有关的函数中被解释为多项式。例如，设有矩阵

```
>> A = [1 2 3  
        4 5 6  
        7 8 0]
```

它的特征多项式是

```
>> p = poly(A)
```

```
p =
```

$$1 -6 -72 -27$$

即是多项式 $s^3 - 6s^2 - 72s - 27$ 的系数行向量。该多项式方程的全部根为

```
>> r = roots(p)
```

```
r =
```

$$12.1229$$

$$-5.7345$$

$$-0.3884$$

这些根恰为矩阵 A 的特征值。MATLAB 函数 poly 的另一个作用是由指定的全部根求出该多项式，又如上述例子：

```
>> p2 = poly(r)
```

```
p2 =
```

$$1 -6 -72 -27$$

设有两个多项式 $a(s) = s^2 + 2s + 3$ 和 $b(s) = 4s^2 + 5s + 6$ ，它们的乘积可以由这两个多项

式的系数向量的卷积得到,即

```
>> a = [1 2 3]; b = [4 5 6]
>> c = conv(a,b)
c =
    4    13    28    27    18
```

利用逆卷积可做多项式的除法,如

```
>> [q, r] = deconv(c, a)
q =
    4    5    6
r =
    0    0    0    0    0
```

MATLAB 提供的多项式函数主要包括有:

<code>poly</code>	矩阵的特征多项式
<code>roots</code>	多项式求根
<code>polyval</code>	多项式函数值
<code>polyvalm</code>	矩阵多项式函数值
<code>conv</code>	多项式乘积或向量之卷积
<code>deconv</code>	多项式除法或向量之逆卷积
<code>residue</code>	部分分式展开
<code>polyder</code>	多项式微商
<code>polyfit</code>	多项式曲线拟合

这里特别说明一下函数 `polyval` 和 `polyvalm` 的区别,设 A 是方阵, p 是多项式 $s^3 - 2s + 12$,那么 `polyval(p, A)` 的含义是

$$A \cdot ^3 - 2 * A + 12 * \text{ones}(\text{size}(A))$$

而 `polyvalm(p, A)` 的含义则是

$$A \cdot A \cdot A - 2 * A + 12 * \text{eye}(\text{size}(A))$$

一般说来,`polyval` 可以作用于任何矩阵,而 `polyvalm` 只能作用于方阵。

(6) 矩阵函数

以矩阵为变量的 MATLAB 函数称为矩阵函数,例如,多项式矩阵函数 `polyvalm` 就是矩阵函数,MATLAB 提供的矩阵函数还有如下几种:

<code>expm(A)</code>	矩阵指数函数,定义为 e^A
<code>logm(A)</code>	矩阵对数函数,定义为 <code>expm(A)</code> 的逆矩阵
<code>sqrtm(A)</code>	矩阵方根,定义为 $X^2 = A$ 的根 X

更一般的矩阵函数用 MATLAB 函数 `funm(A, '函数名')` 来定义。例如, `funm(A, 'exp')` = `expm(A)`,但是在 MATLAB 内部,实现它们的算法各不相同。又如,语句

```
>> S = funm(A, 'sin')
>> C = funm(A, 'cos')
```

与下列语句的计算结果是有差别的,只是在误差允许的范围内相差很小,

```
>> E = expm(i * A);
>> C = real(E)
>> S = imag(E)
```

(7) 矩阵范数与条件数

与矩阵的范数和条件数有关的 MATLAB 函数有：

cond	矩阵的 L_2 范数条件数
norm	矩阵的 L_1 范数、 L_2 范数、F 范数和无穷范数
rank	矩阵的秩
rcond	条件数估计

在不同的情况下，MATLAB 按不同的方法计算矩阵的秩。这些方法包括 rref(A)、A\b、orth(A)、null(A) 和 pinv(A) 等。MATLAB 使用三种算法和计算标准。在有些情况下，对同一矩阵可能会得到不同的秩。在 rref(A) 中，矩阵的秩是约化后矩阵中非零行的行数，rref 使用的是三种求秩算法中速度最快的一种，但是使用的算法技巧最少且可靠性最低；A\b、orth(A) 和 null(A) 使用的是 QR 分解；而 pinv(A) 用的则是奇异值分解算法，奇异值分解算法是最费时的一种算法，但是它的可靠性最高，因而这个算法也用在 MATLAB 函数 rank 中。

3. 稀疏矩阵及其表示

矩阵被用于描述数学模型中多个因子之间的线性关系。模型越复杂细致，模型的因子个数就越多，矩阵的阶数也就会越大。描述 n 个因子的模型矩阵往往是 n 阶的，为了求解 n 阶线性方程组，要求大约 $O(n^2)$ 计算机存储空间和 $O(n^3)$ 计算机单位时间。当 n 很大时，即使是在工作站上，求解满矩阵的方程组，也是一件较为困难的事情。

事实上，模型的每个因子只是与很少的其他因素有直接的关系，因而在矩阵中往往具有很多的零元素。这种具有很多零元素的矩阵在数学上习惯地被称为稀疏矩阵。

MATLAB 用两种方式存储矩阵，即满矩阵存储方式和稀疏矩阵存储方式，简称为满矩阵和稀疏矩阵。下文所说的满矩阵或稀疏矩阵即指存储方式，而不是严格的数学意义上的满矩阵或稀疏矩阵。

一个实 $m \times n$ 阶的矩阵要求用长度为 $m \times n$ 的实数数组来存储。每个零元素所需要的存储单位与非零元素所需要的存储单位是同样多的，而稀疏矩阵在 MATLAB 系统内部是用非零元素和其行列指标的数组来表示的。存储是按列来组织的，对每一列元素，非零元素存在一个实数数组中，而对应的行指标存在整数数组中（对复数矩阵，非零元素存在两个实数数组中），另一个整数数组用来指示各列元素的起点，所以如果一个 $m \times n$ 阶实矩阵含有 nnz 个非零值的元素，那么要求有 nnz 个实数单位和 nnz+n 个整数单位的存储空间。注意，存储量与矩阵的行数 m 无直接的关系。

稀疏矩阵不能自动生成，定义在满矩阵上的运算只能生成满矩阵，不论其中有多少元素为零。但是，一旦矩阵按稀疏矩阵存储方式存储，那么稀疏矩阵的存储方式就会在运算过程中传播下去。所以，定义在稀疏矩阵上的运算生成稀疏矩阵，定义在稀疏矩阵和满矩阵上的混合运算生成稀疏矩阵，除非运算明显地破坏稀疏性质。

MATLAB 提供了两种存储方式之间的转换函数，即 full 和 sparse。对任何形式的矩阵

X , $\text{full}(X)$ 将其转换成满矩阵的存储方式。如果 X 本身是满矩阵,则 X 的存储方式不变,如果 X 是稀疏矩阵,那么零元素被加入到矩阵的存储空间中,形成满矩阵的存储方式。相反地, $\text{sparse}(X)$ 则去掉零元素而转为稀疏矩阵的存储方式。

通常,稀疏矩阵用直接的方式创建,而不是用函数 sparse 对满矩阵进行转换。一种最为直接的方法是输入一列非零元素的值和对应的角标的值。函数 sparse 有多种调用形式,最一般的调用形式是

```
>> S = sparse(i, j, s, m, n, nzmax);
```

表示用 $[i, j, s]$ 各行对应的值来生成一个 $m \times n$ 阶的稀疏矩阵,使其恰好只有 nzmax 个非零元素。

二阶差分算子的矩阵表示是典型的稀疏矩阵的例子。该矩阵的主对角线元素都为 -2 ,而上下次对角线的元素都是 1 。可以用多种方法来生成这个稀疏矩阵,最常用的一种方法如下所示:

```
>> D = sparse(1:n, 1:n, -2 * ones(1,n), n, n, n);
>> L = sparse(2:n, 1:n-1, ones(1, n-1), n, n, n-1);
>> S = L + D + L'
```

当 $n=5$ 时,创建的稀疏矩阵是

$S =$

(1,1)	-2
(2,1)	1
(1,2)	1
(2,2)	-2
(3,2)	1
(2,3)	1
(3,3)	-2
(4,3)	1
(3,4)	1
(4,4)	-2
(5,4)	1
(4,5)	1
(5,5)	-2

而 $F = \text{full}(S)$ 的输出是

$F =$

-2	1	0	0	0
1	-2	1	0	0
0	1	-2	1	0
0	0	0	1	-2

函数 sparse 有 6 个输入参数,它也有几种简化的调用方式。如果省略 nzmax 参数,则用 $\text{length}(s)$ 来代替;参数 s, i 与 j 之一可以是数量,在这种情况下,前面的三个参数被扩展为长

度相同的向量。例如,上面的二阶差分算子可以由以下语句更简单地得到:

```
>> D = sparse(1:n, 1:n, -2, n, n);
>> L = sparse(2:n, 1:n-1, 1, n, n);
>> S = L+D+L'
```

更进一步地,在简化的调用形式 $S = \text{sparse}(ii, jj, s)$ 中,对应于一般调用形式时,取 $m = \max(ii)$ 和 $n = \max(jj)$,以及 $\text{nzmax} = \text{length}(s)$ 。

为了更方便地使用对角线元素生成稀疏矩阵,MATLAB 提供了另一个函数 `spdiags`。利用该函数,可以按下列方式创建前面的例子:

```
>> e = ones(n,1);
>> S = spdiags([e -2*e e], [-1 0 1])
```

与 `sparse` 函数相反,函数 `find` 可以将任意矩阵的非零元素及其角标提取出来,所以下列的语句组也可以生成稀疏矩阵:

```
>> [ii, jj, s] = find(A);
>> [m, n] = size(A);
>> S = sparse(ii, jj, s, m, n)
```

如果矩阵的最后一行和最后一列中存在有非零元素,也可以用下列两条语句实现上述运算:

```
>> [ii, jj, s] = find(A);
>> S = sparse(ii, jj, s)
```

对于前面的二阶差分算子的例子,当 $n=5$ 时,

```
>> [ii, jj, s] = find(S);
>> T = [ ii, jj, s ]
```

的输出结果是

$T =$

1	1	-2
2	1	1
1	2	1
2	2	-2
3	2	1
2	3	1
3	3	-2
4	3	1
3	4	1
4	4	-2
5	4	1
4	5	1
5	5	-2

MATLAB 还可以将外部的稀疏矩阵转换成内部形式,例如,如果 ASCII 码文件 `t.dat`

中存有上述的矩阵 T,那么利用语句

```
>> load t.dat
>> S = spconvert(t);
```

可以将矩阵 T 转换成稀疏矩阵。另外,函数 save 和 load 还可以用来处理 MAT 数据文件中的稀疏矩阵。

函数 nnz(A)返回矩阵 A 中非零元素的个数,对满矩阵来说,其值是 sum(sum(A~=0));对于稀疏矩阵,则是通过访问内部结构求得。函数 nzmax(A)用于求矩阵非零元素的存储量,对满矩阵,其值总是 prod(size(A));但对稀疏矩阵,其值是变化的,依赖于运算过程是否在 A 中加入了非零元素。

函数 spy(T)可以用图来表示稀疏矩阵的形态。

除了函数 sparse、speye 和 spdiags 之外,在满矩阵上的运算总是生成满矩阵。而稀疏矩阵的运算,或满矩阵与稀疏矩阵的混合运算,它们的运算结果遵循下列的规则:

- 从矩阵转为数量或固定长度向量的运算,如 size 和 nnz 等总是返回满矩阵;
- 从数量或固定长度向量转换成矩阵的运算,如 zeros、ones、rand 和 eye 等,总是返回满矩阵。rand 和 eye 对应的稀疏矩阵的生成函数分别是 sprand 和 speye,而 ones 则没有对应的稀疏矩阵生成函数;
- 余下的从矩阵到矩阵或向量的单变元 MATLAB 函数,除了 full 和 sparse 以外,返回的矩阵或向量存储类型与其输入参数矩阵的存储类型相同。例如,如果 S 是稀疏矩阵,那么 chol(S) 的结果是稀疏矩阵,diag(S) 是稀疏向量。同样,max(S)、sum(S) 等也是稀疏向量,虽然可能所有的元素全为非零。
- 当两个输入参数都是稀疏矩阵时,两变元的 MATLAB 函数返回稀疏矩阵存储方式;而当变元都是满矩阵时,其结果为满矩阵,除非运算明显地保持稀疏性质。但是,如果 S 是稀疏矩阵, F 是满矩阵,那么 S+F、S * F 和 S \ F 都是满矩阵,而 S * F 和 S & F 则是稀疏矩阵;
- 矩阵的拼接,如 [A B; C D] 被看作是 2 元运算,在混合情况下,生成稀疏矩阵。因此,由满矩阵构成的大矩阵是满矩阵,但只要有一个是稀疏矩阵,其结果就是稀疏矩阵;
- 矩阵的角标运算被看作是一元运算,对稀疏矩阵 S, T=S(i,j) 的结果是稀疏矩阵,其中 i 和 j 是向量或数量。但是,如果在等式的左端,如 T(i,j)=S,即使 S 是稀疏矩阵,也不改变 T 的存储性质。

在 MATLAB 中,有两种方式表示稀疏矩阵的行列重排矩阵。第一种方式与数学表示一样,可以用矩阵左乘一个排列阵,表示矩阵行的重排;右乘一个排列阵的转置,表示矩阵列的重排。在 MATLAB 的角标表示下,如果 p 是向量 1:n 的一个排列,那么 S(p, :) 表示矩阵 S 的行重排,而 S(:, p) 则恰好表示列的重排。例如:

```
>> p = [1 3 4 2 5]
>> I = eye(5, 5);
>> P = I(p, :)
```

```
>> e = ones(4, 1);
>> S = diag(11 : 11 : 55) + diag(2, 1) + diag(e, -1)
```

的输出结果是

P =

1	3	4	2	5
---	---	---	---	---

P =

1	0	0	0	0
0	0	1	0	0
0	0	0	1	0
0	1	0	0	0
0	0	0	0	1

S =

11	1	0	0	0
1	22	1	0	0
0	1	33	1	0
0	0	1	44	1
0	0	0	1	55

则

S(p, :) = P * S =

11	1	0	0	0
0	1	33	1	0
0	0	1	44	1
1	22	1	0	0
0	0	0	1	55

S(:, p) = S * P' =

11	0	0	1	0
1	1	0	22	0
0	33	1	1	0
0	1	44	0	1
0	0	1	0	55

如果 P 采用的是稀疏矩阵的存储方式,那么两种表示方法所需要的存储量与阶数 n 成正比,计算时间与 nnz(S)成正比,角标向量的表达方式较为紧凑有效。

1.5 信号处理

1.5.1 信号处理函数

在信号处理中,向量被用来描述数字信号的取样或信号序列。对多变参数系统来说,矩阵的每一行对应于一组样本点,而各列则对应于信号不同参数的变量。在 MATLAB 的基本环境中,用于信号处理的函数有:

abs	复向量的模向量
angle	复向量的幅角向量
conv	信号的卷积
cov	向量的协方差
deconv	信号的逆卷积
fft	快速 FOURIER 变换
ifft	逆快速 FOURIER 变换

对 2 维数字信号,与上述函数对应的 MATLAB 函数是:

fft2	2 维快速 FOURIER 变换
ifft2	2 维快速 FOURIER 变换
conv2	2 维离散卷积

在 MATLAB5.0 的信号处理工具箱中,增加了一些信号处理函数,有兴趣的读者可以参考 MATLAB 的《信号处理工具箱使用手册》。

1.5.2 数据滤波

设 a, b, x 是向量,函数 $y = \text{filter}(b, a, x)$ 生成的向量是

$$y(n) = b(1)x(n) + b(2)x(n-1) + \dots + b(n_b)x(n-n_b+1) \\ - a(2)y(n-1) - \dots - a(n_a)y(n-n_a+1)$$

向量 y 被称为是由滤波器 a 和 b 对向量 x 进行数据滤波的向量。与此等价的 Z 变换是

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b(1) + b(2)z^{-1} + \dots + b(n_b)z^{-(n_b-1)}}{1 + a(2)z^{-1} + \dots + a(n_a)z^{-(n_a-1)}}$$

其中, n_a 和 n_b 分别为向量 a 和 b 的长度。

例如,为了图示出滤波器对单位脉冲信号的响应,可以由下列语句实现:

```
>> x = [1 zeros(1, n-1)];
>> y = filter(b, a, x);
>> plot(y, 'o')
```

函数 freqz 返回数字滤波器的复频谱响应。频谱响应是函数 $H(z)$ 在复单位圆周 $z = e^{j\omega}$ 上的值。例如,用 freqz 计算 n 点频谱响应:

```

>> [h, w] = freqz(b, a, n);
>> mag = abs(h);
>> phase = angle(h);
>> semilogy(w, mag);
>> plot(w, phase)

```

在信号处理工具箱中,还有很多函数可以用来设计滤波器。

1.5.3 快速 FOURIER(FFT) 算法

FFT 算法在数字信号处理中扮演着重要的角色,广泛应用于离散卷积、频谱响应计算、幂谱估计中。

MATLAB 函数 $\text{fft}(x)$ 可实现对向量 x 的 FFT 计算。如果 x 是一个矩阵, $\text{fft}(x)$ 则是 x 每一列向量的 FFT。

$\text{fft}(x, n)$ 是 n 点 FFT 计算,如果 x 的长度小于 n ,则将 x 用零扩充成长度为 n 的向量;如果 x 的长度大于 n ,则将 x 截断为 n 点的向量,再作 FFT 计算。如果 x 是一个矩阵,则该矩阵的每一列按上述法则调整。

$\text{ifft}(x)$ 是向量 x 的逆 FFT, $\text{ifft}(x, n)$ 是 n 点逆 FFT。

FFT 和逆 FFT 的计算公式分别为

$$X(k) = \sum_{j=1}^N x(j) \omega_N^{(j-1)(k-1)}$$

$$x(j) = \frac{1}{N} \sum_{k=1}^N X(k) \omega_N^{-(j-1)(k-1)}$$

其中, $\omega_N = e^{-2\pi j/N}$ 。

例如,设

```

>> x = [4 3 7 -9 1 0 0 0]';
>> y = fft(x)

```

上述语句的计算结果如下:

```

y =
6.0000
11.4853 - 2.7574i
-2.0000 - 12.0000i
-5.4853 + 11.2426i
18.0000
-5.4853 - 11.2426i
-2.0000 + 12.0000i
11.4853 + 2.7574i

```

一般说来,实向量的 FFT 是复向量。在上述例子中,变换后数据的第一元素为原数据各分量之和,第五元素对应于取样频率(Nyquist 频率),向量 y 的后 3 个元素对应于负频率项,对实向量 x ,它们关于 Nyquist 频率点是复共轭对称的。

第二章 图形功能

MATLAB 的图形系统提供了用图形的方式表达数据的完美技术,这个系统建立在内部图形对象的管理技术上。在 MATLAB 中,线段、曲面、文字等图形元被称为图形对象,它们的外观特性由它们的一类属性值来定义。在 MATLAB 环境中,用户可以通过对这些属性值的访问,达到修改图形对象的外观特性的目的。不过,由于 MATLAB 的图形功能建立在一些高层次的 2 维和 3 维图形函数的操作上,因此,对于大部分的低层属性,用户可以不必关心。

本章主要介绍 MATLAB 高层的 2 维和 3 维图形对象的生成函数以及图形控制函数的使用方法。通过这一章的学习,读者可以掌握 MATLAB 图形系统的大部分功能,足以应付日常数据处理的需要。本章的后面一部分将简单地介绍 MATLAB 图形的句柄(handle)系统,在此基础上,读者可以操作和控制 MATLAB 的各种图形对象。

2.1 平面图形与坐标系

在数学上,一元函数可以用函数图形来表达,在这里自变量对应横坐标轴,函数变量对应于纵坐标轴。与此类似,MATLAB 提供了许多函数,用 2 维的图形来表达数据,本节将尽可能详细地介绍几种 MATLAB 平面绘图函数。

2.1.1 图形窗口与坐标系

1. 图形窗口

在 Windows 环境下,MATLAB 在图形(`figure`)窗口中绘制或输出图形,MATLAB 系统管理着图形窗口的生成、关闭和控制(详见第三章 3.2 节)。图形窗口好比一张绘图纸,MATLAB 函数 `figure` 可以用来生成一个新的图形窗口。在 MATLAB 下,每一个图形窗口有唯一的一个序号 `h`,称为该图形窗口的句柄。在缺省情况下,该序号出现在图形窗口的标题栏中。

MATLAB 通过管理图形窗口句柄来管理图形窗口,在任何时刻,只有唯一的一个窗口是当前的图形窗口,或称为活跃的图形窗口(即 Windows 95 工作台面上所有图形窗口当中最上面的那个图形窗口)。如果不作特别指定,任何绘图函数都是在当前图形窗口中输出图形;任何窗口控制函数都是处理当前的图形窗口。

MATLAB 函数 `figure` 创建一个新的图形窗口时,它的调用形式有两种,分别是:

```
>> figure;
```

```
>> h = figure;
```

它们创建的图形窗口成为当前的图形窗口,第二种形式将新创建的图形窗口的句柄值赋给变量 h。为了使句柄号为 h 的图形窗口成为当前的图形窗口,可采用两种方法。第一种方法是直接用鼠标点按该图形窗口,在 Windows 95 工作台上,该图形窗口就会位于最前方;第二种方法是使用 MATLAB 的函数 figure,其命令形式为

```
>> figure(h)
```

这种方法常常用在程序设计中。在使用绘图函数输出(绘制)图形之前,可以按照上述的方法打开一个当前的图形窗口,将随后的图形输出到特定的图形窗口中。

如果在 Windows 95 工作台上不存在任何图形窗口时,那么任何绘图函数调用都会自动创建一个图形窗口,成为当前的图形窗口,并在其中进行图形输出。

为了取得当前的图形窗口的句柄号,用户可以调用 MATLAB 函数 gcf,这个函数直接返回当前图形窗口的句柄号。

对不再需要的图形窗口,可以按照通常 Windows 关闭窗口的办法,关闭任何一个图形窗口,也可以在命令窗口中用如下的命令

```
>> close(h)
```

关闭句柄号为 h 的图形窗口。这条命令与 figure(h)一样,主要用在程序设计中。

2. 图形窗口内容打印

当前图形窗口的内容可以直接高质量地输出到缺省的打印机上,可以用图形窗口的 FILE 菜单的 print 选项直接打印出来,也可以使用 MATLAB 的打印命令 print 将当前的图形窗口的内容打印出来,或者打印到指定的文件。命令 print 的调用格式为

```
>> print [-设备名选项] [文件名]
```

如果上述两个参数都省略或省略设备名,则选用 printopt.m 文件中设定的打印设备的格式输出,或按该格式输出到指定的文件;如果省略文件名,则按照指定的输出设备格式打印到相应的设备上;如果指定了文件名和打印设备名,则按指定的设备格式输出到指定的文件中。

值得指出的是,如果将图形输出成图像文件,MATLAB 将按照指定设备的点阵方式打印成图像文件,例如 Window 的 BMP 格式、PCX 图像格式、JPG 图像格式,等等。但是,这样输出的图像文件质量不高,所幸的是 MATLAB 支持 PostScript(PS)的打印格式,它支持的 PS 类型主要有:

类型	设备名参数
PostScript	-dps
Color PostScript	-dpsc
Level 2 PostScript	-dps2
Level 2 color PostScript	-dpsc2
Encapsulated PostScript	-deps
Encapsulated color PostScript	-depsc
Encapsulated level 2 PostScript	-deps2
Encapsulated level 2 color PostScript	-depsc2

例如,语句

```
>> print-depsc2 meshdata
```

表示用 Encapsulated level 2 color PostScript 的格式将当前的图形窗口的内容打印到文件 meshdata. eps 中。一般说来,level 2 的 PS 格式文件更小,图形输出的速度也更快。然而,并非所有的 PS 打印机都能支持 level 2 的 PS 格式,所以,使用 level 2 的 PS 格式时,一定要保证所使用的打印机设备可以支持 level 2 的 PS 格式。

3. 坐标系

若将图形窗口比作绘图纸,则仅有绘图纸还不够,还必须有图形(如曲线等)的定位系统,即坐标系。在一个图形窗口中可以有多个坐标系,但与图形窗口一样,总有一个当前的坐标系。每个坐标系都有唯一的标识值,即句柄值。当前的坐标系句柄可以由 MATLAB 函数 gca 取得。与图形窗口句柄值不同,坐标系的句柄值是实数值,一般用一个变量来传递这个实数值。MATLAB 函数 axis 可以用来使某个句柄标识的坐标系成为当前坐标系,调用方式为

```
>> axis(h)
```

其中,h 为指定的坐标系句柄值。MATLAB 的绘图函数实际上是在当前图形窗口的当前坐标系中输出图形。典型的坐标系如图 2-1 所示。

在一个坐标系中,主要有以下几种要素。

(a) 坐标轴的说明文字,分别用命令 xlabel 和 ylabel 产生。例如:

```
>> xlabel('X Labels');
>> ylabel('Y Labels');
```

(b) 坐标系的标题,由命令 title 给出。例如:

```
>> title('The Title of Axis');
```

(c) 坐标系中的图形元,即各种绘图函数生成的诸如曲线、曲面等。

(d) 图形说明文字可以由命令 text 和 gtext 生成。前者按指定的位置在坐标系中写出说明文字,而后者则按照鼠标点按的位置写出说明文字。例如:

```
>> gtext('Descriptions')
```

(e) 坐标网格线,即在坐标系上平行于横轴与纵轴方向的网格线。要画出或擦去网格线,可以分别使用命令 grid on 或 grid off。

函数 axis 除了设定当前的坐标系外,还可以用来对当前坐标系进行控制和操作。对坐标系来说,一个很重要的概念是坐标系的范围向量 [x1, x2, y1, y2],即坐标系的可见范围。横坐标轴是从 x1 到 x2,纵坐标轴是从 y1 到 y2。可以用命令

```
>> axis([newx1, newx2, newy1, newy2])
```

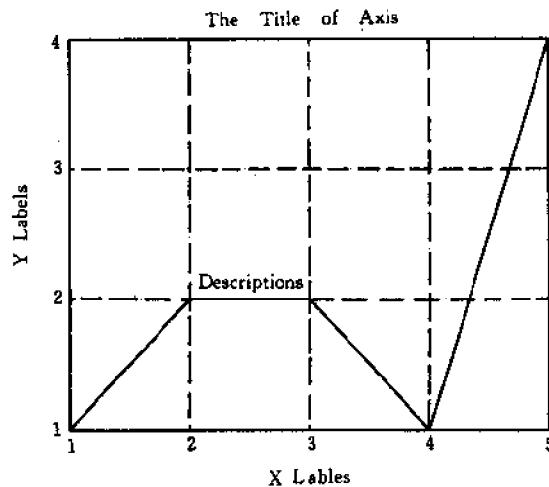


图 2-1 平面坐标系

来重新改变这个向量,以达到改变坐标系比例的目的。而命令

```
>> axis('auto')
```

则将坐标系的范围向量按最大图形元的可见范围重新设置。下列的命令

```
>> v = axis
```

是取得当前的坐标系的范围向量。另外,命令

```
>> axis(axis)
```

的作用是将坐标系当前的比例固定下来。MATLAB 的缺省方式是在绘图时,将所在的坐标系也画出来,为了隐去坐标系,只需使用如下的命令:

```
>> axis off
```

通常,MATLAB 的坐标系是长方形,长宽比例大约是 4:3。为了得到一个方形的坐标系,可以采用下面的命令:

```
>> axis square
```

在显示图像时常用这种方法。由于坐标系的横、纵轴的比例是 MATLAB 自动设置的,比例可能不是一样的。下面的命令被用来得到相同比例的坐标系:

```
>> axis equal
```

另外,坐标系统有两种模式,命令 axis('ij') 将坐标系模式转换到矩阵坐标系模式,此时,坐标系的原点对应于矩阵的左上角,i 轴方向为矩阵的从上到下的垂直方向,j 轴方向为矩阵的从左到右的水平方向,命令 axis('xy') 将坐标系模式转换到缺省的笛卡尔坐标系模式。此时,坐标系的原点为左下角,x 轴方向是水平方向,从左到右;y 轴方向是垂直方向,从下到上。例如,读者可以看一看下列语句的效果:

```
>> plot([1,3,6,4,2])
>> axis('ij')
>> axis('xy')
```

关于坐标系还有一些低层的控制函数,将在 3.3 节中再作详细介绍。

4. 多坐标系统

在缺省的绘图模式下,每个图形窗口中只定义一个坐标系。MATLAB 具有一个图形窗口中定义多个坐标系的功能。最简单的一种实现方法是使用 MATLAB 函数 subplot,其命令形式为

```
>> subplot(m, n, k)
```

其中,m,n,k 为正整数。它的作用是将图形窗口分成 m 行、n 列的 $m \times n$ 块子区域,按行从上到下,从左到右的顺序,在第 k 块子区域中定义一个坐标系,使其成为当前的坐标系,随后的绘图函数将在该坐标系中输出图形。

还有一点值得注意,同一个图形窗口的坐标系可以重叠,这样可以产生前面的坐标系遮住后面的坐标系的各种图形效果。

2.1.2 基本绘图函数

MATLAB 函数 plot 是最简单且用得最为广泛的一个线型绘图函数,利用它可以生成

线段、曲线、参数方程曲线等函数图形。以下将详细介绍 plot 的几种基本命令形式。

1. 向量式 $\text{plot}(v)$

这是最简单的一种调用方式,其中, v 是长度为 n 的数值向量,它的作用是在坐标系中顺序地用直线段连接顶点 $\{(i, v(i)) | i=1, \dots, n\}$,生成一条折(曲)线。坐标系的范围向量由 MATLAB 系统根据向量的长度及其向量元素的大小自动生成(下同)。当向量的元素充分多时,即可以得到一条外观光滑的曲线。例如图 2-1 中的折线就是用下面的语句生成的:

```
>> plot([1, 2, 2, 1, 4])
```

2. 参数式 $\text{plot}(x, y)$

在 $\text{plot}(x, y)$ 中,参数 x 和 y 都是长度为 n 的向量,它的作用是在坐标系中生成顺序连接顶点 $\{(x(i), y(i)) | i=1, \dots, n\}$ 的折(曲)线。这种调用方式可以被用来生成参数方程的图形。例如,为了画出用参数方程

$$\begin{cases} x = \cos(t) \\ y = \sin(t) \end{cases} \quad (0 \leq t \leq 2\pi)$$

表示的单位圆,可以通过下列语句来实现,如图 2-2 所示:

```
>> t = 0 : pi/100 : 2 * pi;
>> x = cos(t);
>> y = sin(t);
>> plot(x, y)
>> axis square
```

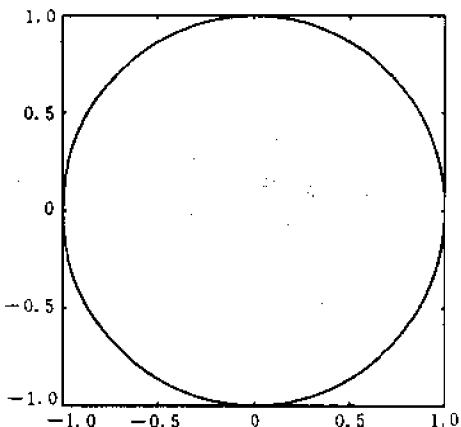


图 2-2 参数方程的图形

上述语句组的最后一条语句是产生一个方形的坐标系,而一般的坐标系的横向与纵向的长度之比大约是 4:3。

3. 矩阵式 $\text{plot}(A)$

在 $\text{plot}(A)$ 中, A 是一个 $m \times n$ 的矩阵,它的作用是以矩阵的行号向量为横坐标值,即 $1:m$,对应于矩阵 A 的每一列向量按第一种方式画出折(曲)线,共 n 条。下面来看一个例子。在 MATLAB 的演示程序中,有一个函数 peaks 可以用来生成数值矩阵,矩阵元素由函数

$$f(x, y) = 3(1 - x^2)e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$

在方形区域 $[-3, 3] \times [-3, 3]$ 的等分网格点上的函数值确定。例如:

```
>> M = peaks(20);
```

将生成一个 20×20 阶矩阵 M ,即分别沿 x 和 y 方向将区间 $[-3, 3]$ 等分成 19 份,并计算这些网格点上的函数值。缺省的等分数是 48。这样,

```
>> plot(peaks)
```

将在同一个坐标系中生成 49 条函数曲线,如图 2-3 所示。该图是函数 $z=f(x, y)$ 的图像从 x 轴方向看去的视图。

4. 混合式 $\text{plot}(X, Y)$

在 $\text{plot}(X, Y)$ 中, 如果 X 和 Y 都是向量, 则长度必须相等, 亦即参数式; 如果 X 是向量, 而 Y 是一个矩阵, X 的长度与矩阵 Y 的行数或列数相等, 则它的作用是将向量 X 与矩阵 Y 的每列或每行的向量相对应作折(曲)线。当 Y 是方阵时, 则将向量 X 与矩阵 Y 的列向量相对应作图; 如果 X 是矩阵, Y 是向量, Y 的长度等于 X 的行数或列数, 则将 X 的每列或每行的向量与 Y 相对应作图。同样, 当 X 是方阵时, 则将 X 的各列与 Y 相对应作图; 如果 X 和 Y 都是矩阵, 且维数相同, 那么按列与列的对应方式来作图。下列语句生成的图形如图 2-4 所示:

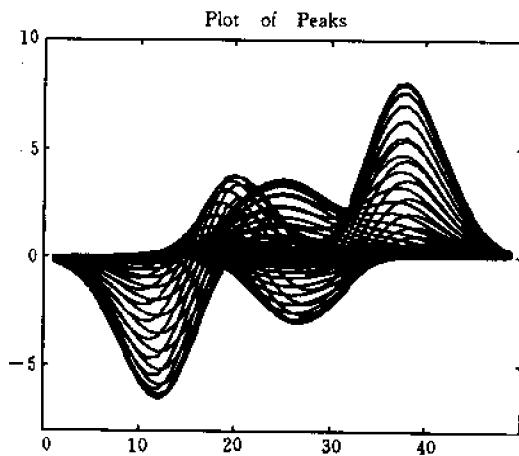


图 2-3 向量对矩阵的图形

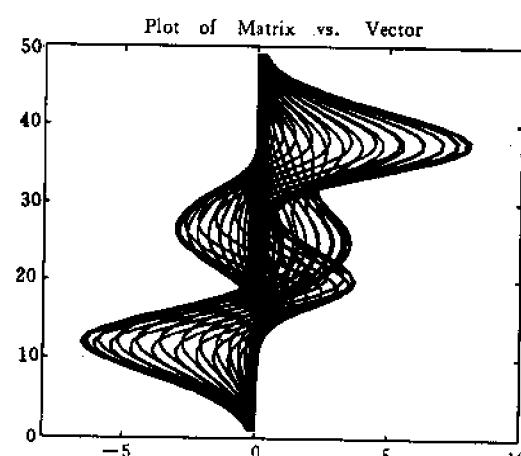


图 2-4 矩阵对向量的图形

```
>> y = 1 : 49;
```

```
>> plot(peaks, y)
```

而

```
>> plot(peaks, rot90(peaks'))
```

生成的图形如图 2-5 所示。其中, $\text{rot90}(A)$ 是将矩阵 A 按逆时针方向旋转 90 度。

5. 复向量式 $\text{plot}(Z)$

在 $\text{plot}(Z)$ 中, 变量 Z 是复向量或复矩阵, 具有非零的虚部。在 MATLAB 中, 这种调用方式是如下调用方式的简化形式:

```
>> plot(real(Z), imag(Z))
```

除此之外, 其他带复数的调用都将忽略复数的虚部。下面的例子展示了一个随机矩阵的特征值的分布, 如图 2-6 所示:

```
>> plot(eig(randn(20,20)), 'x')
```

一般地, 函数 plot 有多个矩阵对的调用方式, 即

```
>> plot(X1, Y1, X2, Y2, ...)
```

这时, 每一对矩阵 (X_i, Y_i) 均按前面的 4 种调用方式之一进行解释, 而不同的矩阵对之间, 其维数可以不同。

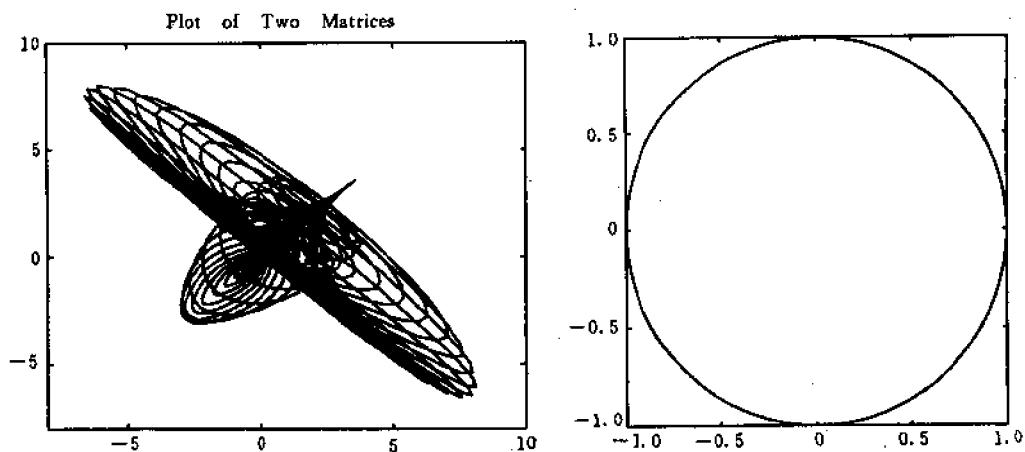


图 2-5 矩阵对矩阵的图形

图 2-6 特征值分布

2.1.3 线型、顶点标记和颜色

前面介绍了 plot 函数的 5 种调用方式。使用这 5 种基本的调用方式时, MATLAB 自动地安排作图的线型和线段的颜色,包括线段顶点的标记。事实上, MATLAB 的 plot 函数还可以设置和管理曲线的线段类型、顶点标记和线段颜色(注:在 MATLAB 的术语中,线型和顶点标记统称为线型,为了方便,本书中将此分开)。

MATLAB 定义的线段类型、顶点标记和线段颜色及其对应的表示字符如表 2-1 所列。

表 2-1 线型、颜色与顶点标记

线型		颜色		顶点标记	
类型	符号	类型	符号	类型	符号
实线	-	黄色	y	实点标记(.)	.
点线	:	洋红色	m	圆圈标记(。)	。
点虚线	-.	蛋青色	c	加号标记(+)	+
虚线	--	红色	r	乘号标记(X)	×
无线	none	绿色	g	星号标记(*)	*
		蓝色	b	方块标记(□)	square
		白色	w	钻石形标记(◇)	diamond
		黑色	k	三角形标记(△)	^
				三角形标记(▽)	v
				三角形标记(▷)	>
				三角形标记(◁)	<
				五角星标记(★)	pentagram
				六角星标记(★)	hexagram
				空标记	none

在表 2-1 的顶点标记栏中,后面的 9 种类型是 MATLAB 5.0 新定义的。

在 plot 函数中,可以传递一个类型参数,类型参数是字符串,由表 2-1 中列出的符号组成。这样可以控制线段类型、顶点标记和线段颜色。plot 的最典型调用方式是所谓的三元组参数,即

```
>> plot(X, Y, S)
```

其中,X 为横坐标矩阵,Y 为纵坐标矩阵,S 为类型说明字符串参数。例如,plot(x, y, 'c+') 在所在顶点处($x(i), y(i)$)标上蛋青色的加号标记。要注意的是,S 字符串可以是 3 种类型的符号之一,也可以是线型与颜色或顶点标记与颜色的组合。但是,3 种类型的符号不能同时出现在 S 中。正如 MATLAB 所区分的那样,顶点标记是一种特殊线型。只在顶点处画标记,没有顶点间的连接线。例如,设 x 和 y 是长度相同的向量,要画出一条红色的虚线,而且在顶点上用黄色的圆圈标记,则可用下面的语句来实现:

```
>> plot(x, y, 'r--o', x, y, 'oy')
```

这个例子表明 plot 函数可以有如下更一般的调用形式:

```
>> plot(X1, Y1, S1, X2, Y2, S2, ...)
```

在作线型图形时,如果不指定作图的颜色,每次使用线型绘图函数 plot 等,MATLAB 将自动循环顺序地使用 y, m, c, r, g, b, w 这 7 种颜色画线。由于 MATLAB 图形窗口缺省的背景颜色是黑色,所以黑色的线段将不可见。这样,黑色可以用来隐藏掉不想见到的线段。在只作一条曲线时,MATLAB 的缺省颜色是黄色。另外,顶点标记可以被放大或缩小,详见 3.4 节线段对象的 markersize 属性。

通常,在当前坐标系中绘图时,每调用一次绘图函数,如调用 plot 时,MATLAB 将擦掉坐标系中已有的图形对象。为了在一个坐标系中增加新的图形对象,可以用 MATLAB 的 hold 命令达到这个目的。在设置 hold on 后,MATLAB 在生成新的图形时保留当前坐标系中已存在的图形对象。此时,MATLAB 根据新图形的大小,可能会重新改变坐标系的比例。例如,下列语句

```
>> t = 0 : pi/100 : 2 * pi;
>> y1 = sin(t);
>> y2 = sin(t + .25);
>> y3 = sin(t + .5);
>> plot(t, y1)
>> hold on
>> plot(t, y2, '--')
>> plot(t, y3, '-.')
>> hold off
```

将在同一个坐标系中画出 3 条曲线,如图 2-7 所示。当然,也可以用下面的语句来达到同样的目的:

```
>> plot(t, y1, 'y', t, y2, '---', t,
y3, '-.')
```

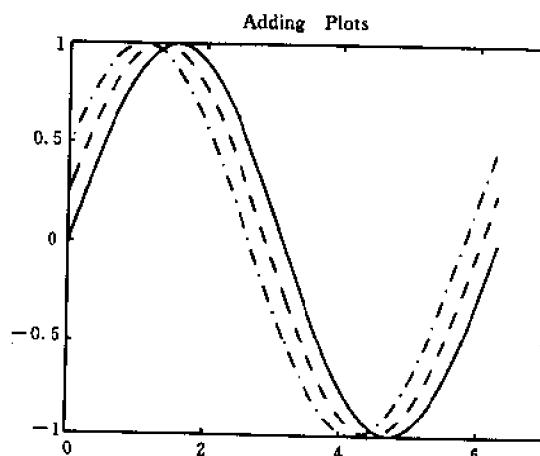


图 2-7 加入新图形

2.1.4 其他 2 维绘图函数

1. 函数简介

为了满足实际应用的需要, MATLAB 还提供了一些绘图函数。下面分别简单地介绍它们的功能:

bar	生成数据的 bar 图形
compass	生成复数的平面向量图形, 向量的起点为坐标原点, 方向由复数的幅角决定, 长度是复数的模长
errorbar	生成误差的 bar 图形
feather	生成复数向量的平面向量图形, 每个向量的方向和长度由对应的复数向量元素决定, 而起点等距分布在水平轴上
hist	生成向量的统计直方图
polar	生成极坐标上的函数图形
quiver	生成向量的梯度场或向量场
rose	生成幅角的统计直方图
stairs	与 bar 的作用相同, 但无区间间隔线段
fill	生成多边形区域并进行着色填充
fplot	生成数学函数的函数图形

2. 多边形填充

MATLAB 函数 fill 的作用是定义一个多边形, 其内部区域根据缺省的或指定的颜色进行着色填充。如果用户已指定多边形各个顶点的颜色, 那么, MATLAB 将会利用这些颜色值进行线性插值, 对内部区域着色。由向量定义的多边形可以是凹的多边形, 也可以自相交。例如:

```
>> t = 0 : 0.05 : 2 * pi;
>> x = sin(t);
>> fill(x, t, 'b')
```

将用蓝色填充正弦曲线与纵轴围成的区域, 如图 2-8(a)所示。

为了得到插值填充的效果, 可以对各顶点着不同的颜色, 如

```
>> colormap(hot);
>> fill(x, t, x)
```

其中, colormap 函数是设置图形窗口的色谱, 读者可以从图形窗口中看到上述语句的填充效果。其填充阴影按 $\sin(t)$ 值的变化而变化, 最小函数值对应于最暗的区域, 而最大函数值则对应于最亮的区域, 如图 2-8(b)所示。

3. 极坐标图形

利用 MATLAB 的函数 polar 可以作极坐标下的函数图形, 这个函数的调用形式是

```
>> polar(theta, rho)
```

其中, theta 是幅角变量, rho 是径向函数变量。一个简单的例子如下:

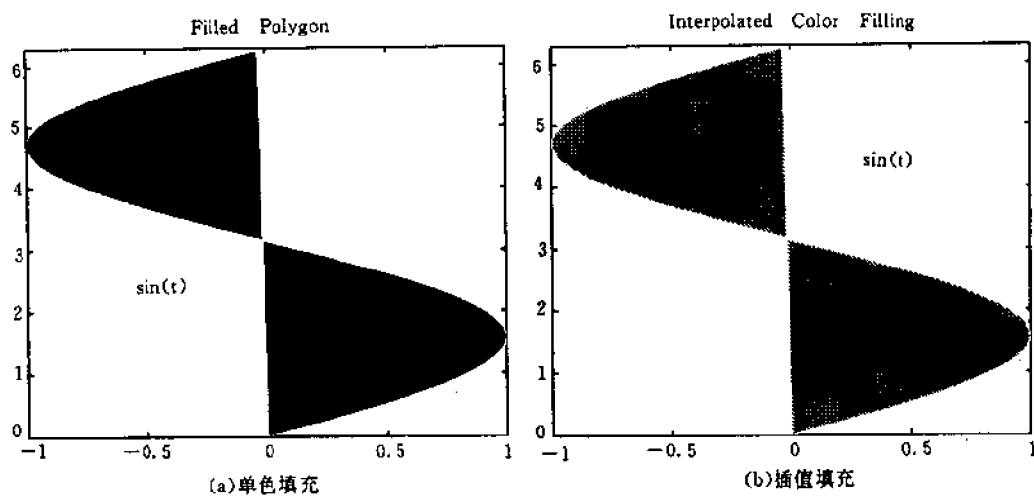


图 2-8 多边形填充

```

>> t = 0 : .01 : 2 * pi;
>> polar(t, sin(2 * t) .* cos(2 * t))

```

上述两条语句生成的图形如图 2-9 所示。

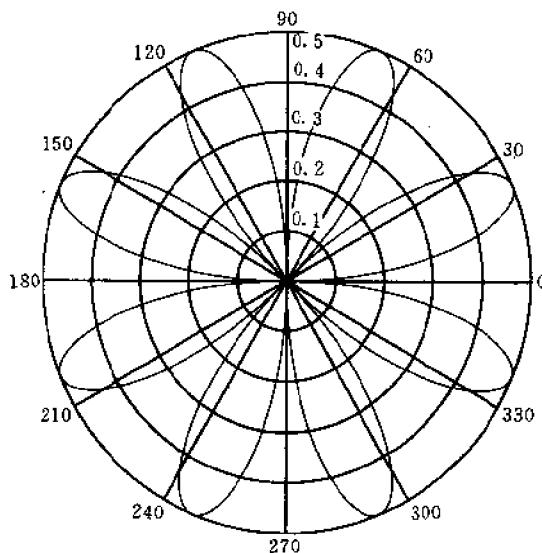


图 2-9 极坐标图形

4. 数学函数图形

在 MATLAB 中, 可以用不同的方法得到数学函数的图形, 最简单的一种方法是在函数的定义域上, 生成一个足够稠密的自变量离散点向量, 这时, 只要计算出函数在该向量上的函数值, 然后利用 plot 函数, 即可作出函数图形。例如, 在区间 $[0, 1]$ 上, 作函数

$$y = \cos(\tan(\pi x))$$

的图像：

```
>> x = (0 : 1/2000 : 1)';
>> plot(x, cos(tan(pi*x)))
```

这两条语句生成的图形如图 2-10(a) 所示。

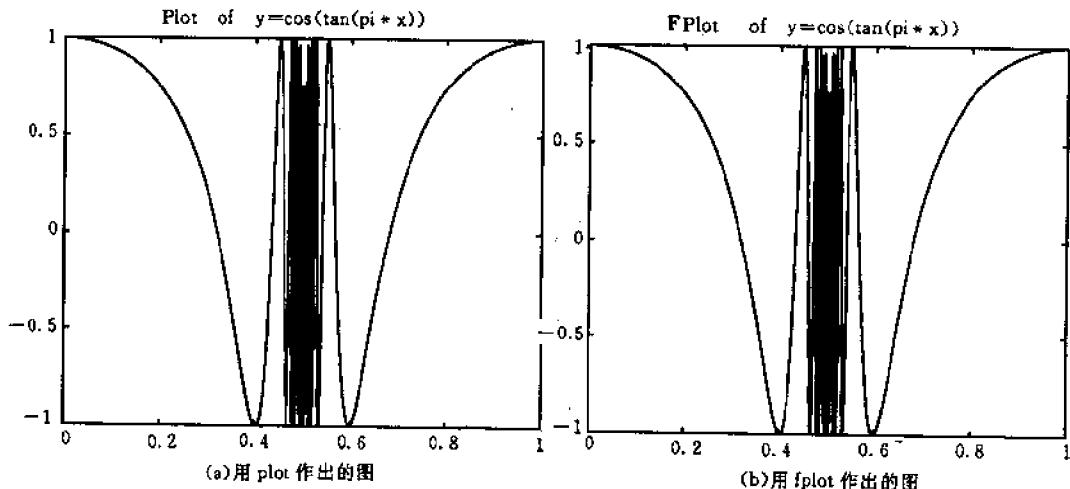


图 2-10 函数的图形

MATLAB 还提供了另一种更有效的函数图形生成方法，即利用函数 fplot 来生成图形的方法。它可以自动或根据用户的指定来安排函数定义区间上的离散点，用 fplot 生成图形比前面的方法的效率要高。在前一种方法中，自变量的离散点在定义区间上是等距分布的，而 fplot 可根据函数值的变化强弱来安排自变量的离散点。函数值变化剧烈的地方，离散点较稠密；反之，则较稀疏。

函数 fplot 接受函数名作为第一个输入变量，所以在使用 fplot 之前，要定义好用于计算函数值的函数 M 文件。对前面的例子，可以定义 fofx.m 如下：

```
function y = fofx(x)
y = cos(tan(pi*x))
```

由于这个函数在定义区间上无穷次振荡，在振荡激烈的地方，MATLAB 可能会用很大的迭代次数计算较多的自变量离散点。为了节省时间，可以特别指定迭代的次数，例如：

```
>> fplot('fofx', [0, 1], 25, 20, 10)
```

在这个例子中，用较少的函数计算量，可得到较为精确的函数图像，如图 2-10(b) 所示。

2.2 3维图形

MATLAB 具有强大的 3 维图形功能，包括 3 维数据显示、空间曲线、曲面、分块及填充，以及曲面光顺着色、视点变换、旋转、隐藏等功能和操作。本节将分别对上述功能予以详细的介绍。

2.2.1 3维图形函数简介

MATLAB 的基本系统中主要包括下列几个 3 维图形函数：

plot3	这是与平面线型绘图函数 plot 对应的 3 维绘图函数
contour, contour3	生成矩阵数据图形或函数曲面的等高线图形
pcolor	根据矩阵的数值对平面的小矩形区域着伪色, 可以用来标识矩阵元素的大小分布
image	用于在平面坐标系上显示一幅图像, 该图像的像素的灰度值或颜色索引号由该函数的输入矩阵变量决定, 该矩阵称为颜色矩阵。一般使用当前的图形窗口色谱对图像像素着色
mesh, meshc, meshz	生成 3 维网格曲面
surf, surfc, surfz	与上述的网格曲面生成函数相对应, 其差别在于这些函数还将对网格曲面进行着色处理, 得到视觉效果更好的彩色曲面
fill3	生成 3 维的多边形区域并按一定的方式填充区域内部
view	设置 3 维坐标系的视点

2.2.2 3维线型图形

函数 plot3 是函数 plot 在 3 维时的调用方式, 用来生成 3 维空间的折线或曲线, 它主要有以下几种调用方式。

1. 向量式 plot3(x, y, z)

这是最基本的调用形式, 其中 x、y 和 z 都是长度相同的向量, 它的功能是在 3 维坐标系中, 生成顺序连接顶点 $\{(x(i), y(i), z(i))\}$ 的折(曲)线。例如, 语句

```
>> t = 0 : pi/50 : 10 * pi;
>> plot3(sin(t), cos(t), t);
```

生成 3 维坐标系中的螺旋线, 如图 2-11 所示。

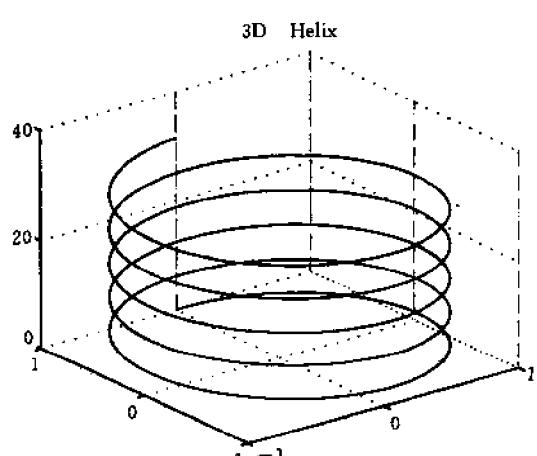


图 2-11 3 维螺旋线型图形

2. 矩阵式 plot3(X, Y, Z)

在这种调用方式中, X、Y、Z 都是维数相同的矩阵, 它按 3 个矩阵的列向量对应地生成多条折(曲)线。每条折(曲)线的颜色由 MATLAB 自动设置, 与 plot 函数一样。另外, 变量 X、Y、Z 中可以有一个或两个是向量, 这时, 向量的长度必须等于矩阵变量的行数或列数。MATLAB 按长度对应的方式选择矩阵的行或列向量来生成折(曲)线。

3. 标准式 plot3(X, Y, Z, s)

与 plot 相类似, 用户可以设置绘制折(曲)线的线型、顶点标记和颜色。plot3(X, Y, Z, s) 中的 s 就是上节中介绍的线型与颜色或顶点标记与颜色的字符串变量, 这是比较一般的调用方式。更为一般的调用形式即所谓的 4 元组形式:

```
>> plot3(X1, Y1, Z1, S1, X2, Y2, Z2, S2, ...)
```

其中, X_i, Y_i, Z_i 可以是矩阵或向量, 而 S_i 是类型字符串变量。每一组 (X_i, Y_i, Z_i) 必须满足维数的要求, 而不同的组之间矩阵的维数可以不同。在不至混淆的情况下, S_i 可以被省略, 这时, 线型类型由 MATLAB 按缺省的方式自动设定。

2.2.3 3维曲面

1. 平面网格点的生成

在数学上, 函数 $z=f(x, y)$ 的图形是 3 维空间的曲面, MATLAB 是如何实现 3 维函数曲面的绘制的呢? 在 MATLAB 中, 总是假设函数 $z=f(x, y)$ 是定义在一个矩形的区域 $D=[x_0, x_m] \times [y_0, y_n]$ 上的。为了绘制在区域 D 上的 3 维曲面, MATLAB 的方法是首先将 $[x_0, x_m]$ 在 x 方向分成 m 份, 将 $[y_0, y_n]$ 在 y 方向分成 n 份, 由各分划点分别作平行于坐标轴的直线, 将区域 D 分成 $m \times n$ 个小矩形块, 计算出在网格点的函数值。对于每个小矩形, 在空间中决定出 4 个顶点 $(x_i, y_j, f(x_i, y_j))$, 连接 4 个顶点得到一个空间中的四边形片。所有这些四边形片一起构成函数 $z=f(x, y)$ 定义在区域 D 上的空间网格曲面。

为方便起见, MATLAB 用函数 meshgrid 来生成 x-y 平面上的小矩形顶点坐标值的矩阵。函数 meshgrid 的调用形式是

```
>> [X, Y] = meshgrid(x, y)
```

或

```
>> [X, Y] = meshgrid(x)
```

第二种形式等价于 $\text{meshgrid}(x, x)$ 。这里, x 是区间 $[x_0, x_m]$ 上分划点组成的向量, 而 y 是区间 $[y_0, y_n]$ 上分划点组成的向量。输出变量 X 和 Y 都是矩阵, 矩阵 X 的行向量都是向量 x , Y 的列向量都是向量 y 。于是, X 和 Y 的元素组 $(X(i, j), Y(i, j))$ 恰好是区域 D 的第 (i, j) 网格顶点。例如, $(X(1, 1), Y(1, 1))$ 对应于 (x_0, y_0) , 而 $(X(m+1, n+1), Y(m+1, n+1))$ 对应于 (x_m, y_n) 。换句话说, 函数 meshgrid 将由两个向量决定的区域转换成对应的网格点矩阵。

下一步的工作就是计算出所有网格点处的函数值。由于矩阵 X 和 Y 的对应元素恰好组成某个网格点, 因此, 利用 MATLAB 的矩阵运算能力, 可以很容易地求出所有网格点上的函数值组成的矩阵。下面通过例子来说明其具体做法。

考虑数学函数 $z=\sin(\sqrt{x^2+y^2})/\sqrt{x^2+y^2}$, 函数定义在区域 $[-8, 8] \times [-8, 8]$ 上。在生成网格点后, 为了计算网格点上的函数值, 只需按下列方法进行:

```
>> x = -8 : 0.5 : 8;
>> y = x;
>> [X, Y] = meshgrid(x, y);
>> R = sqrt(X.^2 + Y.^2) + eps;
```

```
>> Z = sin(R)./R;
```

由于在邻近原点处, R 的某些元素可能会很小, 加入 eps 是避免出现零除数。

2. 网格曲面

在得到网格点上的函数值矩阵后, 即可以用 MATLAB 函数 mesh 来生成函数的网格曲面, 即各网格线段组成的曲面。例如对上述的例子, 下列语句生成的图形如图 2-12 所示:

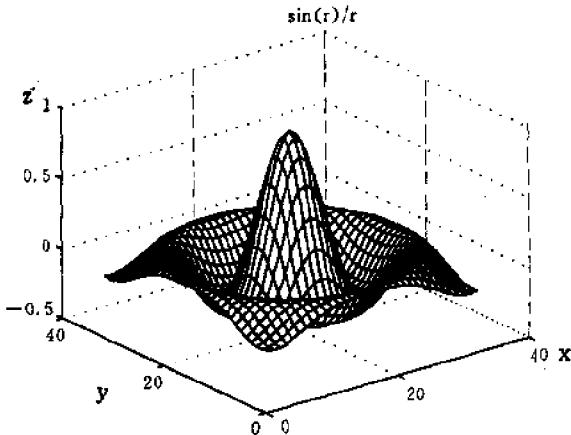


图 2-12 网格曲面

```
>> mesh(Z)
```

函数 mesh 有多种调用形式, 在图 2-12 中, x 轴和 y 轴是按矩阵 Z 的行列序号定义的。为了使其与该函数的定义域一致, 需要使用其他的调用方式。下面分别给予介绍。

- (a) `mesh(X,Y,Z,C)`, 这是最一般的调用形式, X、Y、Z、C 是同维数的矩阵, C 称为颜色矩阵。网格曲面的顶点对应于空间的顶点 $(X(i,j), Y(i,j), Z(i,j))$, 而网格曲面的网格线的颜色由 C 值根据当前的色谱来着色。这种调用形式还可以用来生成参数曲面片;
- (b) `mesh(X,Y,Z)`, 这是在调用形式(a)中, 取 $C=Z$ 的简单调用形式;
- (c) `mesh(x,y,Z,C)`, 其中, x 和 y 是向量, Z 和 C 是同维数的矩阵, 且向量 x 的长度等于矩阵 Z 的列数, 而向量 y 的长度等于矩阵 Z 的行数。在这种情况下, 网格曲面的网格顶点是 $(x(j), y(i), Z(i,j))$, 网格线的颜色由矩阵 C 决定;
- (d) `mesh(x,y,Z)`, 这是在调用形式(c)中, 取 $C=Z$ 的简单调用形式;
- (e) `mesh(Z,C)`, Z 和 C 都是 $m \times n$ 的矩阵, 该形式与 `mesh(x, y, Z, C)` 等价, 其中向量 $x=1:n$, 向量 $y=1:m$;
- (f) `mesh(Z)`, 这是在调用形式(e)中, 取 $C=Z$ 的简单调用形式。

与 mesh 相关的另外两个函数是 meshc 和 meshz, 它们的调用形式与 mesh 相同, meshc 除了生成网格曲面外, 还在 x-y 平面上生成曲面的等高线图形, 如图 2-13(a)所示。而函数 meshz 的作用除了生成与 mesh 相同的网格曲面之外, 还在曲面下面加上一个长方体的台柱, 使图形更加美观, 如图 2-13(b)所示。下面的语句

```
>> nk = 5, n = 2^n k-1;
>> theta = pi * [-n : 2 : n]/n;
```

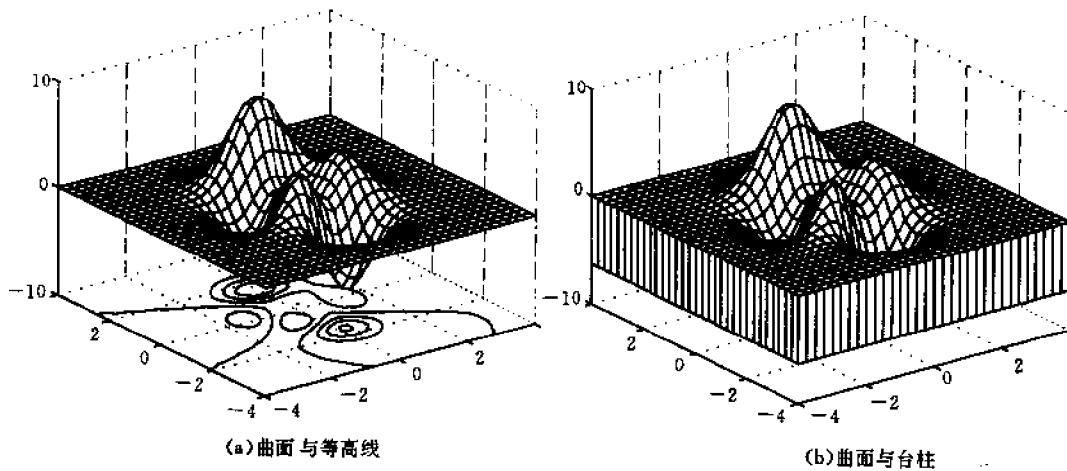


图 2-13 meshc 与 meshz 生成的图形

```

>> phi = (pi/2) * [-n : 2 : n]'/n;
>> X = cos(phi) * cos(theta);
>> Y = cos(phi) * ones(size(theta));
>> C = hadamard(2^nk);
>> mesh(X,Y,Z,C);

```

所给出的图形如图 2-14 所示。

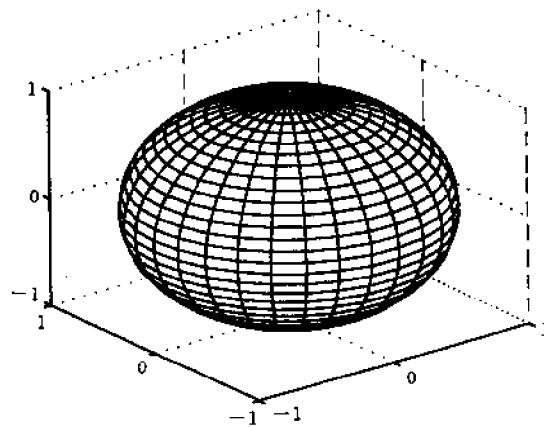


图 2-14 参数曲面

3. 实曲面的绘制

实曲面就是对网格曲面的网格块(四边形片)区域进行着色的结果。这对读者来说很容易理解。MATLAB 函数 surf 可提供这种功能,它的调用方式与 mesh 完全一样,这里不再重复。下面主要就着色机制作些说明。surf 的曲面生成过程与 mesh 是类似的,所不同的是 mesh 仅对网格线进行着色,surf 是对网格片进行着色,而网格线用黑色标出(可以修改)。通常,surf 用缺省的着色方式对曲面片着色(详细的着色原理在第 2.3 节介绍)。除此之外,还

可以用 MATLAB 函数 shading 来改变着色方式。例如：

```
>> shading faceted
```

这是缺省的着色模式，网格线为黑色。而

```
>> shading flat
```

与 faceted 模式类似，只是网格线也分块着色。又如：

```
>> shading interp
```

其着色效果如图 2-15 所示。其中图 2-15(a)是网格曲面，不难看出，图 2-15(d)的着色光顺性最好。

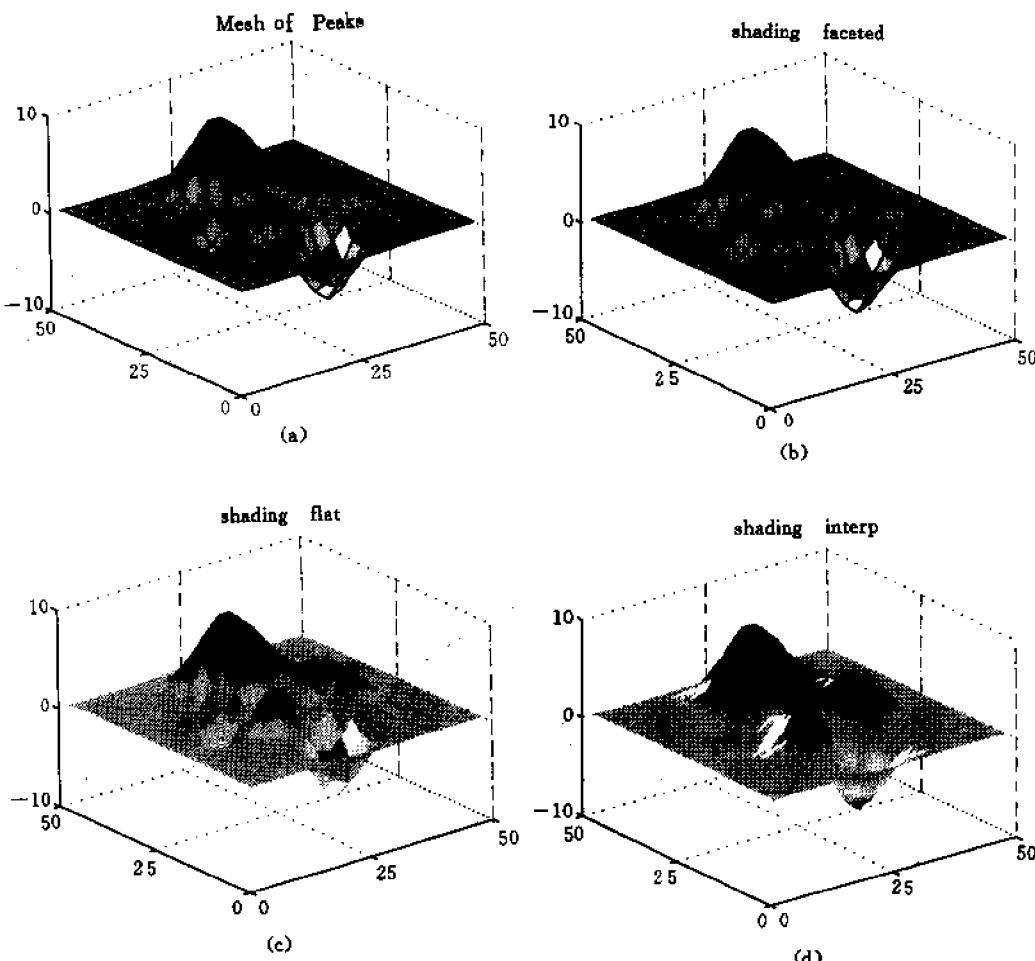


图 2-15 着色的曲面

网格块区域内部像素的颜色由该片的 4 个顶点的颜色值作双线性插值得出。另一个很重要的函数是 surfl，它能生成具有光照效应的曲面，使图形更加美观。除了与 surf 有相同的输入参数外，还可以指定 3 维坐标系的光源点坐标 [x1, y1, z1]，或光源方向的球面坐标值，即经度和纬度(仰角)向量 [azimuth, elevation]。

例如，设光源的方向为经度 = -10 度，纬度 = 50 度，那么下面的语句

```
>> surfl(peaks(200), [-10, 50]);
>> colormap(gray);
>> shading flat
```

生成的图形如图 2-16 所示。

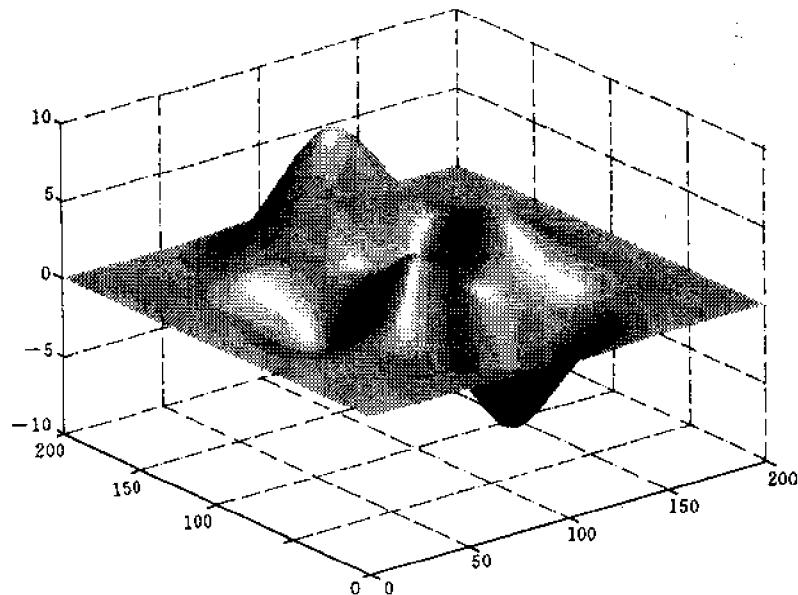


图 2-16 光照效应

还有两个与 mesh 和 meshz 相对应的函数 surf 和 surfz, 这里不再详述。

2.2.4 等高线图形

MATLAB 支持 2 维和 3 维的等高线图形, 函数 contour 和 contour3 被用来实现这种功能。它们将输入的矩阵变量 M 看作是定义在 $[1, m] \times [1, n]$ 上的函数, 生成若干条常数值的等值线段。用户也可以指定等高线的条数, 可以指定坐标系的比例, 以及指定某值上的等高线。例如, 下面的语句生成的等高线图形有 20 条等高线段, 这里用 peaks.m 文件生成输入数据:

```
>> contour(peaks, 20)
```

如图 2-17 所示。

等高线的线型、顶点标记和颜色类型类似于 plot 函数的线型定义。contour3 的功能是生成 3 维的等高线, 调用方式与 contour 是一样的。例如, 最简单地, 利用下面的语句

```
>> contour3(peaks, 20)
```

则生成的 3 维等高线如图 2-18 所示。

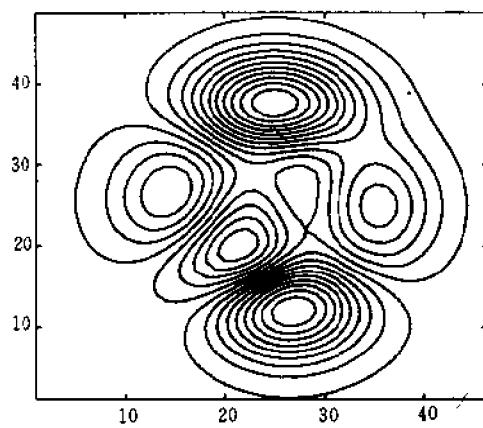


图 2-17 2 维等高线图形

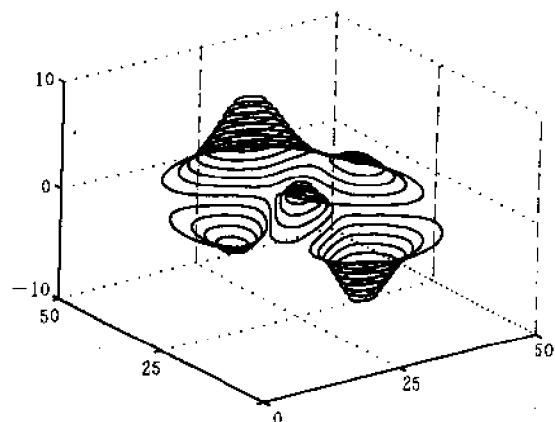


图 2-18 3 维等高线图形

2.2.5 3 维坐标系及图形元的控制

本小节将介绍如何对前面的图形函数生成的图形元进行控制操作。首先，3 维坐标系与 2 维坐标系一样，可以用函数 `axis` 来改变坐标系的形态。与 2 维坐标系的一个差别是，3 维坐标系的范围向量的长度是 6，即 $[x_1, x_2, y_1, y_2, z_1, z_2]$ 。

1. 3 维图形的视点

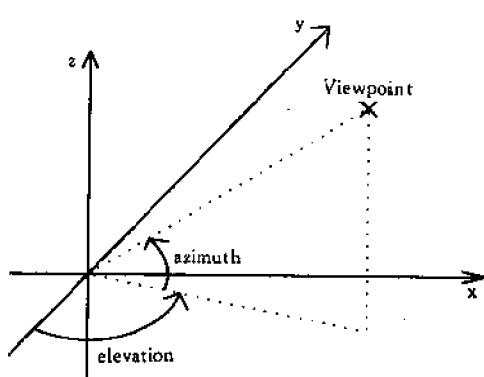


图 2-19 视点的位置

MATLAB 可以让用户从指定的视角（或方向）去观看生成的 3 维图形。函数 `view` 被用来设置新的视角方向。`view` 的输入参数是球面坐标，即关于原点的经度和纬度（或仰角）。经度是 x-y 平面的极坐标角度，正向角度表示视点的逆时针方向旋转的角度，其 0 度值对应于视点在 y 轴的负方向；而纬度（或仰角）是视点与原点构成的视线与 x-y 平面的夹角，纬度为 0 表示视点在 x-y 平面上，正值表示视点在 x-y 平面之上，负值表示视点在 x-y 平面之下。图 2-19 示意了坐标系中视点的定义，箭头方向表示正的方向。

例如，下面的例子展示了 4 个不同视点的图形，如图 2-20 所示：

```
>> subplot(2, 2, 1);
>> surf(peaks);
>> view(-37.5, 30);
>> subplot(2, 2, 2);
>> surf(peaks);
>> view(-7, 80);
```

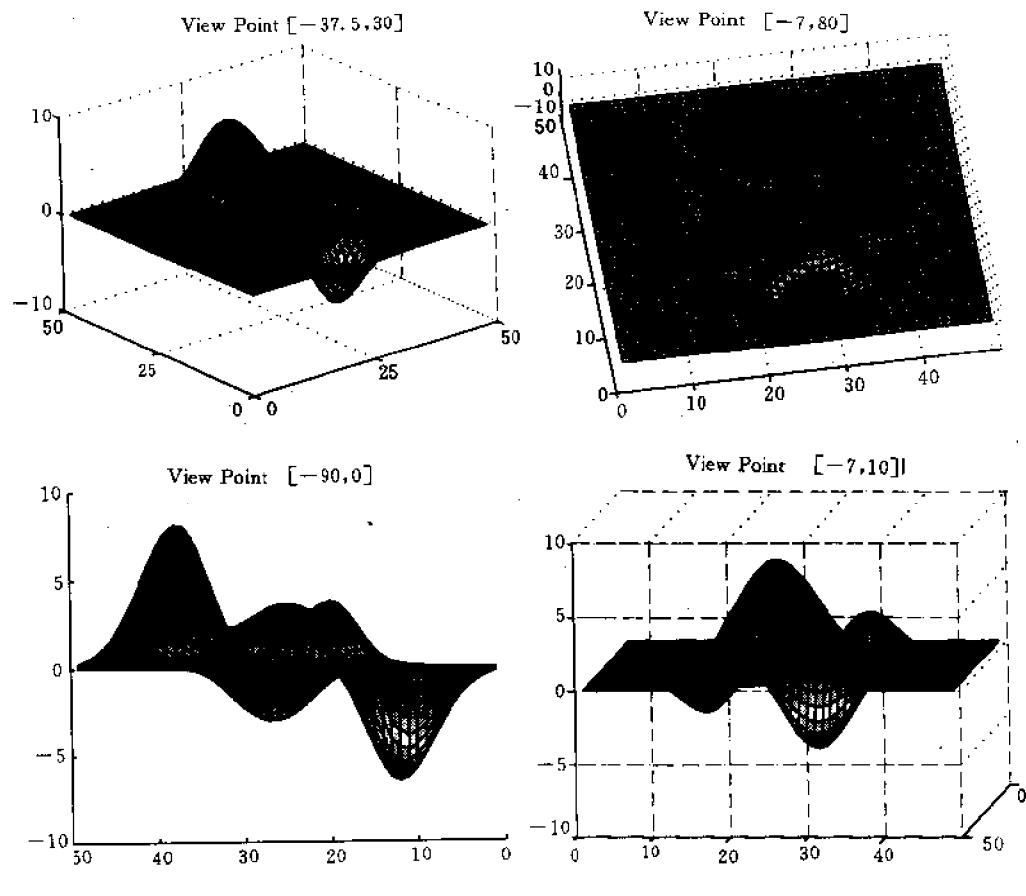


图 2-20 不同视点下的图形

```
>> subplot(2, 2, 3);
>> surf(peaks);
>> view(-90, 0);
>> subplot(2, 2, 4);
>> surf(peaks);
>> view(-7, -10);
```

视点的缺省值是 $[-37.5, 30]$, 通过改变同一坐标系的视点, 可以达到旋转坐标系中图形的目的, 在动画设计中, 经常使用这种技巧。值得一提的是, 在 MATLAB 的坐标系管理下, 平面坐标系被看作是视点在 $[0, 90]$ 方向上的 3 维坐标系, 因此, 可以通过改变经度的大小来旋转平面图形。

函数 `view` 作用在当前的坐标系上。对任何坐标系, 还可以通过修改坐标系视点属性来改变坐标系的视点。

2. 透視效应

MATLAB 绘制图形时, 在缺省的模式下, 前面的图形将挡住后面的图形, 即消去隐藏线, 但是, 可以用命令

```
>> hidden off
```

改变这种模式。相反地,可用命令

```
>> hidden on
```

重新返回到缺省的模式。例如,下面的语句首先生成 peaks 的网格曲面,然后在 x-y 平面上生成 peaks 的伪图。很明显,网格曲面挡住了部分伪图,如图 2-21(a)所示。使用 hidden off 命令后,被挡住的部分成为可见的,如图 2-21(b)所示:

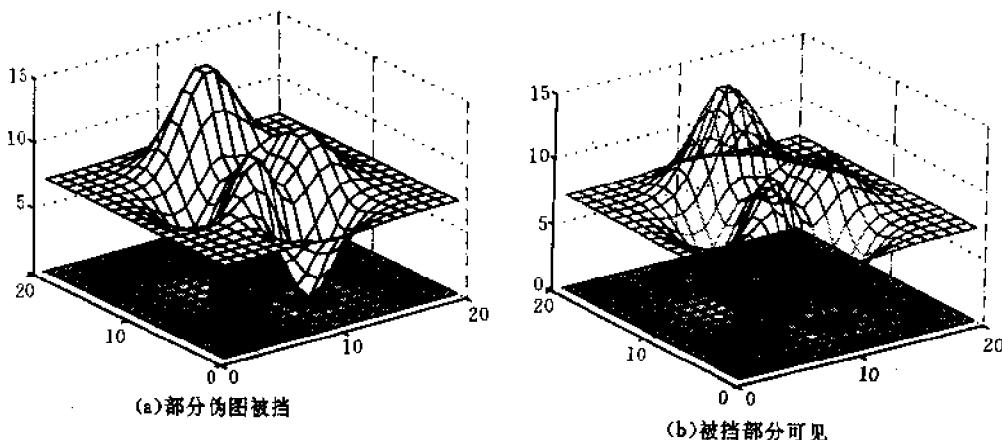


图 2-21 hidden on 与 hidden off 模式的图形

```
>> mesh(peaks(20)+7);
```

```
>> hold on;
```

```
>> pcolor(peaks(20));
```

```
>> hidden off
```

3. 曲面的裁剪技巧

MATLAB 总是在平面的矩形区域上表达曲面,那么,如何绘制其他区域上的曲面呢?这可以使用 MATLAB 的曲面裁剪功能来实现。在 MATLAB 中有两种实曲面的裁剪技巧:一种是利用着色原理将要裁剪的部分着当前的图形窗口的背景色,使其在坐标系中不可见,这种方法的技巧性较高,而且用这种方法裁剪的部分在图形打印时是可见的;另一种方法是使用 MATLAB 定义的特殊数 NaN,这个特殊数的一个重要性质是,它可以用于表示那些不可使用的数据,可利用这种特性进行曲面的裁剪。这种方法的原理是,将要裁剪的曲面片部分对应的矩阵(或函数值)元素设置为 NaN,然后,或者将该矩阵作为图形函数的矩阵变量,或者作为图形函数的颜色矩阵进行作图,即可以达到曲面裁剪的目的。例如,读者可以看看下列语句生成的图形,如图 2-22(a)所示:

```
>> P = peaks;
```

```
>> P(30 : 40, 20 : 30) = NaN * P(30 : 40, 20 : 30);
```

```
>> mesh(peaks, P)
```

当然,上述例子中的最后一条语句可以用下面的语句来代替:

```
>> mesh(P);
```

再看一个较复杂的例子,作函数 $z=x^2+y^2$ 在区域 $x^2+y^2 \leq 9$ 上的曲面片,如图 2-22(b)

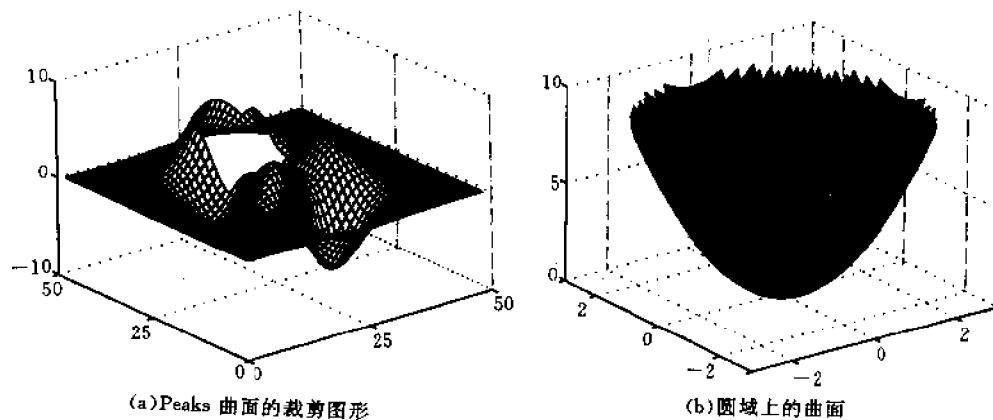


图 2-22 曲面裁剪技巧

所示：

```

>> [X,Y] = meshgrid(-3 : 0.1 : 3);
>> Z = X.^2 + Y.^2;
>> [I,J] = find(X.^2 + Y.^2 > 9);
>> for ii=1 : length(I)
>>     Z(I(ii), J(ii)) = NaN;
>> end
>> surf(X,Y,Z)

```

2.3 MATLAB 的色谱与着色原理

2.3.1 色谱

1. MATLAB 颜色表示法

在 MATLAB 中,每种颜色都用一个长度为 3 的实数向量表示,向量的元素取值范围是 $[0, 1]$,这 3 个数值分别表示计算机屏幕显像管红、绿、蓝 3 种颜色的光强度值,称为 RGB3 元数组值。表 2-2 中列出了几种常见颜色的 RGB 值。

表 2-2 常见颜色 RGB 对照表

红	绿	蓝	颜色	红	绿	蓝	颜色
0	0	0	黑色	1	0	1	洋红色
1	1	1	白色	0	1	1	蛋青色
1	0	0	红色	0.5	0.5	0.5	灰色
0	1	0	绿色	0.5	0	0	暗红色
0	0	1	蓝色	1	0.62	0.40	黄铜色
1	1	0	黄色				

2. 色谱

色谱(colormap)是 MATLAB 系统引入的概念。在 MATLAB 中,每个图形窗口都配备着一个色谱。色谱定义为由一组颜色的 R、G、B 值组成的列数为 3 的数值矩阵,称为色谱矩阵。色谱矩阵的第一列为各颜色的 R 值,第二列为 G 值,第三列为 B 值。色谱矩阵的行数称为色谱矩阵的长度。在弄清楚着色原理之前,有必要理解下列几个概念的区别:

颜色索引表

这是计算机的硬件术语

色谱

是由 RGB 值组成的 $m \times 3$ 维数值矩阵,每个 MATLAB 图形窗口关联一个色谱矩阵,即该图形窗口中图形对象可使用的颜色

伪色谱

相对于图像来说的非图像真实颜色的任何一个色谱

调色板

图像的特定色谱,在该色谱下,图像的颜色被真实地表示出来

MATLAB 的函数 colormap 可以用来取得当前的图形窗口的色谱,其调用命令为

`>> M = colormap;`

MATLAB 在创建图形窗口时,自动地为图形窗口设置一个色谱,随后用户可以通过函数 colormap 为图形窗口设定特定色谱。设 M 是一个色谱矩阵,那么语句

`>> colormap(M)`

的作用就是将当前图形窗口的色谱设置为 M,同时该语句使得该图形窗口中已存在的所有图形对象按新色谱改变颜色;随后创建新的图形对象时,也将使用这个新的色谱进行着色。

利用 8 bits 作为颜色索引表的计算机系统可以允许使用 256 种颜色的色谱,即色谱矩阵的行数可以是 256 行,可以从 $2^8=16\ 777\ 216$ 种颜色中选择出 256 种颜色。如果用户在一个图形窗口中使用颜色索引表中的 256 种颜色,那么 Windows 操作系统可能会改变窗口的背景、边界、字符等的显示颜色。所幸的是,目前的计算机硬件配置大都至少支持 64K 种颜色。这样,对用户来说,颜色资源是足够丰富的。

3. 系统色谱

由于色谱矩阵是一个数值矩阵,因此,它可以被看作是一个数据表,可以由 MATLAB 的数组或矩阵运算来定义。MATLAB 提供几个函数定义几种系统色谱。这些函数是: hsv、hot、cool、pink、copper 和 flag。每个函数可以有一个可选择的输入参数 m,表示生成长度为 m 的色谱矩阵。例如,函数 hot 的命令形式是

`>> M = hot(m);`

其他的函数调用方式相同。M 是一个 $m \times 3$ 的色谱矩阵,hot 生成的色谱矩阵表示的颜色是从黑色、暗红色、橙色、黄色到白色的由暗到明变化的各种颜色。

值得一提的是,在不指定参数 m 时,上述的色谱矩阵生成函数都是先取得当前图形窗口的色谱长度,再生成同样长度的色谱矩阵。通常 MATLAB 自动生成图形窗口时,设置的色谱矩阵是由 hsv 生成的色谱矩阵,长度为 64。这种做法的好处在于,可以同时有 3 个或 4 个图形窗口的色谱的颜色互不相同(即不超过 256 种颜色)。对于颜色资源有限的计算机系统,如果在不同的图形窗口中使用较长的色谱矩阵,那么在不同的图形窗口之间进行切换时,Windows 操作系统可能会改变非活跃图形窗口中图形对象的颜色。

使用色谱着色的图形生成函数有 mesh、surf、pcolor 和 image(下节中介绍),所有以这些函数为基础的图形生成函数都将使用色谱矩阵进行着色,但是线型图形生成函数,如

plot、plot3、contour 和 contour3 等则使用系统预定义的 7 种颜色着色,与色谱无关。

2.3.2 着色原理

1. 缺省着色方式

MATLAB 通过线性变换的方式将颜色矩阵的数值元素映射到色谱矩阵的行索引号,以该行的 RGB 颜色值所决定的颜色对颜色矩阵元素对应的图形位置进行着色。例如,在下述命令

```
>> surf(Z, C)
```

中,C 是颜色矩阵,MATLAB 将 C 中的最小元素映射到色谱矩阵的第一行 RGB 值,最大元素映射到色谱矩阵的最后一行 RGB 值,其他的元素按线性的方式映射到色谱矩阵的某一行 RGB 值。例如,如果颜色矩阵的元素 $C(1,1)$ 被映射到色谱矩阵的第一行 RGB 值,即 $C(1,1)$ 是 C 的最小元素,那么 $C(1,1)$ 对应的顶点或网格片就按色谱矩阵第一行 RGB 值对应的颜色进行着色处理,具体的着色方法由着色模式决定。

实现线性映射的算法如下:设 C 是颜色矩阵或数组, $c_{min} = \min(C)$, $c_{max} = \max(C)$, $m = \text{length}(\text{colormap})$,于是,C 的各元素 c 的色谱矩阵的索引号 index 由下列公式得出:

如果 $c_{min} \leq c < c_{max}$, 则

$$\text{index} = \text{fix}((c - c_{min}) / (c_{max} - c_{min}) * m) + 1$$

如果 $c == c_{max}$, 则

$$\text{index} = m$$

2. 坐标系颜色范围

为了有效地使用和控制色谱矩阵的颜色,MATLAB 提供了坐标系颜色范围的概念。坐标系颜色范围是一个二元的向量,它描述了在进行图形元着色时,最小和最大的可见颜色的颜色值的范围。例如,前面的缺省着色方式的颜色范围是 $[c_{min}, c_{max}]$,亦即颜色矩阵元素的最小值和最大值。这种做法有一个缺点,如果只有极少数元素数值远大于或远小于其他的颜色矩阵元素,那么,按缺省方式着色后,就很难通过图形的颜色来判定数值矩阵元素之间的大小关系。为此,MATLAB 通过设置 $[m_{inc}, m_{exc}]$ 的办法剔除一些不很明显的部分。设置当前坐标系的颜色范围可以通过下面的语句实现:

```
>> caxis([m_{inc}, m_{exc}])
```

在这种情况下,颜色矩阵与色谱矩阵的行指标的线性关系为:

如果 $m_{inc} \leq c < m_{exc}$, 则

$$\text{index} = \text{fix}((c - m_{inc}) / (m_{exc} - m_{inc}) * m) + 1$$

如果 $c == m_{exc}$, 则

$$\text{index} = m$$

如果 $c < m_{inc}$ 或 $c > m_{exc}$ 或 $c == \text{NaN}$, 则不对应任何颜色。

所以,利用 caxis,可以达到以下的效果:

- (a) 如果将 $[m_{inc}, m_{exc}]$ 设置得比颜色矩阵 C 的最小与最大值范围还要大,那么,真实的数据着色只是用到色谱中的部分颜色;

- (b) 如果将 [minc, maxc] 设置在颜色矩阵 C 的最小值与最大值的范围内, 那么那些小于 minc 和大于 maxc 的 C 的元素对应的顶点或区域将不被着色, 或者说其值为 NaN。

3. 全真颜色

如果计算机系统支持 24 位的全真颜色, 即同时支持 2^{24} 种不同的颜色, 那么 MATLAB 5.0 则提供了一种全新的着色方案。下面以函数 surf 为例来说明这种方法。对于命令

```
>> surf(X, Y, Z, C)
```

在前面介绍的着色法中, 变量 C 是一个与 Z 同维数的矩阵。对于曲面片上的每小块, 先将颜色矩阵 C 相应的元素映射到色谱矩阵的行索引号中, 再由该行提供的 RGB 值来决定用什么样的颜色着色。在 MATLAB 5.0 中, 变量 C 称为颜色数据, 它既可以是一个与 Z 同维数的矩阵, 也可以是一个 $m \times n \times 3$ 的 3 维数组, 其中 m 是 Z 的行数, n 是 Z 的列数。此时 $\{C(i, j, 1), C(i, j, 2), C(i, j, 3)\}$ 组成 RGB 值, 用于对 $Z(i, j)$ 对应的曲面小块着色。也就是说, 在这种情况下, 不必从色谱中挑选颜色。

例如, 下面的语句将对 peaks 曲面随机地进行着色:

```
>> Z = peaks(25);
>> C(:, :, 1) = rand(25, 25);
>> C(:, :, 2) = rand(25, 25);
>> C(:, :, 3) = rand(25, 25);
>> surf(Z, C);
```

2.3.3 色谱矩阵的分析

一幅图像具有或明或暗的效果, 这种效果可以从图像的调色板色谱上反映出来。本节对此略作分析。

1. 颜色强度曲线

设 M 是 $m \times 3$ 的色谱矩阵, 于是, 语句

```
>> plot(M)
```

将生成红、绿、蓝 3 种颜色光强度的曲线图形。在本书中, 用实线、虚线和点线分别表示红、绿、蓝 3 种颜色光强度的曲线。在计算机屏幕上, 可以用语句

```
>> rgbplot(M)
```

将 M 用红、绿、蓝的线型表示出来。MATLAB 的灰色色谱用函数 gray 生成, 该色谱矩阵每行的 3 个元素的数值相同, 即 3 种颜色的强度比例是一样的, 于是, 利用语句

```
>> plot(gray)
```

生成的图形如图 2-23(a) 所示, 3 条曲线是重合的。

色谱 hsv 也是 MATLAB 系统预定义的色谱, 它常用来显示极坐标系中的周期函数的图形。色谱 hot 对应的颜色是由暗到明地经过黑、暗红、橙、黄到白各种颜色。数学上, 红、绿、蓝 3 种颜色的强度值是按斜梯方式增加的。plot(hot) 的图形可以证实这一点, 如图 2-23(b) 所示。

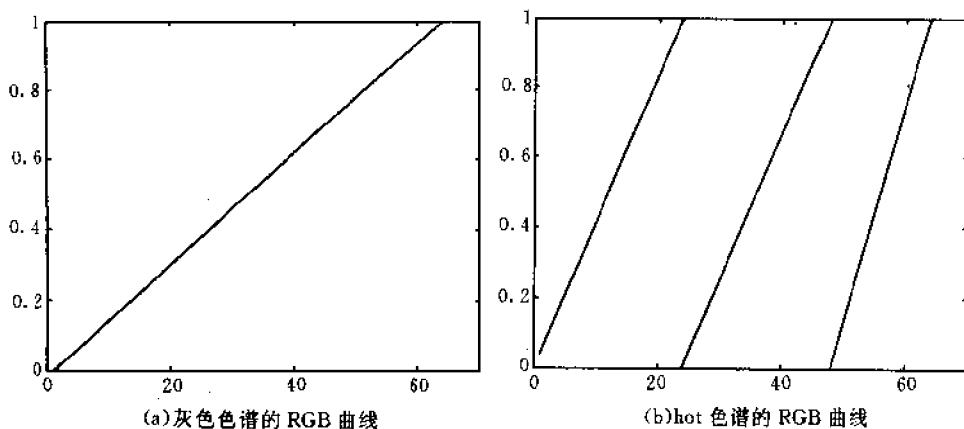


图 2-23 色谱 RGB 曲线

2. 色谱图像

利用 `pcolor` 函数可以将色谱在平面上展示出来。例如,为了看一看 8 个灰度的灰色色谱(见图 2-24),只需给出以下命令:

```
>> colormap(gray(8))
>> pcolor([1:9; 1:9]')
```

3. 色谱的调整

由于色谱是一个数值矩阵,故对色谱矩阵进行的数学运算可以获得其他性质的色谱。MATLAB 函数 `brighten` 可以用来增加或减弱色谱暗色部分的强度值,从而改变色谱的视觉效果。例如,语句

```
>> brighten(gray, 0.5)
```

将灰色色谱的强度按 \sqrt{x} 增强,新的色谱强度曲线如图 2-25 所示。

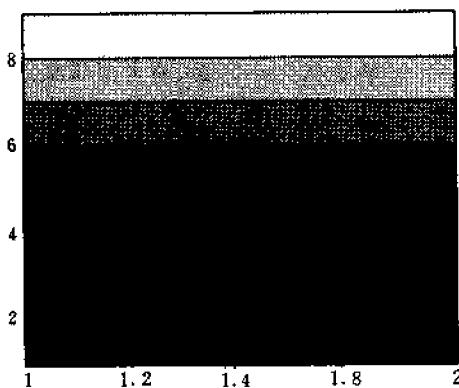


图 2-24 8 个灰度的灰色色谱

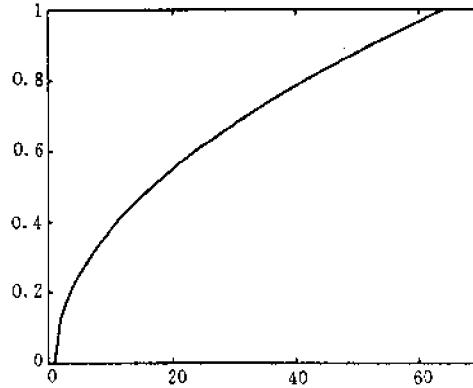


图 2-25 灰色的增强色谱 RGB 曲线

又如语句

```
>> brighten(hot, -0.5)
```

的作用是按 x^2 的方式减弱色谱的强度,如图 2-26 所示。

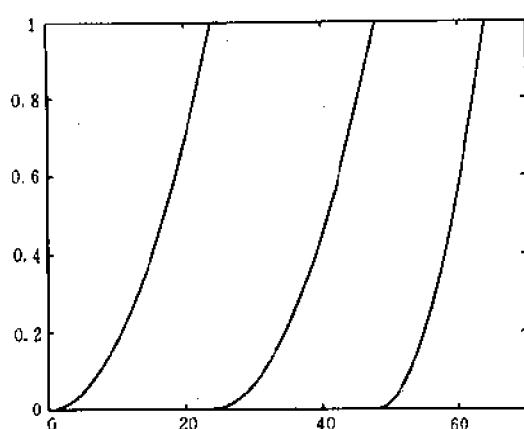


图 2-26 hot 的减弱色谱 RGB 曲线

也可以用数学的方式对色谱矩阵进行处理,得到新的色谱矩阵。当然,新的色谱矩阵表示的颜色效果有时是很难知道的。系统色谱 pink 就是按下列的方式得到的:

```
>> sqrt(2/3 * gray + 1/3 * hot)
```

2.4 图像处理

在 MATLAB 的基本系统中,有几个函数用于图像处理,进行图像的显示、矩阵伪色图像、动画演示等操作。这对于设计教学演示程序是很有用的。除此之外,MATLAB 的图像处理工具箱则提供了更多的图像操作功能,对图像处理感兴趣的读者可以进一步参考图像处理工具箱的使用说明。

2.4.1 伪色图像

读者可以在 MATLAB 系统上演示如下的例子:

```
>> Z = peaks;
>> pcolor(Z);
>> colormap(hot);
```

这 3 条语句在平面坐标系中根据色谱 hot 生成矩阵 Z 的伪色图。函数名 pcolor 是英文 pseudocolor(伪色)的缩写,这个函数的作用是将矩阵 Z 作为颜色矩阵,根据上一节提供的着色方法,在平面网格点(i, j)的右上角小区域内用 Z(i, j)对应的色谱矩阵的颜色着色。同时,网格线用黑色着色。

由于 contour 和 pcolor 都以图示的方法表示矩阵的数值大小关系,前者用等高线分出数值大小不同的区域,后者通过不同的颜色分出数值大小不同的区域。将二者结合在一起,可以更清楚地表达矩阵数值的大小关系。例如,下列语句生成如图 2-27 所示的图形:

```
>> colormap(hot);
>> pcolor(peaks);
>> shading flat
>> hold on
>> contour(peaks, 20, 'k');
>> hold off
```

这里,pcolor 用 flat 的着色模式,以消去黑色的网格线,而将所有的等高线着黑色,这样会使得效果更加明显。

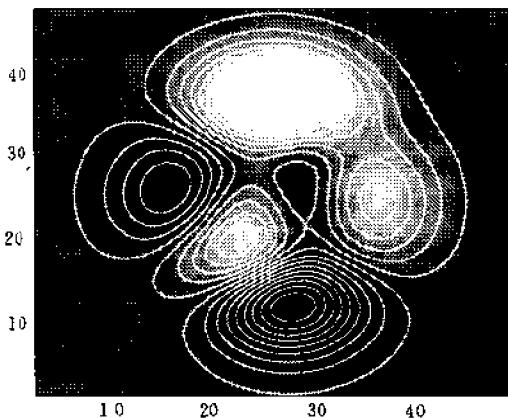


图 2-27 伪色与等高线

2.4.2 图像显示技术

1. image 与 imagesc 函数

我们知道,数字图像是指将一幅 2 维的图像表示成一个数值矩阵,矩阵的元素被解释为像素的颜色值(或灰度值),或被解释为调色板颜色的索引号。为了显示由矩阵表示的数字图像,MATLAB 最一般的做法是将矩阵的每个元素对应到当前色谱的某个行标号,并取该行的颜色值作为图像相应点的颜色。一般说来,每幅图的色调不同。因此,作为图像,必须有自己特殊的色谱,这样才能真实地显示图像。这个特殊的色谱就称为该图像的调色板。

MATLAB 用函数 `image` 显示图像,其命令形式为

```
>> image(X)
```

为了正确地显示图像,还必须将该图像的调色板装入到图形窗口的色谱中。例如,在 MATLAB 中,有一幅图像的数值 MAT 文件是 `gatlin.mat`,它包括了图像矩阵 `X` 和调色板矩阵 `map`,于是,下列语句就可以真实地再现这幅图像:

```
>> load gatlin
>> image(X);
>> colormap(map);
>> axis equal
>> axis off
```

在一般情况下,要保证正确的图像纵、横方向的比例。上述例子中的语句 `axis equal` 就是为了达到这样的目的。

在没有装入图像的调色板之前,`image(X)` 用的是当前的色谱,它的效果与 `pcolor` 的效果相类似。但是,它们之间有明显的区别。首先,`image` 是用来显示真正的图像的,包含装入调色板;而 `pcolor` 则是用图形的方式表达抽象的数学对象的数量关系。除此之外,它们还在以下几个方面存在着差别。

设矩阵 `A` 是 $m \times n$ 的数值矩阵

- (a) `image(A)`对 $m \times n$ 片小矩形进行着色, 横轴和纵轴网格的分点在 $k + 1/2$ 点处, 而 `pcolor(A)` 在 $(m-1) \times (n-1)$ 片小矩形上着色, 且对网格线着黑色(缺省色);
- (b) `image(A)` 对每小片矩形区域按 flat 模式(分片常值)着色, 而 `pcolor(X)` 可以有多种着色模式, 用 `shading` 函数来设置;
- (c) `image(A)` 采用 i-j 矩阵坐标系, 而 `pcolor(A)` 的缺省方式是 x-y 笛卡尔坐标系;
- (d) `image(A)` 总是生成等分的网格, 而 `pcolor(X, Y, A)` 可以生成参数型的网格;
- (e) `image(A)` 生成的 2 维图像, 只能从标准的 2 维视点(0, 90)观看, 即不能改变视点, 而 `pcolor(A)` 生成的图形可以从任何视点观看;
- (f) `image(A)` 直接用 X 的值作为索引在色谱中对应颜色值, 即 A 的元素被截成整数, 其范围是 0 到 `length(colormap)`, 而 `pcolor` 是根据坐标系的颜色区间(可以由 `caxis` 设置)按线性变换方式计算索引值来对应色谱矩阵的颜色值。`image(A)` 的图像不受 `caxis` 函数的影响;
- (g) `image(A)` 能固定与图像同样大小的坐标系, 使图像充满坐标系, 而 `pcolor` 不具备这种功能。

另一个与 `image` 函数类似的函数是 `imagesc`, 它的命令形式与 `image` 的命令形式是一样的。在讨论 `image` 与 `pcolor` 函数的区别时, 已经看到, `image(X)` 是将数据矩阵 X 的值直接作为索引号在色谱矩阵中提取 RGB 颜色值进行着色的。事实上, 对于任何矩阵 X, `image(X)` 可以生成一幅图像, 如果 X 的元素的数值大小十分接近, 或超出色谱矩阵的长度, 那么 `image(X)` 就不能有效地用图像表达矩阵 X, 而函数 `imagesc` 就可以做到这一点。`imagesc` 在功能上与 `image` 是一样的, 只是按线性变换的方式计算索引号, 即与 `pcolor` 使用的方式相同。于是, `imagesc(X)` 生成的图像将受 `caxis` 函数的影响。

2. 图像类型

从前面的分析已经看到, MATLAB 的图像由两部分构成, 一个是数据矩阵, 另一个 是图像的色谱, 即调色板。为了有效地使用全真颜色的图像, MATLAB 5.0 扩展了它的图像处理功能。它将图像分成以下 3 种类型。

(1) 索引式图像

图像数据矩阵 X 的元素定义为对应到色谱矩阵的索引号, 所以索引式图像必须有自己的调色板, 即一个 $m \times 3$ 的色谱矩阵 map。一般用下列语句显示一个索引式图像:

```
>> image(X); colormap(map)
```

(2) 强度式图像

图像数据矩阵 X 的元素值落在某个强度范围内, 一般情况下, 强度范围是 [0, 1] 或 [0, 255], 其强度值按线性变换的方式映射成色谱矩阵的行索引号, 例如, 如果强度范围是 [0, 1], 那么矩阵 X 中值为 1 的元素就映射成色谱矩阵的最大索引号, 即色谱矩阵的长度。这种方式一般应用于灰度级的黑白图像。函数 `imagesc` 用于显示这种类型的图像。例如:

```
>> imagesc(X,[0 1]); colormap(gray)
```

如果不指定强度范围, 那么, 图像数据矩阵中的最大元素值定义为最大强度, 最小元素值定义为最小强度。例如, 下列两组语句是等价的:

```
>> imagesc(X); colormap(map)
```

```
>> imagesc(X, [min(X(:)) max(X(:))]); colormap(map)
```

(3)全真颜色图像

与通常的全真颜色图像文件一样,全真颜色图像没有调色板,也就没有索引方式,只有一个 $m \times n \times 3$ 的 3 维数组,定义图像像素颜色的 RGB 值,记这个数组为 RGB,那么,RGB(:, :, 1)、RGB(:, :, 2)、RGB(:, :, 3)分别记录 R(红色)值, G(绿色)值和 B(蓝色)值。例如,像素点(10,5)的 R、G、B 值分别为 RGB(10,5,1)、RGB(10,5,2)和 RGB(10,5,3)。在 MATLAB 5.0 的系统中,用 image 函数显示全真颜色图像。例如:

```
>> image(RGB)
```

3. 图像数据的格式

在 MATLAB 中,系统用双精度浮点数即 64 位来表示一个数据。由于图像的数据量总是很大,为了节省有限的内存空间,MATLAB 提供了单字节(8 位)无符号整型类型数据来记录某些图像数据,这种类型称为 unit8。如果图像数据是用 unit8 类型的数据表示的,则这样的图像又称为 8 位型图像。

对于索引式图像,如果图像数据矩阵 X 的类型是 unit8,那么,它的色谱矩阵的长度最少为 256,这是因为 unit8 类型的数据值的范围是 0~255,构造索引号时,将 X 的数据加 1。用函数 image 显示这样的图像时,它自动对 X 的数据加 1。

对于强度式图像,如果图像数据矩阵 X 的类型是 unit8,那么,它的强度范围一般是[0, 255]。例如,用下面的命令显示 unit8 类型的强度式图像:

```
>> imagesc(X, [0 255]); colormap(gray)
```

对于全真颜色图像,如果 3 维数组 RGB 的类型是 unit8,那么任何一种 R、G、B 单色值的范围是[0, 255]。在这种情形下,如果一个像素的 RGB 值是(255, 255, 255),那么该像素就被着白色。对于全真颜色图像来说,无论 RGB 的类型是 unit8,还是 64 位的浮点型数,都用下面的命令显示该图像:

```
>> image(RGB)
```

可以用函数 unit8 和 double 进行两种类型之间的转换。例如:

```
>> RGB64 = double(RGB8)/255;
```

4. 读写图像文件

MATLAB 5.0 改进了原来的两个函数,即 imread 和 imwrite,它们分别用于将图像文件读入 MATLAB 工作空间,以及将图像数据和调色板数据一起写成一定格式的外部图像文件。在 MATLAB4.2 版本中,imread 和 imwrite 只支持 RAW 图像格式,新版本扩展了它的功能,能支持下列几类图像格式:

- BMP 格式(文件扩展名. bmp)
- HDF 格式(文件扩展名. hdf)
- JPEG 格式(文件扩展名. jpg)
- PCX 格式(文件扩展名. pcx)
- TIFF 格式(文件扩展名. tif)
- XWD 格式(文件扩展名. xwd)

函数 `imread` 在读入图像文件数据时, 根据图像文件的格式, 在 MATLAB 工作空间中创建 3 种类型的图像之一, 即索引式图像、强度式图像和全真颜色图像。

当图像文件是强度型的图像数据, 特别是灰度级图像文件时, `imread` 就创建一个类型为 `unit8` 的图像数据矩阵 `X`。例如:

```
>> X = imread('image.bmp')
```

当图像文件包含有索引数据和调色板数据时, `imread` 就创建一个类型为 `unit8` 的图像数据矩阵 `X`, 而将图像调色板中的数据转换为 `double` 类型的色谱矩阵 `map`。例如:

```
>> [X, map] = imread('image.pcx')
```

当图像文件是全真颜色的图像时, `imread` 将每个像素的 RGB 值直接读入到一个 3 维数组 `X` 中。例如, `image.jpg` 是全真颜色的 JPEG 格式的图像文件, 那么, 下列语句就把全部像素的 R、G、B 值装入到 3 维数组 `X` 中:

```
>> X = imread('image.jpg')
```

反过来, `imwrite` 则将 MATLAB 工作空间中的图像数据及其调色板数据按指定的图像格式写入到一个外部图像文件中。例如, 下列语句

```
>> load clown
>> imwrite(X, map, 'clown.bmp')
```

是将 MATLAB 的图像 `clown` 按 BMP 格式重新存储起来。

5. 图像调色板分析

在大部分的计算机数字图像文件格式中, 都包含有图像的调色板, 即该图像的主要的颜色成分。MATLAB 将图像对应于颜色矩阵, 而将图像的调色板看成是一个特定的色谱。在 MATLAB 系统中, 有一个图像文件 `clown.mat`, 可以用命令

```
>> load clown
```

将该图像装入到 MATLAB 工作空间中, 这个图像的图像数据矩阵是 `X`, 色谱矩阵是 `map`。于是, 利用语句

```
>> image(X);
>> colormap(map);
```

就会得到正确的彩色图像, 参见图 2-28 所示的黑白模拟图像。该图像的色谱即调色板的 R、G、B3 种颜色的强度曲线, 如图 2-29 所示。不难看出, 红色成分占主导地位, 所以该图像的色调是以红色为主。

在电视技术中, 表示明亮程度的 NTSC 标准值是 R、G、B3 值的线性变换, 即

```
>> b = 0.30 * red + 0.59 * green + 0.11 * blue
    = sum(diag([0.30 0.59 0.11]) * map')';
```

如果图像的调色板使得 $b = 0.30 * \text{red} + 0.59 * \text{green} + 0.11 * \text{blue}$ 的值是单调增的, 那么图像在灰色色谱下的视觉效果会比较好。一般说来, 可以用如下的非线性灰色色谱

```
>> colormap([b b b])
```

将其转换成 NTSC 的黑白图像。此时, 可以用线性的灰色色谱或者用 pink 色谱作伪色图, 其效果会是令人满意的。



图 2-28 clown 黑白模拟图像

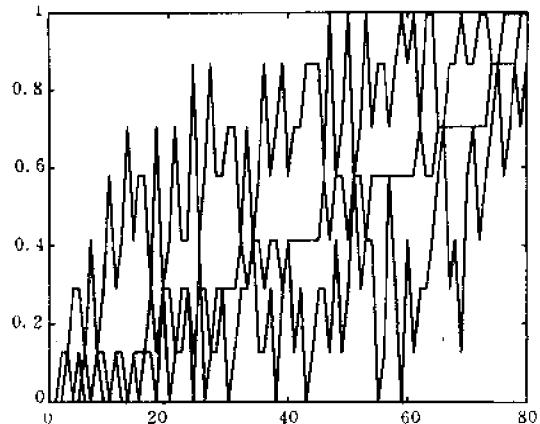


图 2-29 clown 的色谱强度曲线

2.4.3 动画

MATLAB 能提供动画功能, 用户可通过存储一系列的图片(图像), 然后将这一系列图片用 MATLAB 函数 movie 播放出来。系列图片必须用 MATLAB 函数 getframe 来制作。制作动画时, 对计算机的内存有较高的要求。为此, MATLAB 专门提供一个函数 moviein 来检测并生成存储系列图片的矩阵变量, 亦即分配足够的内存空间。下面的例子生成 16 帧图片的动画矩阵:

```
>> M = moviein(16);
>> for j = 1:16
    plot(fft(eye(j+16)))
    M(:,j) = getframe;
end
```

函数 getframe 返回当前图形窗口每个像素的颜色索引值矩阵, 作为一帧图片, 每帧图片存储在动画矩阵的某列向量中。由于每帧图片是按图形窗口的像素来构造的, 因此, 矩阵 M 列向量的长度只与图形窗口的当前尺寸有关, 而与图形内容的复杂程度无关。大的图形窗口要求较大的存储空间。

一旦动画矩阵生成, 就可以播放动画了。例如, 要演示上述动画 30 次, 则键入命令

```
>> movie(M, 30)
```

即可。这个例子的动画系图片只有 16 帧。只要内存允许, 可以用类似的方法生成较长的动画片断。

2.4.4 图形像素位置动态输入

MATLAB 函数 ginput 使用户可以用鼠标或方向键在图形窗口中进行动态输入, 该函

数返回鼠标箭头在当前坐标系中的位置坐标。下面是一个利用 ginput 函数在坐标系中根据所选点生成样条曲线的例子，平面中的点序列由 ginput 动态地取得，再根据样条函数 spline 进行插值，得到一条样条曲线，如图 2-30 所示：

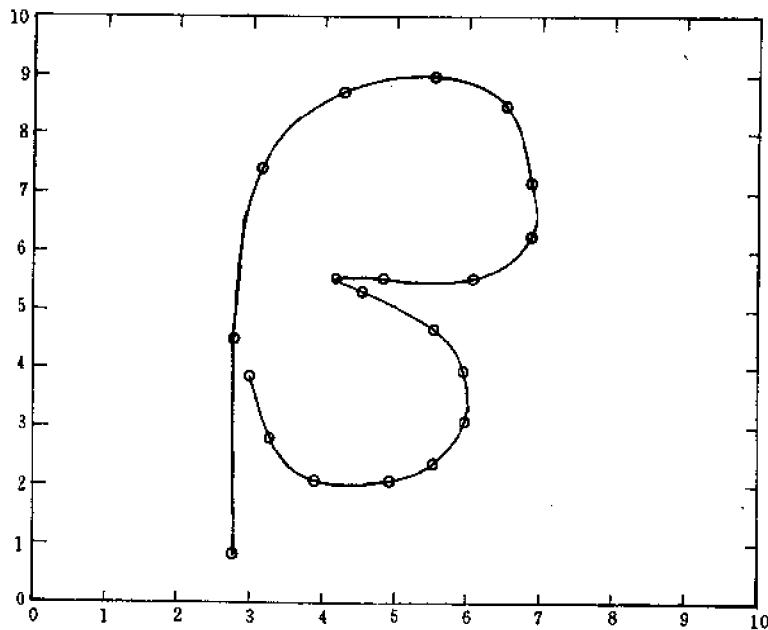


图 2-30 动态生成的 spline 曲线

```

>> clf
>> axis([0, 10, 0, 10])
>> hold on
>> x = [ ]; y = [ ];
>> n = 0;
>> disp('Left mouse button picks points')
>> disp('Right mouse button picks last points')
>> but = 1;
>> while but == 1
>     [xi, yi, but] = ginput(1);
>     plot(xi, yi, 'go')
>     n = n+1;
>     x(n, 1) = xi;
>     y(n, 1) = yi;
>     end
>> t = 1 : n;
>> ts = 1 : 0.1 : n;

```

```
>> xs = spline(t, x, ts);
>> ys = spline(t, y, ts);
>> plot(xs, ys, 'c-');
>> hold off
```

第三章 图形对象控制

到目前为止,本书所介绍的 MATLAB 图形的生成、管理和控制,都是 MATLAB 图形系统的高层操作。事实上,MATLAB 的图形系统还具有低层的图形操作功能,这就是 MATLAB 的图形句柄管理模式。

3.1 MATLAB 图形对象简介

3.1.1 图形对象类型与结构

MATLAB 系统内部使用对象语言描述各种图形单元,并将这些图形单元按照树型结构组织起来进行管理和实施各种操作。在 MATLAB 中,图形对象被分成:计算机屏幕(root)、图形窗口.figure)、坐标系(axes)、线段(line)、区域片(patch)、曲面(surface)、图像(image)、文字(text)、光源(light)、用户界面控制元(uicontrol)和用户界面菜单(uimenu)等图形对象类型。后两种对象统称为用户界面对象,第五章将对此作详细介绍。这里先介绍其他的图形对象。

在图形对象的树型结构中,计算机屏幕作为该结构的根,它的一级树节点是图形窗口对象;二级节点即图形窗口的子对象可以是坐标系对象、两种用户界面对象等;第三级树节点是坐标系对象的子对象,它们是线段对象、区域片对象、曲面对象、图像对象、文字对象、光源对象等,有时称为叶节点。

各种图形对象类型及其功能分述如下:

- root MATLAB 图形系统中的 root 对象是计算机屏幕,也是唯一的 root 对象,其他任何图形对象都是它的子孙
- figure 图形窗口对象是根对象的子对象。根对象可以有多个图形窗口对象,所有其他的图形对象都是图形窗口对象的子孙。在图形窗口对象未被创建时,所有的对象生成函数或高层的图形生成函数自动地创建一个图形窗口对象。也可以用函数 figure 直接创建一个图形窗口对象
- axes 称为坐标系对象。坐标系对象是图形窗口对象的子对象,其表现形式是图形窗口对象中的一块矩形区域。它的图形子对象都将在这块区域中显示。坐标系对象又是线段对象、曲面对象、文字对象、区域片对象等的父对象。在坐标系对象未被创建之前,所有的子对象生成函数和所有的高层图形生成函数都将创建一个坐标系对象,也可以用函数 axes 直接创建坐标系对象

line	线段对象是基本的图形对象之一,可以用来构成大部分的 2 维和 3 维图形,它们是坐标系对象的子对象,它的位置由它的坐标系对象决定。线段对象主要由高层函数如 plot、plot3、contour 和 contour3 等创建,也可以由低层生成函数 line 创建
patch	区域片对象是带边界的填充的多边形区域。它们是坐标系对象的子对象,其位置由坐标系对象决定,有常值型和插值型的填充模式。高层绘图函数 fill 和 fill3 生成区域片对象,也可以用低层函数 patch 创建
surface	曲面对象主要用来表达数据矩阵的 3 维图形,由数据矩阵定义的众多小四边形片构成。曲面对象可以按常值型或插值型的方式进行着色,曲面对象是坐标系对象的子对象,其位置由坐标系对象决定。高层函数 pcolor、surface 和 mesh 及其变形的函数都可以创建曲面对象,也可以由低层函数 surface 直接创建
image	图像对象是数据矩阵的映射图,数据矩阵被当作颜色矩阵来对平面的网格区域进行着色。图像对象通常有自己的调色板,即色谱,图像对象是 2 维的图形对象,不能改变图像对象坐标系的观点。图像对象是坐标系对象的子对象,它由函数 image 创建
text	文字对象是坐标系中的字符串。文字对象也是坐标系对象的子对象,其位置由坐标系决定。它们可以用函数 text 和 gtext 等创建
light	光源对象在一个坐标系内定义一个光源,坐标系中所有的图形对象都受它的影响。光源对象是不可见的,但是可以根据需要设置它的属性值,以控制光源的类型、颜色和位置,也可以设置其他一些图形对象的公共属性。它们可以用函数 light 创建
uicontrol	用户界面控制元对象是交互程序设计中引入的图形对象,利用它们可以启动某个函数或子程序的执行。它们是图形窗口对象的子对象,与坐标系无关
uimenu	用户界面菜单对象是图形窗口对象的子对象,它是图形窗口菜单条中的菜单项,也与坐标系无关

3.1.2 图形对象句柄及其访问

MATLAB 系统管理着众多的图形对象,在 MATLAB 中,每个具体的图形对象都有唯一的标识值,称为句柄。MATLAB 系统正是通过图形对象的句柄来标识、管理和操作每个图形对象的。句柄值是在图形对象创建时由系统自动设定的。下面对图形对象句柄的特点作简单介绍。

根对象(即计算机屏幕)的句柄值被赋予特别的标识值 0,且保持不变。图形窗口对象的句柄值是正整数,根据图形窗口对象创建的顺序给定其值。在缺省状态下,这个正整数被显示在图形窗口的标题栏上。其他的图形对象的句柄值是一个双精度的实数,句柄值关联着图形对象的系统管理信息。当用户使用句柄值来标识某个图形对象时,一定要保证该值的正确性。在不同的数据输出模式下,命令窗口回显的数据不一定是精确的句柄值,所以最好的办

法是将要使用的句柄值传递给一个变量,由这个变量将句柄值传送到所需要的地方。

MATLAB 定义了 3 个特别的函数来访问一些当前的图形对象,如:

- gcf 取得当前的图形窗口的句柄值;
- gca 取得当前的坐标系的句柄值;
- gco 取得当前的(第三级)图形对象的句柄值。

可以将这些函数取得的句柄值作为输入变量送到需要这些句柄值的函数中。MATLAB 的函数 delete 可以用来删除句柄值标识的任何图形对象。例如,要删去当前坐标系,其命令为

```
>> delete(gca)
```

这种方法可以用来关闭一个图形窗口。例如,语句

```
>> delete(2)
```

将关闭 2 号图形窗口。

所有的图形生成函数(无论低层还是高层函数)都返回生成对象的句柄值。例如,语句

```
>> h = plot([1 2 3 4 3 2])
```

就得到该线段对象的句柄值 h。有些函数如 contour 等生成多条线段对象,因此,返回一个句柄值的向量。

3.1.3 图形对象属性

在 MATLAB 中,每种图形对象都具有一定的属性,MATLAB 正是通过对属性的操作来控制和改变图形对象的。这些属性包括对象类型、对象的父对象和子对象句柄、对象是否可见等对象的公共属性,还包括不同类型对象的特殊属性,如坐标系对象的 x 轴性质等等。

在生成一个新的对象时,必须给对象的各种属性赋予必要的属性值。当用户不提供某种属性值时,系统将自动使用缺省的属性值。用户可以向系统询问某个对象的当前属性值,也可以设置新的属性值(注意:有些属性值只是可读的,不可以改变)。

1. 属性名与属性值

为了方便起见,MATLAB 给每种对象的每种属性规定了一个名字,称为属性名,而属性名的取值称为属性值。例如,LineStyle 是线段对象的一个属性名字,它的值决定着该线段的显示类型,比如是实型线段、虚型线段,还是点型线段等等。函数 plot 调用中的字符串变量的值就被传递到这个属性上。在属性名的写法中,MATLAB 不区分大小写字母。

2. 属性的访问

MATLAB 提供两种属性访问机制,可以在对象的生成函数中设定必要的属性值,也可以在对象生成后用函数 set 改变或设置新的属性值。例如:

```
>> figh = figure('Color', 'white');
>> axh = axes('View', [-37.5 30], 'XColor', 'k',...
    'YColor', 'k', 'ZColor', 'k');
>> surfh = surface(peaks, 'FaceColor', 'none', 'LineStyle', '.');
```

这 3 条语句的执行结果如图 3-1 所示。第一条语句创建一个图形窗口对象,用于画图,将图

形窗口的背景颜色设置为白色(缺省色为黑色);第二条语句在图形窗口中创建一个坐标系,坐标系的视点为[-37.5, 30](缺省的坐标系为标准的2维坐标系,即视点为[0, 90]),3条坐标轴都设置为黑色(缺省为白色);第三条语句生成 peaks 曲面,曲面网格片不着色,网格线段的线型为实点标记。

在程序设计中,经常使用的访问属性的方法是用 MATLAB 函数 set 和 get,使用这两个函数之前,对象应该已存在。

set 函数可以让用户设置图形对象的任何属性(除了受 MATLAB 保护的私有属性外)的属性值,它的第一个输入参数是图形对象的句柄值,其他的参数为“属性/属性值”对。例如,对前面的例子,如果要将网格线的线型改为实线线型,只需执行语句

```
>> set(surfh, 'LineStyle', '-');
```

就可以做到。又如,如果要将视点移到[-45, 45]的方向,则用下面的语句

```
>> set(axh, 'View', [-45, 45])
```

就可以实现。

set 函数的第一个变量可以是句柄值向量,此时它的作用是将指定的所有对象的指定属性设置为特定的属性值,当然每个句柄值对应的图形对象必须具有指定的属性。

在 MATLAB 5.0 中,用户甚至可以定义一个属性名和属性值的结构数组或块数组,来设置某些图形对象的相同属性值。例如:

```
>> view1.CameraViewAngleMode = 'manual';
>> view1.DataAspectRatio = [1 1 1];
>> view1.ProjectionType = 'Perspective';
>> set(gca, view1)
```

若要得出某个类型的对象的所有可设置的属性,调用 set 函数即可以达到目的。例如:

```
>> set(surfh)
Data
EdgeColor
FaceColor
LineStyle
MarkerSize
MeshSize
XData
YData
ZData
Children
```

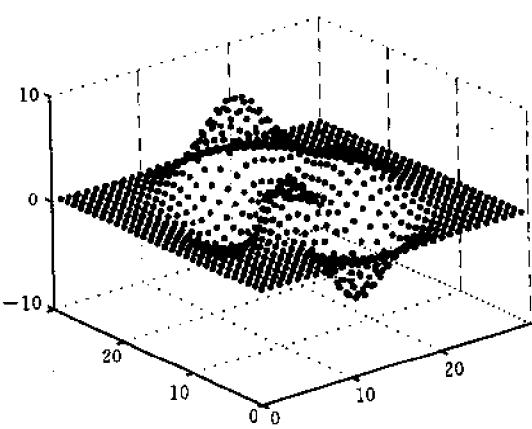


图 3-1 点阵型曲面 Peaks

```
Clipping:[on | off]
```

```
UserData
```

```
Visible:[on | off]
```

也可以向 MATLAB 系统询问对一个属性可以设置哪些合法的属性值,例如:

```
>> h = plot([1 2 3 4]);
>> set(h, 'Linestyle')
[ - | -- | : | -. | + | o | * | . | x ]
```

与 set 相对应, get 函数可以用来取得图形对象的各项属性的当前值。想要得到某个图形对象的所有当前值时,可以用 get(h),其中 h 为该图形对象的句柄值。例如:

```
>> get(surfh)
CData = [(too many rows)]
EdgeColor = black
FaceColor = none
LineStyle =
MarkerSize = [6]
MeshStyle = both
XData = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
         17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
         34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49]
YData = [(too many rows)]
ZData = [(too many rows)]
Children = []
Clipping = on
Parent = [0.000366211]
Type = surface
UserData = []
Visible = on
```

为了取得某个特殊属性值,除了给出图形对象的句柄外,只要在函数 get 的调用中给出属性名即可。例如,要确定曲面的最小 z 坐标值,可以由下面的语句实现:

```
>> z = get(surfh, 'ZData');
>> min(min(z))
ans = -6.5466
```

又如,要求得当前坐标系对象的所有子对象的句柄,只需执行下列语句:

```
>> h = get(gca, 'Children')
```

如果将 get 的输出值赋给一个变量,那么,MATLAB 将创建一个结构数组,该结构的各个域名是图形对象的属性名,而每个域值是当前的属性值。例如,设 x 和 y 是两个等长的向量,下面的命令创建一个 line 对象:

```
>> h = plot(x,y);
```

```
>> a = get(h);
```

而第二条语句将该对象的所有属性值按结构的方式赋给变量 a, 例如, a.color 的值就是该线段对象的颜色值。

通常, h 是一个列向量, 其元素的顺序由相应的对象的显示顺序决定, 第一个元素是最上面的图形对象的句柄, 如果还需要知道每个子对象的类型, 只要执行语句

```
>> get(h(1), 'Type')
```

```
>> get(h(2), 'Type') ...
```

即可。有些高层的图形生成函数生成多个图形对象, 这些函数返回所生成的图形对象句柄值的列向量, 例如:

```
>> h = surf(peaks)
```

```
h =
```

```
1. 0026  
2. 0026  
3. 0026  
4. 0026  
5. 0017  
6. 0015  
7. 0013  
8. 0013  
9. 0013
```

由于该函数可生成曲面和等高线, 因此, 通过这些句柄值, 可以访问它的属性 Type, 来判别每个句柄对应的图形对象的类型。例如:

```
>> get(h(1), 'Type')
```

```
ans =
```

```
surface
```

```
>> get(h(2), 'Type')
```

```
ans =
```

```
line
```

使用函数 get 和 set 访问对象属性的好处在于可以改变单个的图形对象的特性, 而不会破坏其他图形对象的属性。

函数 surf 生成的曲面用图形窗口的色谱着色, 而平面上的等高线用 MATLAB 预定义的 7 种颜色着色。如果希望将等高线的颜色改变成曲面上对应的颜色, 只要重新设置线段对象的 Color 属性值即可达到目的。为了简化这个过程, 用 findobj 函数找出全部线段对象, 该函数接受句柄值向量, 而输出具有指定属性和属性值的对象的句柄值组成的向量。下面的函数用于设置每个线段对象的颜色, 设该函数保存为 chagline.m。

```
function changeline(h)
```

```
lh = length(h);
```

```
map = colormap; % Get the current colormap
```

```

lm = length(map);
for jj = 1 : lh
    set(h(jj), colormap(jj * lm/lh, :))
end

```

该函数的功能是根据线型对象的多少,均匀地使用色谱中的几种颜色对线段型对象进行着色。将函数 findobj 和 chagline 组织在一起,构成新的函数 surfcc,用以生成新的着色的等高线:

```

function out = surfcc(x, y, z, c)
error(nargchk(1, 4, nargin))
if nargin == 1
    h = surfc(x)
elseif nargin == 2
    h = surfc(x, y)
elseif nargin == 3
    h = surfc(x, y, z)
elseif nargin == 4
    h = surfc(x, y, z, c)
end
hlines = findobj(h,'Type','Line');
chagline(hlines)
if nargout > 0
    out = h
end

```

3. 对象的拷贝与删除

MATLAB 5.0 提供了一个新的函数 copyobj,利用这个函数可以将图形对象从一个父对象拷贝到另一个父对象中,从而创建一个新的图形对象。新的图形对象与旧的图形对象只有句柄值和 Parent 属性值是不同的。既可以将多个图形对象拷贝到一个父对象中,也可将一个图形对象拷贝到多个父对象中,只要有正确的 Parent/Child 关系即可。

copyobj 的命令形式是

```
>> copyobj (handle1, handle2);
```

其中,handle1 是被拷贝的图形对象的句柄值,而 handle2 是希望拷贝到的父对象的句柄值。

为了将某些图形对象从图形中删除,只需使用 delete 命令

```
>> delete(handles)
```

即可。其中 handles 是句柄值向量。例如,如果要删除当前的坐标系,下列语句

```
>> delete(gca)
```

即可做到。

4. 公共属性

MATLAB 为每种图形对象定义了许多不同的属性,但是有些属性是所有的图形对象

共同具备的。如: Children 属性, Clipping 属性, Parent 属性, Selected 属性, SelectedHighlight 属性, Tag 属性, Type 属性, UserData 属性, Visible 属性等等。这里将对这些公共属性及其作用进行详细介绍。

(1) Children 属性

Children 属性的取值是对象句柄值组成的向量。这个向量由该图形对象所有的子对象的句柄值组成。例如, 对图形窗口对象来说, 其 Children 属性值就是所有的 Axes 句柄值, Uimenu 句柄值和 Uicontrol 句柄值。用户可以改变该向量元素的顺序, 那么, 所对应的子对象的显示顺序也就随之改变了。

坐标系的图形子对象可以是图像、光源、线段、区域片、曲面、文字等对象。虽然坐标轴的说明文字对象也是坐标系的子对象, 但是它们的句柄值不会出现在 Children 属性向量中, 因为这些文字对象的 HandleVisibility 属性取值为“callback”。如果需要临时从 Children 属性中查看这些子对象的句柄值, 可以将根对象的 ShowHiddenHandles 属性设置为“on”。例如:

```
>> h1=plot([1,2,3]);  
>> h2=xlabel('This is a test')  
>> h=get(gca,'children')
```

这时 h2 的值不会出现在 h 的分量中, 但是

```
>> set(0,'ShowHiddenHandles','on')  
>> h=get(gca,'Children')
```

可以使得 h2 出现在向量 h 的分量中。

对于所有第三级图形对象, Children 属性的取值为空矩阵。例如, 因为线段对象是第三级子对象, 故它自己没有子对象。

(2) Clipping 属性

Clipping 属性的取值是 on(缺省值)或 off。它决定图形对象被显示时, 超出图形窗口或坐标系范围的部分是否需要剪裁。

图形窗口对象本身不用这个属性, 它对坐标系对象也没有任何影响。对坐标系对象, Clipping 属性值为 on 时, 超出当前坐标系范围之外的曲面、区域、文字部分不会显示出来。通常, MATLAB 总是将图像放在坐标系的整个区域上。但是, 如果该属性值为 off, 图像可以在坐标系区域外显示出来。

(3) Parent 属性

Parent 属性的取值是某个图形对象的句柄值。对图形窗口对象而言, 它的 Parent 属性取值总是 0, 这是因为图形窗口对象的父对象总是根对象, 即计算机屏幕, 其句柄值为 0。

坐标系对象的 Parent 属性的取值是某个图形窗口的句柄值, 这是因为坐标系的父对象总是图形窗口对象。函数 gcf 返回当前坐标系所在的图形窗口对象的句柄值。

线段对象、曲面对象、区域片对象、图像对象、文字对象、光源对象等的 Parent 属性的取值是某个坐标系对象的句柄值, 因为这些对象总是坐标系对象的子对象, 该句柄值标明了这些对象所在的坐标系。如果用 set 函数将 Parent 属性设置为另一个坐标系的句柄值, 相当于将这些对象移到另一个坐标系中。例如,

```

>> load gatlin
>> subplot(211), h_image1=image(X)
>> hold on
>> load clown
>> h_image2=image(X)    %将原图像挡住
>> h_axes=subplot(212)
>> handles=get(h_axes,'Children')
>> set(h_axes,'Children',[handles,h_image2]) %将 clown 移开

```

(4) Tag 属性

Tag 属性的取值是一个字符串。在 MATLAB 系统内，系统是用句柄值来标识图形对象的。为了在程序与被调用的 M 函数之间共享该句柄值，需要将保存句柄值的变量定义为 global 变量，或作为实参数传递到被调用的 M 函数，这显然不够方便。

例如，如果希望将图形都输出到某个特定的图形窗口，可以用下列语句

```
>> figure('Tag','My Plotting Figure')
```

达到这个目的。即定义一个 Tag 属性值为 My Plotting Figure 的图形窗口，然后在每次要输出图形之前用 findobj 函数找到这个图形窗口，并使它成为当前的图形窗口：

```
>> figure(findobj('Tag','My Plotting Figure'))
```

又例如，如果在某段 M 文件子程序中，希望将所有的图形输出到同一个坐标系中，那么只要定义一个具有特定 Tag 属性的坐标系即可：

```
>> axes('Tag','My Special Axes')
```

然后，在每条图形输出语句前加入下列语句

```
>> axes(findobj('Tag','My Special Axes'))
```

即可达到目的。当然，此时其他的图形对象 Tag 属性值不能为 My Special Axes。

句柄值是 MATLAB 系统管理图形对象时所用的标识值，而 Tag 属性值是用户管理图形对象时所用的标识值。

Tag 属性的取值是字符串，它给该图形对象定义了一个标识值，当定义了 Tag 属性后，在任何程序中都可以通过这个标识值找出该图形对象。在整个程序中，每个图形对象的 Tag 属性值必须是唯一的。

(5) Type 属性

Type 属性的取值是字符串，这个属性说明该图形对象的类型，它的值是不可改变的。若 Type 属性的取值是 axes，说明该图形对象是坐标系对象；取值是 line，说明图形对象是线段对象；若取值是 surface，说明图形对象是曲面对象；取值是 patch，说明图形对象是区域片对象；若取值是 image，说明图形对象是图像对象；取值是 text，说明图形对象是文字对象；若取值是 light，说明图形对象是光源对象，等等。

(6) UserData 属性

UserData 属性的取值是一个矩阵。对所有的图形对象，这个属性的意义是一样的。在程序设计中，可以将一个与图形对象有关的比较重要的数据存储在这个属性中。图形对象本身并不一定使用这个矩阵，但是，用户可以用函数 set 和 get 访问这个属性。

性, HandleVisibility 属性, Interruptible 属性等。在介绍这些属性之前, 先来简要地说明 MATLAB 的一个很重要的概念 callback。在 MATLAB 中, 一个 callback 通常是指那种打断程序运行的某种操作, 即中断。它们可以是由一小段 MATLAB 语句或 M 文件定义的。

(10) HandleVisibility 属性

HandleVisibility 属性的取值为 on(缺省值), 但有些对象, 例如坐标轴说明文字的该缺省值为 callback 或 off, 这个属性定义图形对象句柄的可访问权限, 决定其句柄值是否在父对象的 Children 属性中出现。如果 HandleVisibility 的属性值是 on, 那么图形对象的句柄值总是可见的; 例如, 该图形对象的句柄值可以出现在定义对象的 Children 属性中; 如果 HandleVisibility 的属性值为 callback, 那么该图形对象句柄值只在图形对象的 Callback 调用以及 CallBack 调用函数内是可见的, 但是命令行内调用的函数则不能访问这样的图形对象句柄, 也就是说, 这样的句柄对于该图形对象自己的 CallBack 调用是私有的。这对于保护交互式的程序是十分有用的; 如果 HandleVisibility 属性值被设置为 off, 那么图形对象的句柄在任何时候都是不可见的。如果句柄是不可见的, 则任何查询句柄的函数都不能返回它的值, 这些函数包括 get、findobj、gcf、gca、gco、newplot、cla、clf、close 等。

当 HandleVisibility 属性值是 callback 或 off 时, 这样的图形窗口对象不会出现在根对象的 Children 属性和 CurrentFigure 属性中, 而且它的任何子对象也不会出现在根对象的 CallbackObject 属性和 CurrentObject 属性中。但是, 可以将根对象的 ShowHiddenHandles 属性设置为 on, 临时地使得所有句柄都是可见的, 而不管其 HandleVisibility 属性值是什么。

被隐藏的句柄仍是有效的, 如果知道这样的句柄值, 则一切关于句柄的操作都是合法的。

(11) BusyAction 属性

BusyAction 属性的取值是 cancel 或 queue(缺省值), 该属性决定 MATLAB 采取的控制中断执行对象的 Callback 调用的方式。在 MATLAB 环境中, 如果某个图形对象的 Callback 调用的子程序正在运行, 那么随后的其他类型的 Callback 调用(例如用户按下某个控制按钮, 希望运行另一个子任务)总是试图打断正在运行的 Callback。此时, 正在运行的 Callback 是否可以被中断, 要由该图形对象的 Interruptible 属性和 BusyAction 属性共同决定。如果此时 Interruptible 属性值为 yes, 那么 MATLAB 在处理事件队列时, 就会中断正在运行的 Callback, 转而执行用户所希望的动作; 如果 Interruptible 属性值为 no, 那么就由 BusyAction 属性来决定 MATLAB 对事件处理的方式。如果 BusyAction 属性值是 Cancel, 就从事件队列中取消企图运行第二个 Callback 调用的事件(例如, 按下鼠标的事件); 如果 BusyAction 的属性值是 queue, 就将企图运行第二个 Callback 调用的事件加入到事件队列, 直到当前的 Callback 调用完成为止, 再处理该事件对应的 Callback 调用。

(12) ButtonDownFcn 属性

ButtonDownFcn 属性的取值是字符串, 一般是某个 M 文件名或一小段 MATLAB 语句。当用户在图形对象上按下鼠标键时, 程序执行该属性定义的 Callback 调用。图形对象决定了一个作用区域, 称为热区。鼠标键在该区域内按下时, 就执行 ButtonDownFcn 定义的 Callback。例如, 设 h 是某个图像对象的句柄值, 执行下列语句:

```
>> set(h,'Button Down Fcn','disp("This is an image")')
```

此后,在该图像上点按鼠标一次,命令窗口中就回显一行文字“This is an image”。

(13)CreateFcn 属性

CreateFcn 属性的取值是字符串,一般是某个 M 文件名或一小段 MATLAB 语句,即当 MATLAB 创建该图形对象时,MATLAB 事先应该执行的程序段。

这个属性只能按设置缺省值的方式定义,这是为了保证它能事先运行。例如,下列语句

```
>> set(0,'DefaultLineCreateFcn','set(gca,"color",[0.5 0.5 0.5])')
```

是从根对象层设置线段对象的缺省值,随后,MATLAB 生成线段时,就用灰色作坐标系背景色。对于已经存在的线段对象,改变该属性之值,对该线段无任何影响。

MATLAB 在创建图形对象时,总是在完成所有缺省属性值的设置后,再执行 CreateFcn 属性定义的操作。事实上,CreateFcn 属性是用户自定义的缺省值,当然在图形对象生成之前执行。参见 3.10 节。如果一个图形对象的 CreateFcn 属性定义的操作正在运行,那么该图形对象的句柄值不可访问,必须用函数 `gcbo` 获取。

(14)DeleteFcn 属性

DeleteFcn 属性的取值是字符串,一般是某个 M 文件名或一小段 MATLAB 语句。当删除图形对象时,MATLAB 在清除该对象属性之前,执行由 DeleteFcn 属性定义的 Callback。

当用 `delete` 命令删除图形窗口对象时,一般不执行 DeleteFcn 属性定义的 Callback。

如果一个图形对象 DeleteFcn 定义的 Callback 正在运行,那么该图形对象的句柄值不可访问,必须用函数 `gcbo` 获取。

(15)Interruptible 属性

Interruptible 属性的取值是 `on` 或 `yes`(缺省值)与 `off` 或 `no`,这个属性决定图形对象的 Callback 是否可以被随后的 Callback 调用中断。只有由 ButtonDownFcn 属性定义的 Callback 受 Interruptible 属性值的影响。在程序运行过程中,只有当遇到 `drawnow`、`figure`、`getframe` 和 `pause` 等命令时,MATLAB 系统才检查事件队列。

3.2 图形窗口对象

图形窗口对象是 MATLAB 系统管理的一类很重要的图形对象。MATLAB 的一切图形图像的输出都是在图形窗口对象中完成的。同时,在交互式程序设计中,图形窗口对象的设计是任务的重点。因此,掌握好图形窗口对象对于充分发挥 MATLAB 系统的优秀图形功能和设计高质量的交互界面是十分必要的。本节将全面介绍图形窗口对象的创建、设计与管理方法。

3.2.1 图形窗口对象创建函数

在 MATLAB 图形对象的树型结构中,图形窗口对象是第一级树节点,亦即根节点(计算机屏幕)的子对象。在本书第二章中,已经介绍过创建图形窗口对象的几种常用方法。一是按照 MATLAB 的规则,当不存在任何图形窗口对象时,所有的图形窗口子对象创建函数

将自动地创建一个图形窗口对象作为该子对象的图形输出窗口；二是利用高层图形窗口创建函数 `figure`，它具有 3 种基本命令形式：

```
>> figure
>> h = figure
>> figure(h)
```

第一、二种形式用于创建一个新的图形窗口对象，且第二种形式返回该对象的句柄值；第三种形式有两种用处，如果以 `h` 为句柄值的图形窗口存在，那么这条语句的作用是使这个图形窗口成为当前的图形窗口；如果句柄值为 `h` 的图形窗口对象不存在，那么这条语句的作用就是创建一个新的图形窗口对象，并且以给定的 `h` 为其句柄值。

创建图形窗口对象的更一般方法是使用低层创建函数，该函数名为 `figure`，它的命令形式是

```
>> h = figure('属性名', 属性值...)
```

这种方法主要用在需要特定属性的图形窗口对象的情况下。通常都是使用高层创建函数按 MATLAB 设定的缺省属性值方式创建图形窗口对象的。

3.2.2 图形窗口对象的属性

MATLAB 5.0 为每个图形窗口对象新增加了 14 个属性，加上原有的 38 个属性，每个图形窗口对象共有 52 个属性。这些属性及其取值控制着图形窗口对象。除了公共属性外（参见 3.1.3 节），所有属性分述如下。

1. 外观特征属性

(1)MenuBar 属性

`MenuBar` 属性的取值可以是 `figure`（缺省值）或 `none`。它是一个开关，用来控制图形窗口是否应该具有菜单条。如果它的属性值为 `none`，则表示该图形窗口没有菜单条。

(2)Name 属性

`Name` 属性的取值可以是任何字符串，它的缺省值为空。这个字符串作为图形窗口的标题，一般情况下，其标题形式为：“Figure No. 1：字符串”。

(3)NumberTitle 属性

`NumberTitle` 属性的取值是 `on`（缺省值）或 `off`。它是一个开关，决定着在图形窗口的标题中是否以“Figure No. N：”为标题前缀，这里 `N` 是图形窗口的序号，即句柄值。

(4)Resize 属性

`Resize` 属性的取值是 `on`（缺省值）或 `off`。它是一个开关，决定着在图形窗口创建后可否用鼠标按 Windows 系统的方式改变该窗口的大小。在交互式程序设计中，`Resize` 属性值为 `off` 的图形窗口一般用来作说明页，其窗口大小不可用鼠标改变。

(5)WindowStyle 属性

`WindowStyle` 属性的取值是 `normal`（缺省值）或 `modal`。这是 MATLAB 5.0 新增加的一个属性，在 5.0 版中，图形窗口对象被分为两类：`normal` 图形窗口和 `modal` 图形窗口。`modal` 类图形窗口捕捉所有 MATLAB 窗口上的键盘和鼠标数据，只要这些窗口是可见的

(对于图形窗口,它的 Visible 属性值为 on)。modal 图形窗口总是位于窗口栈的最上层,如果有多个 modal 图形窗口存在,最后一个创建的 modal 图形窗口位于所有窗口的上端,直到它不可见,或 WindowStyle 属性转为 normal,或被删除时,第二个 modal 图形窗口便成为最上层的 modal 图形窗口。

如果一个 modal 图形窗口的 Visible 属性值为 off,那么这个图形窗口便不再有 modal 图形窗口的功能,除非它成为可见的窗口。因此,将一个 modal 图形窗口隐藏起来比删除该窗口更有效些。例如,在设计对话框窗口时可以使用这种技巧。

modal 图形窗口不显示菜单栏,但是可以在一个图形窗口中创建菜单对象。这些对象是存在的,当 WindowStyle 属性被设置为 normal 时,这些菜单项就会显示出来。

modal 图形窗口一般用于设计对话框,迫使用户必须对对话框作出回应。在 MATLAB 命令窗口中,命令

```
>> control C
```

将所有的 modal 图形窗口转为 normal 图形窗口。

(6)Position 属性

Position 属性的取值是一个 4 元素的向量,其形式为 [left, bottom, width, height]。这个向量定义了图形窗口对象在计算机屏幕上的位置,其中, left 为图形窗口的内左边界到计算机屏幕左边界的距离, bottom 为图形窗口内底边界到计算机屏幕底边界的距离, width 为图形窗口内区域的宽度, height 为图形窗口内区域的高度,它们的单位由 Units 属性决定。

Position 定义的区域不包括图形窗口的边界和窗口标题栏及菜单栏。

(7)Units 属性

Units 属性的取值可以是下列字符串中的任何一种:pixel(像素,为缺省值),normal(相对单位),inches(英寸),centimeters(厘米),points(磅)。

这个属性定义图形窗口使用的长度单位,由此决定图形窗口的大小与位置。除了 normal 以外,其他单位都是绝对度量单位。相对单位 normal 将计算机屏幕左下角对应为 (0,0),而右上角对应为 (1.0, 1.0)。该属性对 CurrentPoint 属性和 Position 属性的取值有影响,如果在程序中改变过 Units 属性值,在完成相应的计算后,最好将 Units 属性值设置为缺省值,以防止影响其他函数计算。在创建图形窗口对象时,如果不使用 Units 属性缺省值,那么必须在给定依赖 Units 属性的属性值之前先定义 Units 属性值。请读者自己检验下列 3 条语句的效果:

```
>> figure('position',[0.1 0.1 0.5 0.5])  
>> figure('units','normal','position',[0.1 0.1 0.5 0.5])  
>> figure('position',[0.1 0.1 0.5 0.5],'units','normal')
```

不难发现,第一条语句与第三条语句的作用是一样的,这是因为在这两条语句中,Position 属性所使用的单位都是缺省单位 pixels。

(8)Color 属性

Color 属性的取值是一个颜色值,既可以是 MATLAB 预定义的几种颜色(用字符表示,如'r'代表红色),也可以是任何一个三元素的 RGB 向量。它的取值决定了图形窗口的背景颜色。缺省值为 'k',即黑色。

2. 控制色谱的属性

(1) Colormap 属性

Colormap 属性的取值是一个 $m \times 3$ 阶的 RGB 颜色矩阵, 它的缺省值是包含 MATLAB 预定义的 64 种颜色的色谱。色谱矩阵的每一行 RGB 值决定一种颜色。在这个图形窗口中, MATLAB 通过行索引号决定某种图形对象应着的颜色。色谱矩阵的长度 m 可以是任意的, 但在 PC 机上一般限制为 256。只有 Surface、Image、Patch 等图形对象的着色受 Colormap 属性值的影响。

(2) DitherMap 属性

DitherMap 属性的取值是一个 $m \times 3$ 阶的 RGB 颜色矩阵, 它的缺省值所定义的颜色均匀地覆盖了整个色谱。这个色谱矩阵用在低颜色分辨率的计算机系统上, 模拟全真颜色 (TrueColor) 的色谱。MATLAB 5.0 系统本身支持全真颜色。但是使用 MATLAB 5.0 的计算机系统不一定能支持全真颜色。为了模拟出全真颜色的图形和图像, 此时就要使用这个色谱矩阵。用户可以改变这个色谱矩阵, 以使模拟的图形图像更逼真。例如, 如果一幅图像的色调是偏红的, 那么系统缺省的 DitherMap 色谱就不适于对这幅图像着色, 因为缺省的 DitherMap 均匀地定义整个色谱上的某些颜色, 这时可以重新定义一个新的 DitherMap 色谱矩阵。

(3) DitherMode 属性

DitherMode 属性的取值是 auto 或 manual(缺省值), 定义 MATLAB 系统使用与生成 DitherMap 色谱矩阵的模式。在 manual 模式下, MATLAB 用 DitherMap 属性定义的色谱矩阵模拟全真颜色的图形和图像; 而在 auto 模式下, MATLAB 根据图形窗口当前使用的颜色生成一个 DitherMap 色谱矩阵, 这对于显示出较满意的图形图像是必要的。但是重新生成 DitherMap 色谱矩阵是一件费时的工作, 而且当向该图形窗口增加新图形对象或改变图形窗口大小时, 生成 DitherMap 色谱矩阵的过程会重新进行。一般情况下, 可以将生成的 DitherMap 色谱矩阵保存下来, 并将 DitherMode 属性值设置为 manual。

(4) FixedColors 属性

FixedColors 属性的取值是一个 $m \times 3$ 阶的 RGB 颜色矩阵, 它的缺省值为 $[0, 0, 0]$ 。这是一个只可读取的属性, 用户不能用 set 函数对其进行更改。它记录着图形窗口中使用的那些不是由 Colormap 定义的所有颜色的 RGB 值, 这些颜色包括图形窗口的背景色、坐标轴颜色、Line、Text、Uicontrol 和 Uimenu 等子对象的颜色, 以及用户直接定义的颜色等等。例如, 从下列语句

```
>> get(gcf,'FixedColors')
>> set(gcf, 'Color', [.3 .7 .9])
>> get(gcf,'FixedColors')
```

不难看出, FixedColors 色谱矩阵的长度增加了。

由 FixedColors 属性定义的颜色与 Colormap 色谱矩阵定义的颜色在计算机系统的色谱表中占据着不同的位置, 因此, 如果 FixedColors 属性定义的颜色太多, 对于颜色资源有限的计算机系统来说, 可同时显示的颜色数会受到影响。

(5) MinColorMap 属性

MinColorMap 属性的取值是一个整数值,它的缺省值为 64。它定义了 MATLAB 向操作系统申请的系统颜色表项的最小数目。每个表项对应 Colormap 中的一种颜色。通常情况下,需要增大这个属性值,以保证 MATLAB 的图形窗口真正使用 ColorMap 定义的全部颜色。例如,如果某个图形窗口的 ColorMap 色谱矩阵的长度是 200,而此时计算机系统可供使用的颜色数目(也即系统颜色表所剩空间不多)低于 200,那么这个图形窗口中显示的图形或图像的着色只能用到少于 200 种的颜色。为了使用 ColorMap 中的全部颜色,办法之一就是将 MinColorMap 属性值设置为 ColorMap 色谱矩阵的长度,即

```
>> set(gcf, 'MinColormap', length(get(gcf,'Colormap')))
```

但是这样做的结果是可能导致其他非活跃窗口的颜色失真。

(6) ShareColors 属性

ShareColors 属性的取值是 yes(缺省值 5.0 版为 on)或 no。它的作用是对于相近的颜色,在计算机系统颜色表中共享同一种颜色。这个属性对 MATLAB 系统向计算机系统请求系统颜色表项的方式有所影响。在缺省模式下,MATLAB 找出计算机系统颜色表中已定义的颜色,并用其对图形窗口中的指定像素着色,这可以有效地使用计算机系统有限的颜色资源,并增加同时正确显示颜色的图形窗口的数目。如果在改变图形窗口色谱时,用户不希望 MATLAB 对图形窗口中已存在的图形对象重新着色,则可以先将 ShareColors 属性设置为 off。

3. 控制窗口刷新方式的属性

(1) BackingStore 属性

BackingStore 属性的取值是 on(缺省值)或 off。如果 BackingStore 属性值为 on,则 MATLAB 在内存缓冲区中保存该图形窗口的拷贝。当图形窗口被挡住的部分需要重新显示时,MATLAB 就直接从缓冲区中拷贝出窗口的内容,而不必重新生成那些图形子对象。这样可以加快屏幕的刷新速度。如果计算机系统内存资源有限,而在乎屏幕刷新速度时,就可以将 BackingStore 属性值设置为 off,以便节省内存开销。

(2) Renderer 属性

Renderer 属性的取值是 painters(缺省值)或 zbuffer,这个属性决定了图形窗口中图形的着色手段。painters 模式是 MATLAB 定义的基本方式,当图形窗口中只有简单或较小的图形子对象时,这种模式提供的显示速度较快;在 zbuffer 模式下,MATLAB 能更快地以逐个像素的方式绘制图形子对象,因为 MATLAB 只需要对在屏幕上可见的像素点进行着色。但是,这种模式要消耗大量的内存资源。Renderer 属性的取值由 RendererMode 属性决定,Renderer 属性的缺省值是 auto,意味着 Renderer 属性的取值由 MATLAB 自己根据情况决定。如果它的取值设置为 manual,则表明 Renderer 的属性值可以由用户用函数 set 来设置。

4. 当前子对象属性

(1) CurrentAxes 属性

CurrentAxes 属性的取值是图形窗口中当前坐标系子对象的句柄值。一个图形窗口对象可以有多个坐标系子对象,但是在任一时刻只有一个坐标系子对象是当前的坐标系,它的句柄值记录在 CurrentAxes 属性值中。

(2) CurrentCharacter 属性

CurrentCharacter 属性的取值是一个字符, 记录着用户在该图形窗口中最后一次键入的字符。

(3) **CurrentObject** 属性

CurrentObject 属性的取值为图形窗口的某个子对象的句柄值, 这个图形子对象位于图形窗口当前点(**Current Point**)下, 且图形窗口的子对象栈顺序中位于最上层。所以, 可以通过这个属性知道在图形窗口中用户用鼠标最后选择的子对象。一般用函数 **gco** 取得这个句柄值。

(4) **CurrentPoint** 属性

CurrentPoint 属性的取值是两元素的向量。这个向量表示在图形窗口中用户最后一次按鼠标的位置, 其值是相对于图形窗口的左下角点为原点的坐标值, 单位由 **Units** 属性确定。在图形窗口中, 每按一次鼠标, MATLAB 就会更新这个属性值。

(5) **SelectionType** 属性

SelectionType 属性的取值可以是 **normal**(缺省值)、**extended**、**alt**、**open** 中的任何一个, 用户不能修改这个属性值。**SelectionType** 记录着在图形窗口中最后一次按鼠标的方式。在 Windows 环境中, **normal** 表明点按鼠标左键; **extended** 表明按 Shift 键+点按鼠标左键或同时点按鼠标左右键; **alt** 表明按 Control 键+点按鼠标左键; **open** 表明双击鼠标左键。

5. Callback 调用属性

图形窗口对象共有 9 个属性可以定义 **Callback** 操作, 除了图形对象的公共属性 **ButtonDownFcn**、**CreateFcn**、**DeleteFcn** 外, 还有以下几个定义 **Callback** 调用的属性。

(1) **CloseRequestFcn** 属性

CloseRequestFcn 属性的取值是字符串, 一般是某个 M 文件名或一小段 MATLAB 语句。它定义该图形窗口在关闭时, 若用户发出 **close** 命令, 应执行的 **Callback** 调用。但是, **delete** 命令会无条件地关闭图形窗口, 此时这个属性定义的 **Callback** 调用不会被执行。**DeleteFcn** 属性与 **CloseRequestFcn** 属性的作用类似, 所不同的是, 无论用户采用什么样的方式关闭图形窗口, **the DeleteFcn** 属性定义的 **Callback** 都要被执行。

(2) **KeyPressFcn** 属性

KeyPressFcn 属性的取值是字符串, 一般是某个 M 文件名或一小段 MATLAB 语句。它定义了用户在图形窗口中按下某键时所要执行的 **Callback** 调用。例如, 在这个属性的 **Callback** 调用中可以定义一条语句访问 **CurrentCharacter** 属性, 然后判别 **CurrentCharacter** 属性值是否为某个特别的字符, 再决定是否执行某些操作。这种方法可以让用户在图形窗口中按下某个字符键, 完成一定的操作任务的字符时执行。

请看如下的例子:

```
>> h=figure
>> set(h,'KeyPressFcn',[ 'Key=get(h,"CurrentCharacter"),',...
    'if Key=="X",delete(h),end' ])
```

那么, 当在图形窗口 h 中按下 X 键时, 就会关闭该图形窗口。

另一方面, 在 **KeyPressFcn** 属性定义的 **Callback** 中, 可以访问根对象的 **PointerWindow** 属性, 以确定是在哪个图形窗口中按下该键的, 因为鼠标标记落在某个图形

窗口上时,键入一个字符,并不能使这个图形窗口成为当前的图形窗口。KeyPressFcn 属性定义的 Callback 受 Interruptible 属性的影响。

(3) ResizeFcn 属性

ResizenFcn 属性的取值是字符串,一般是某个 M 文件名或一小段 MATLAB 语句。它定义了用户在改变图形窗口大小时所要执行的 Callback 调用。此时,可以访问图形窗口的 Position 属性,以了解图形窗口的大小与位置。如果正在用鼠标改变图形窗口的大小,那么正在运行的程序中就不能直接访问读取这个窗口的句柄,只能从根对象的 Callback Object 属性中访问该图形窗口。一般用函数 gcbh 获取。

(4) WindowButtonUpFcn 属性

WindowButtonUpFcn 属性的取值是字符串,一般是某个 M 文件名或一小段 MATLAB 语句。它定义用户在图形窗口中释放鼠标按键时程序应执行的 Callback 调用。WindowsButtonUpFcn 属性定义的 Callback 受 Interruptible 属性的影响。

(5) WindowButtonDownFcn 属性

WindowButtonDownFcn 属性的取值是字符串,一般是某个 M 文件或一小段 MATLAB 语句。它定义用户在图形窗口中按下鼠标键时程序应执行的 Callback 调用。WindowButtonDownFcn 属性定义的 Callback 受 Interruptible 属性的影响。

(6) WindowButtonMotionFcn 属性

WindowButtonMotionFcn 属性的取值是字符串,一般是某个 M 文件或一小段 MATLAB 语句。它定义用户在图形窗口中移动鼠标时程序应执行的 Callback 调用。WindowsButtonMottonFcn 属性定义的 Callback 受 Interruptible 属性的影响。

值得注意的是,ButtonDown Fcn 属性和 WindowButtonDownFcn 属性都是定义用户在图形窗口中按下鼠标键时所要执行的 Callback。如果某个图形窗口的这两个属性都有定义,那么当在该图形窗口中按下鼠标键时,先执行 WindowButtonDownFcn 定义的 Callback,执行完后,再执行 ButtonDownFcn 属性定义的 Callback。例如,读者可对下列语句的效果进行检验:

```
>> h=figure  
>> set(h,'ButtonDownFcn','disp("Starting ButtonDownFcn Callback.")')  
>> set(h,'WindowButtonDownFcn','disp("Window Button Down Callback.")')  
>> set(h,'WindowButtonUpFcn','dis("Button Up!")')
```

那么,在该图形窗口按下鼠标键,并放开后,命令窗口中显示如下三行文字:

```
Window Button Down Callback.  
Starting ButtonDownFcn Callback.  
Button Up!
```

6. 鼠标标记属性

(1) Pointer 属性

Pointer 属性的取值是 arrow(缺省值)、crosshair、watch、topl、topr、botl、botr、circle、cross、fleur、left、right、top、bottom、fullcrosshair、ibeam、custom,共定义了 17 种鼠标标记,用户可以根据工作任务的类型,设定不同的鼠标标记,其中,custom 表示由用户自定义鼠标

标记。例如,表示程序正在运行时,通常用 watch 标记,这符合 Windows 系统的习惯。读者自己可以看一看每种标记在所用的计算机系统上的形状,例如:

```
>> h = figure
>> set(h,'Pointer','crosshair')
```

运行上述两条语句后,在 Windows 95 上,当把鼠标移入图形窗口时,鼠标箭头会变成十字架的形状。

(2) PointerShapeCData 属性

PointerShapeCData 属性的取值是 16×16 阶矩阵。当 Pointer 属性值为 custom 时, MATLAB 以这个矩阵定义的 16×16 像素位图为用户自定义的鼠标标记。该矩阵各元素之值为 1、2 或 NaN。1 表示像素标记黑色,2 表示像素标记白色,而 NaN 表示像素是透明的。PointerShapeCData 矩阵的(1,1)元素对应它定义的鼠标标记的左上角点,(16,16)元素对应右下角点。PointerShapeCData 属性定义的鼠标标记只在 Pointer 属性取值为 custom 时起作用,改变 Pointer 属性值,不会对 PointerShapeCData 属性值产生影响。

(3) PointerShapeHotSpot 属性

PointerShapeHotSpot 属性的取值为二元素向量,它是 PointerShapeCData 矩阵的某个行列指标,决定了 PointerShapeCData 矩阵中的某个元素,这个元素对应了鼠标图标的一个像素点。此时,鼠标的位置就是这个像素点的位置。鼠标位置记录在 CurrentPoint 属性和根对象 PointerLocation 属性中。PointerShapeHotSpot 属性的缺省值是(1,1),即鼠标位置由鼠标图像的左上角点确定。

7. 图形窗口管理属性

(1) IntegerHandle 属性

IntegerHandle 属性的取值是 on(缺省值)或 off,这个属性值决定了图形窗口对象句柄值的数值类型。一般来讲,图形窗口句柄值总是正整数。当 MATLAB 创建一个新的图形窗口时,总是将最小的可用的正整数作为新图形窗口的句柄值。如果用户删除了某个图形窗口,那么再创建图形窗口时,就可以使用删除的图形窗口句柄值。如果将 IntegerHandle 属性设置为 off,则 MATLAB 在创建新图形窗口对象时,就用一个实型数值表示图形窗口的句柄值。

(2) NextPlot 属性

NextPlot 属性的取值是 add(缺省值)、replace 或 replacechildren,这个属性决定了 MATLAB 用于图形输出的图形窗口。如果当前的图形窗口的 NextPlot 属性值是 add,则表明用当前的图形窗口作图形输出;如果它的取值是 replace,那么除了它的 Position 属性外,所有的属性都设置为缺省值,并且在输出图形之前,删除已存在的所有图形子对象;如果它的取值是 replacechildren,那么只是在新图形输出之前,删除已存在的所有图形子对象,而不改变其他的属性值。例如,请读者逐行演示如下的语句:

```
>> h=figure
>> set(h,'Color','r')
>> subplot(221),plot([1,2])
>> subplot(222),plot([1,2])
```

```
>> set(h,'NextPlot','replace')
>> subplot(223),plot([1,2])
>> set(h,'Color','r','NextPlot','replacechildren')
>> subplot(224),plot([1,2])
```

8. 管理图形打印的属性

(1) InvertHardcopy 属性

InvertHardcopy 属性的取值是 on(缺省值)或 off,这个属性只影响打印输出。当它的属性值是 on 时,图形窗口的背景色等转换成白色,而坐标轴、线段等转换为黑色;而若其属性值为 off,则打印图形窗口时不会进行任何颜色的转换。

(2) PaperOrientation 属性

PaperOrientation 属性的取值是 portrait(缺省值)或 landscape,这个属性决定着图形窗口打印输出时的方向。portrait 是标准的打印方式,而设置 landscape 会使打印的结果按逆时针方向旋转 90°。

(3) PaperPosition 属性

PaperPosition 属性的取值是一个 4 元素向量 [left, bottom, width, height],此向量定义了打印纸上的一个长方形区域,其左下角相对于纸张的左下角的位置,由 (left, bottom) 决定,它的长度单位由 PaperUnits 属性定义。图形窗口的内容按一定的比例缩放打印在这个区域上,所以用户可以设置这个属性值来得到所需要大小的图形输出,但要注意打印的结果还受下面的属性值影响。

(4) PaperPositionMode 属性

PaperPositionMode 属性的取值是 manual(缺省值)或 auto。在 manual 模式下,输出的图形在打印纸上的位置由 PaperPosition 属性决定;而在 auto 模式下,MATLAB 将按照屏幕上图形窗口的尺寸把图形打印在打印纸的中央。

(5) PaperSize 属性

PaperSize 属性的取值是二元素向量 [width, height],它定义打印纸的大小。这个属性只可读,用户不能更改。它的具体值由 PaperType 属性决定,其单位由 PaperUnits 属性定义。

(6) PaperType 属性

PaperType 属性的取值是 usletter(缺省值)、uslegal、a3、a4letter、a5、b4 和 tabloid。它定义了 MATLAB 打印支持的几种标准纸张。如果用户使用 A4 纸,最好将这个属性设置为 a4letter。

(7) PaperUnits 属性

PaperUnits 属性的取值是 normalized(相对单位)、inches(英寸:缺省值)、centimeters(厘米)和 points(磅)。这个属性定义打印纸的度量单位。

3.2.3 属性应用技巧

1. 图形窗口对象的位置

MATLAB 图形窗口在屏幕上的位置由其 Position 属性和 Units 属性共同决定。Units 属性定义长度单位,Position 属性值是一个四元素向量 [left, bottom, width, height], 其中, left 和 bottom 决定图形窗口(内区域)左下角点相对于屏幕左下角点的横纵坐标值, 而 width 和 height 分别决定图形窗口内区域的宽度和高度, 但不包含窗口边界。

由于图形窗口实际上是 Windows 操作系统下的窗口, 所以可以用 Windows 的窗口操作方法对其进行操作, 例如改变窗口的大小和位置, 这时 MATLAB 系统自动更新 Position 属性的值。同时, 还要注意, 对于不同的 Units 属性设置, 同一个图形窗口的 Position 属性值是不同的。

对于不同的显示器, 屏幕的尺寸是不同的, 可以从对象的 ScreenSize 属性得知所使用的屏幕尺寸。例如:

```
>> get(0,'ScreenSize')
ans =
    1    1    1024    768
```

为了写出与设备无关的程序, 可以使用先读出根对象信息的技巧。例如, 如果要在屏幕上定义两个图形窗口, 使得它们并排地占据上 $1/3$ 屏幕, 且图形窗口的边界宽为 5 像素, 而上边界在一般系统上为 30 像素高, 则下面的语句组可以实现这个目的:

```
>> bdwidth = 5;
>> topbdwidth = 30;
>> set(0, 'Units', 'pixels')
>> scnsize = get(0, 'ScreenSize')
>> position1 = [bdwidth, 2/3 * scnsize(4) + bdwidth, ...
    scnsize(3)/2 - 2 * bdwidth, scnsize(4)/3 - (topbdwidth + bdwidth)];
>> position2 = [position1(1) + scnsize(3)/2, ...
    position1(2), position1(3), position1(4)];
>> figure('Position', position1);
>> figure('Position', position2);
```

另一种编写与设备无关程序的方式是利用图形窗口 Units 属性的相对单位 normal。如果要使得创建的图形窗口位于屏幕的正中央, 那么下列语句组可以实现这个目的:

```
>> width = 0.5; % the width of Figure window
>> height = 0.5; % the height of Figure window
>> pos = [0.5 - width/2, 0.5 - height/2, width, height];
>> figure('Units', 'normal', 'position', pos)
```

应该注意的是, 在 figure 语句调用中, 必须最先设置 Units 属性的值, 否则难以保证结果正确。

再看一个例子。我们知道, 图形窗口本身是 Windows 系统的窗口, 所以在该窗口的标题栏中按下鼠标时, 可以将该图形窗口移动到任意的位置。能否在图形窗口内按下鼠标并移动鼠标时也能拖动图形窗口呢? 下面是实现该操作的最简单的实例:

```
>> h=figure
```

```

>> set(h,'WindowButtonDownFcn','pos1=get(h,"CurrentPoint");'
>> set(h,'WindowButtonMotionFcn',[ 'pos2=get(h,"CurrentPoint");',...
'pos3=get(h,"position");',...
'set(h,"position",[pos3(1)+pos2(1)-pos1(1),pos3(2)+pos2(2),...
'-pos1(2),pos3(3),pos3(4)']);',...
'pos1=pos2;'])

```

2. 色彩控制原理

MATLAB 5.0 定义了 6 个图形窗口属性来控制图形窗口使用计算机系统颜色资源的模式和对输出的图形图像着色的方法。这 6 个属性分别是, Colormap, FixedColors, MinColormap, ShareColor, Dithermap, DithermapMode 属性等。这些属性是如何控制 MATLAB 工作的呢? 下面对其原理作简单的描述。

对每个图形窗口对象, MATLAB 定义了唯一的色谱(由 Colormap 属性定义)和一个固定颜色色谱(由 FixedColors 属性定义), 它们互不相干且作用不同。当一个图形窗口对象成为活跃窗口时, 计算机操作系统将这个图形窗口的 Colormap 和 FixedColors 装入系统的颜色表中。在索引式颜色系统上, 屏幕上的每个像素对应于颜色表中的一项。颜色表的内容大体上包括三个部分: 系统占用的颜色, 图形窗口的 FixedColors 和 Colormap, 而且优先装入 FixedColors。MATLAB 用固定颜色色谱定义的颜色对线段型对象、坐标轴、字符等着色。由于 FixedColors 占据系统颜色表, 所以要控制固定颜色的数量。如果系统颜色表不能完全装入全部 Colormap, 那么 MATLAB 系统便按线性方式从 Colormap 中装入部分颜色到系统颜色表中。另一方面, MATLAB 提供了 MinColormap 属性, 要求系统在颜色表中至少保留 MinColormap 项, 用于装入 MATLAB 图形窗口 Colormap 定义的颜色。所以, 要使得操作系统装入全部 Colormap, 必须设置 MinColormap 的值不小于 Colormap 色谱矩阵的长度。值得注意的是, 如果 MinColormap 属性值大于系统颜色表可用的项数, MATLAB 就会占用系统的颜色资源, 对装入的 FixedColormap 色谱不会发生影响, 在这种情况下, 某些非活跃的窗口的颜色会发生变化。

为了节省操作系统有限的颜色资源, MATLAB 提供了一种工作机制。如果有多个定义了较大 Colormap 的图形窗口同时存在, 那么操作系统就难以保证每个图形窗口中显示的图形图像着色是正确的。但是当图形窗口 ShareColors 属性取值为 yes 时, 对于系统颜色表中已存在的颜色, 图形窗口不再在该表中重新定义, 而直接利用表中这一项。遗憾的是, 这种工作方式虽然节省了颜色资源, 但是却不能让 MATLAB 系统很快地更换色谱。例如: gatlin. mat 和 clown. mat 是两幅图像数据。前者为黑白, 后者为彩色。请看下列语句产生的效果:

```

>> load gatlin
>> subplot(211),image(X)
>> colormap(map)
>> load clown
>> subplot(212),image(X)

```

此时, 两幅图像都是黑白的。因为 clown 图像共享 gatlin 图像的黑白色谱, 已由

colormap(map)装入系统颜色表中。如果再用 $\gg colormap(map)$ 装入 clown 的正确色谱，虽然 clown 的显示颜色是正确的，但是 gatlin 却变成了伪色图。在这种情况下，唯一的解决办法是将该图形窗口的 ShareColors 属性重新定义为 no。

请读者再注意下列语句的作用效果：

```

>> load gatlin
>> subplot(211),image(X)
>> colormap(map)      %装入 gatlin 的黑白色谱
>> set(gcf,'ShareColors','no')    %改变 ShareColors 的缺省状态 yes
>> load clown
>> colormap(map)      %装入 clown 的彩色色谱
                           %此时对 gatlin 图像的黑白颜色无影响
>> subplot(212),image(X)

```

我们注意到，在同一个图形窗口中同时显示了黑白图像和彩色图像。这个例子说明了 ShareColors 属性的作用，这就是说，如果 ShareColors 的属性值为 no(或 off)，那么图形窗口已存在的对象的着色不受随后色谱改变的影响，除非 ShareColors 重新设置为 yes(或 on)，此时按图形窗口的当前色谱对所有图形对象重新着色。

MATLAB 5.0 支持全真颜色 (True Colors)，但是在每一个索引式颜色系统上，MATLAB 的工作过程是将每全真颜色的 RGB 值映射到 Dithermap 色谱中最相近的颜色，使用 Floyd Steinberg 映射算法。Dithermap 是图形窗口定义的一个色谱，在全真颜色的情况下，可用这个色谱代替 Colormap 定义的色谱。一般情况下，MATLAB 自动生成 Dithermap，但是这个过程很费时。通常的方法是，先让 MATLAB 自动生成 Dithermap，再设置 DithermapMode 属性为 manual，并将已生成的 Dithermap 保存下来，供下次使用，例如：

```

>> set(gcf, 'DithermapMode','auto')
>> dmap = get(gcf, 'Dithermap')
>> set(gcf, 'DithermapMode','manual')
>> save DithMaps dmap

```

3. 图形窗口的鼠标图标

MATLAB 预定义了 15 种鼠标图标，另外，还提供了一种用户自定义模式，即

```
>> set.figure_handle, 'pointer','custom')
```

这时，由 PointerShapeCData 属性和 PointerShapeHotSpot 属性共同来定义鼠标图标和鼠标的定位点。PointerShapeCData 的取值是 16×16 矩阵，其元素取值可以是 1(表示黑色)、2(表示白色)或 NaN(表示透明)。例如：

```

>> P = ones(16)+1;
>> P(1, :) = 1;
>> P(16, :) = 1;
>> P(:, 1) = 1;
>> P(:, 16) = 1;

```

```

>> P(1 : 14,8 : 9) = 1;
>> P(13 : 16,8 : 9) = 1;
>> P(8 : 9,1 : 4) = 1;
>> P(8 : 9, 13 : 16) = 1;
>> P(5 : 12, 5 : 12) = NaN;
>> set(gcf,'Pointer','custom','PointerShapeCData', P, ...
    'PointerShapeHotSpot', [9,9])

```

还可以用数学函数定义 PointerShapeCData，例如

```

>> g = linspace(0,20,16);
>> [X,Y] = meshgrid(g);
>> Z = 2 * sin(sqrt(X.^2+Y.^2));
>> set(gcf,'Pointer','custom','PointerShapeCData', (Z>0)+1)

```

3.3 坐标系对象

坐标系对象是 MATLAB 系统管理的另一类很重要的图形对象。坐标系对象是图形窗口对象的子对象，每个图形窗口对象中可以定义多个坐标系对象，但是只有一个坐标系是当前坐标系，在没有指明坐标系时，所有的图形生成函数都是在当前坐标系中输出图形或图像。必须弄清楚一个概念，所谓在某个图形窗口中输出图形图像，是指在该图形窗口的当前坐标系中输出图形图像。

3.3.1 坐标系对象生成函数

通常，在图形窗口中没有定义坐标系之前，MATLAB 的每个图形生成函数会自动地创建一个坐标系作为它输出的图形图像的坐标系。用户可以根据自己需要利用坐标系创建函数创建特殊的坐标系。坐标系创建函数是 axes，它有如下多种命令形式：

```

>> axes
>> h = axes
>> axes('坐标系属性名', 坐标系属性值,...)
>> h = axes('坐标系属性名', 坐标系属性值,...)
>> axes(h)

```

坐标系生成函数 axes 是系统低层生成函数，上述第一和第二条命令用系统坐标系缺省属性值在当前图形窗口中生成一个坐标系，并将该坐标系句柄值存在变量 h 中；第三和第四条命令根据指定的“属性名/属性值”对在当前图形窗口中创建坐标系，并且将该坐标系的句柄值存在变量 h 中；第五条命令使句柄值为 h 的坐标系成为当前的坐标系，且该坐标系所在的图形窗口自动成为当前的图形窗口。在这种命令形式中，h 必须是一个合法坐标系句柄，即这个坐标系是存在的，这与图形窗口创建函数 figure(h) 的调用是不同的。如果 h 对应的图形窗口不存在，那么 figure(h) 就创建一个新的图形窗口对象，并且以 h 为其句柄值。

可以用函数 `gca` 获得当前图形窗口的当前坐标系的句柄值, 即当前图形窗口对象 `CurrentAxes` 属性值。如果当前图形窗口中没有坐标系子对象, 或没有图形窗口对象存在, 那么 `gca` 将在当前图形窗口中创建一个坐标系, 或同时创建一个图形窗口对象, 这等价于函数 `axes` 调用。

3.3.2 坐标系对象属性

MATLAB 为每个坐标系对象定义了 76 种属性, 它们共同控制着坐标系的外形、方位和大小等。坐标系对象的部分属性分述如下, 其公共属性参见第 3.1.3 节。

1. 外观特征属性

(1) Box 属性

`Box` 属性的取值是 `on` 或 `off`(缺省值), 它决定坐标系是否带有边框。在 2 维情形, 坐标系绘制成矩形框, 图形在框内输出; 在 3 维情形, 坐标系绘制成长方体, 图形在体内输出。大部分的 2 维图形函数, 如 `plot`、`contour` 等在绘图时, 自动为图形加上边框。

(2) GridLineStyle 属性

`GridLineStyle` 属性的取值是 `-`、`--`、`:`(缺省值)、`.` 或 `none`, 该属性定义坐标系网格线的线型。“`-`”表示实线, “`--`”表示虚线, “`:`”表示点线, “`.`”表示点虚线。命令

```
>> grid on
```

表示在当前坐标系上画网格线, 而命令

```
>> grid off
```

则表示将当前坐标系的网格线取消。

(3) Layer 属性

`Layer` 属性的取值是 `bottom`(缺省值)或 `top`, 对于 2 维坐标系, 或视点 `[0 90]`, 当该属性值为 `top` 时, 坐标轴刻度标记、网格线等画在坐标系图形子对象之上(即可见的); 而当该属性值为 `bottom` 时, 坐标轴刻度、网格线等在图形子对象之下。例如, 执行下列语句

```
>> pcolor([1 2 3; 4 5 6; 1 2 3])
```

```
>> grid on
```

后的图形如图 3-2(a) 所示, 而执行下列语句

```
>> pcolor([1 2 3; 4 5 6; 1 2 3])
```

```
>> grid on
```

```
>> set(gca,'Layer','top')
```

后的图形如图 3-2(b) 所示。

(4) LineStyleOrder 属性

`LineStyleOrder` 属性的取值为线段类型值, 其缺省值为 `-`(实线), 表示对每条线段型图形使用实线作图。该属性决定在生成多条线段型图形时使用的线段类型和顶点标记及其顺序。例如:

```
>> set(gca,'LineStyleOrder',' -x | :|o')
```

表示顺序地使用实线加 `x` 顶点标记、点线、圆圈顶点标记作线段型图形, 也可以用块数组设

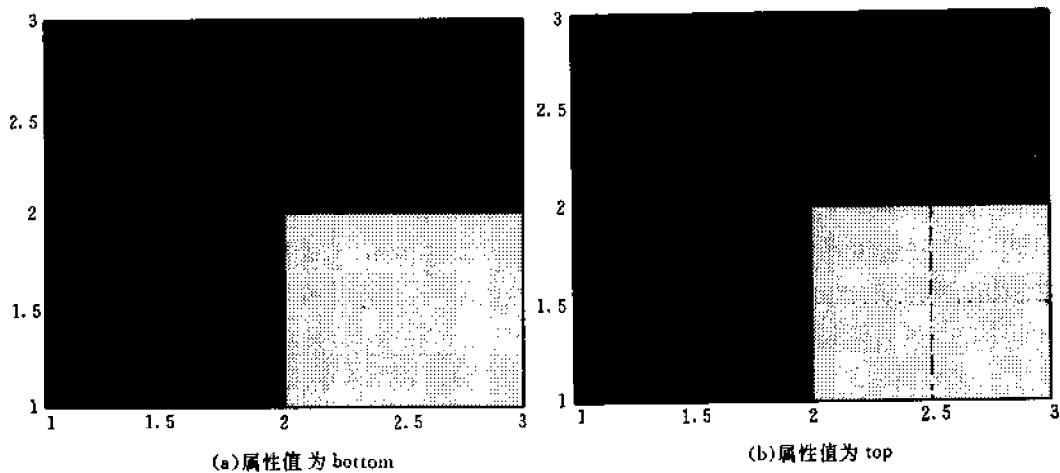


图 3-2 Layer 属性的作用

置 LineStyleOrder 属性值,例如,上述语句等价于

```
>> set(gca,'LineStyleOrder','-x',':', 'o')
```

MATLAB 自定义了 4 种线段类型,在作图时,依次使用这 4 种类型的线段作图,而每种线段类型重复地使用 ColorOrder 属性定义的不同颜色,直到所有颜色都使用过一次后,再换下一种线段类型。例如,假设 ColorOrder 属性定义了 7 种颜色,那么作图时,用第一种线段类型作前 7 条线段图形,但是使用 7 种不同的颜色着色,如此循环。

(5) LineWidth 属性

LineWidth 属性的取值是以磅为单位的宽度值,缺省值为 0.5 磅,它定义坐标轴线段的宽度。

(6) TickDir、TickDirMode 属性

TickDir 属性的取值是 in 或 out,定义坐标轴刻度标记的方向。在缺省模式下,对 2 维坐标系,坐标轴刻度标记向坐标系内;对于 3 维坐标系,坐标轴刻度标记向坐标系外。

TickDirMode 属性的取值是 auto(缺省值)或 manual,定义 MATLAB 选择坐标轴刻度方向的方式。在 auto 模式下,MATLAB 总是将 2 维坐标系的坐标轴刻度方向置为 in,而把 3 维坐标系置为 out。但是,如果用户设置了 TickDir 的值,MATLAB 会自动地将 TickDirMode 属性设置为 manual。在 manual 模式下,MATLAB 不改变特定的坐标轴刻度方向。

(7) TickLength 属性

TickLength 属性的取值是二元素向量。第一个元素定义 2 维视点时坐标系刻度线段的长度,第二个元素定义 3 维视点时坐标系刻度线段的长度,其值是相对于最长坐标轴的相对单位值,缺省值为 [0.010 0.025]。

(8) CurrentPoint 属性

CurrentPoint 属性的取值为 2×3 的数值矩阵,它定义最后一次按下鼠标键的位置,由坐标系数据单位决定其数值单位。这个矩阵记录着鼠标决定的两个顶点的 3 维坐标值,这两个顶点位于垂直于屏幕并通过鼠标顶点的直线上,是该直线与 3 维坐标系的 x 轴、y 轴、z 轴

的 Limits 属性决定的矩形体边界面的交点。其中一个交点位于前端,另一个交点位于后端。数值矩阵的第一行为后端交点的坐标系值,第二行为前端交点的坐标系值。对于 2 维坐标系,前端交点的 z 坐标值为 1,后端交点的 z 坐标值为 0,而它们的 x 和 y 坐标值是相同的。

当在图形窗口中点按鼠标键时, MATLAB 就更新该图形窗口中所有坐标系的 CurrentPoint 属性值。

(9) Position 属性

Position 属性的取值为 4 元素的数值向量 [left, bottom, width, height]。这个向量在图形窗口中决定一个矩形区域,坐标系就位于这区域中,或者说这个向量确定坐标系在图形窗口中的位置。该矩形的左下角相对于图形窗口左下角的坐标为 (left, bottom), 矩形的宽度为 width, 高度为 height, 其数值单位由坐标系 Units 属性值定义。在通常情况下, 坐标轴标记、坐标轴说明文字等都落在由 Position 定义的矩形区域之外, 即坐标系撑满模式。但是如果坐标系不取撑满模式,那么坐标系及坐标轴标记、坐标轴说明文字等都落在 Position 决定的区域内。

(10) Units 属性

Units 属性的取值是 inches (英寸)、centimeters(厘米)、normalized (相对单位: 缺省值)或 points (磅)。这个属性值决定 Position 属性的度量单位。所有的度量都是从图形窗口左下角 (0,0) 开始, 而相对单位 normalized 将图形窗口左下角对应为 (0,0), 右上角对应为 (1,1), 其他都是绝对单位。

2. 辅助属性

(1) FontAngle 属性

FontAngle 属性的取值是 normal(缺省值)、italic 或 oblique, 它定义坐标系的坐标轴刻度值、坐标轴说明文字等所用的字体。normal 表示使用正体, italic 和 oblique 都表示使用斜体。

(2) FontName 属性

FontName 属性的取值为字符串, 它定义坐标系的坐标轴刻度值、坐标轴说明文字等所用的字体。它的合法值与计算机系统上使用的字体资源有关, 通常的缺省字体是方头字体。在改变这个属性时, 坐标系刻度值字体立即更新, 但是坐标系说明文字的字体必须通过重新设置 xlabel、ylabel 和 zlabel 属性或重新用 xlabel、ylabel 和 zlabel 命令来更新。

(3)FontSize 属性

FontSize 属性的取值是一个数值, 它定义坐标系坐标轴刻度数值、坐标轴说明文字以及坐标系标题 (Title) 文字的大小尺寸, 其尺寸单位由坐标系 FontUnits 属性决定, 缺省值为 12 磅。当改变 FontSize 属性值后, 坐标轴刻度数值字体大小立刻改变, 但是必须重新分别设置 xlabel、ylabel、zlabel 属性或调用命令 xlabel、ylabel、zlabel 才能改变坐标轴说明文字的字体大小。

(4) FontUnits 属性

FontUnits 属性的取值可以是 points(磅: 缺省值)、normalized(相对单位)、inches(英寸)、centimeters(厘米)或 pixels(像素)。MATLAB 5.0 以前的版本没有这个属性, 字体大小只以磅为单位度量。FontUnits 属性值 normalized 是相对于坐标系高度的相对单位, 例

如,FontSize 属性值为 0.1 时,表示字符的高度为坐标系高度的十分之一。

(5) FontWeight 属性

FontWeight 属性的取值是 normal(缺省值)、bold、light 或 demi。该属性决定坐标系使用文字字体的粗细类型。通常, bold 表示某种字体的黑体类型, light 表示相应的细体类型。与 FontSize 属性一样, 更新该属性值时, 坐标轴刻度数值说明文字立即改变, 但是必须重新分别设置 XLabel、YLabel、ZLabel 属性或调用命令 xlabel、ylabel、zlabel 才能改变坐标轴说明文字的字体形态。

(6) Title 属性

Title 属性的取值为坐标系标题文字对象的句柄值, 可以通过该属性对坐标系标题文字对象进行操作。例如, 要改变标题的颜色, 可执行语句

```
>> set(gcf,'Title','Color','g')
```

而 MATLAB 函数 title 直接对坐标系标题文字对象操作。

(7) XLabel、YLabel、ZLabel 属性

XLabel、YLabel、ZLabel3 种属性的取值分别是 x 轴、y 轴、z 轴说明文字对象的句柄值, 可以通过这些句柄值对坐标轴说明文字进行操作或将某个文字对象指定为坐标轴说明文字。例如, 语句

```
>> set(gca,'XLabel',text('String','Axis Label'))
```

与下面的命令

```
>> xlabel('Axis Label')
```

作用完全相同。坐标系上任何文字对象的句柄值可以赋给 XLabel、YLabel、ZLabel, 那么该文字对象就被移到相对坐标轴说明文字的位置上, 例如在 [0 1 0 1] 的坐标系上, 下面的语句

```
>> h = text(0.5, 0.5, 'LABELS')
```

在坐标中央定义了文字对象“LABELS”, 则语句

```
>> set(gca, 'XLabel', h)
```

会把文字“LABELS”移到 x 坐标轴的说明文字的位置上。

(8) XTickLabels、YTickLabels、ZTickLabels 属性

XTickLabels、YTickLabels、ZTickLabels3 种属性的取值都是字符串矩阵或字符串块数组, 它定义了各坐标轴刻度上的标记, 矩阵的每行文字对应一个刻度。如果矩阵行不足以对应所有坐标轴刻度, 矩阵行就被循环使用。例如, 下列语句

```
>> gca  
>> set(gca,'xticklabels','one|two|three')
```

将 x 轴刻度依次标记为 one、two、three、one、two、three……

在 MATLAB 5.0 中, XTickLabels、YTickLabels、ZTickLabels 属性接受多种形式的输入, 在用函数 set 设置这些属性时, 其值可以是字符串矩阵、字符串块数组、由竖线分隔的字符串或数值向量(其中数值被转化为数字字符串)。例如, 下列各条语句是等价的:

```
>> set(gca, 'XTickLabels','1','50','100')  
>> set(gca, 'XTickLabels','1|50|100')  
>> set(gca, 'XTickLabels',[1;50;100])
```

```
>> set(gca, 'XtickLabels', ['1'; '50'; '100'])
```

(9) XTickLabelMode、YTickLabelMode、ZTickLabelMode 属性

XTickLabelMode、YTickLabelMode、ZTickLabelMode 3 种属性的取值都是 auto(缺省值)或 manual, 它们分别记录着各坐标轴刻度标记的生成方式。在 auto 模式下, MATLAB 用数值作刻度标记, 而 manual 模式是指用户自定义的刻度标记, 如文字标记等等。当用户用 set 函数分别设置 XTickLabels、YTickLabels、ZTickLabels 属性时, MATLAB 分别自动地将 XTickLabelMode、YTickLabelMode、ZTickLabelMode 属性置为 manual。

3. 控制坐标系的属性

(1) XDir、YDir、ZDir 属性

XDir、YDir、ZDir 3 种属性的取值都是 normal(缺省值)或 reverse, 这些属性决定着各坐标轴的方向。在缺省设定下, 取右手坐标系, x 轴从左到右为正向, XDir 属性值 reverse 则将这个方向反过来; y 轴从下到上(2 维视点)或由前向后(3 维视点)为正向, YDir 属性值 reverse 将这个方向反过来; z 轴从屏幕向外(2 维视点)或从下到上(3 维视点)为正向, ZDir 属性值 reverse 将这个方向反过来。

(2) XGrid、YGrid、ZGrid 属性

XGrid、YGrid、ZGrid 3 种属性的取值都是 on 或 off(缺省值)。当属性值分别为 on 时, 在坐标系上相应的坐标轴刻度处作垂直于该坐标轴的网格线, 函数 grid on 或 grid off 同时对 3 条坐标轴起作用。

(3) XLim、YLim、ZLim 属性

XLim、YLim、ZLim 3 种属性的取值都是 2 元素的数值向量, 它们分别定义各坐标轴的可见范围, 缺省值为 [0, 1]。改变该向量, 相当于改变坐标轴的比例。

(4) XLimMode、YLimMode、ZLimMode 属性

XLimMode、YLimMode、ZLimMode 3 种属性的取值都是 auto(缺省值)或 manual。这些属性记录着 MATLAB 设定坐标轴范围向量的方式。在 auto 模式下, MATLAB 根据坐标系中图形的数据自动计算范围向量, 保证所有图形子对象都落在坐标系中。用户可以用 set 函数设置 XLim、YLim、ZLim 属性值, 此时 MATLAB 将 XLimMode、YLimMode、ZLimMode 属性置为 manual。

(5) XScale、YScale、ZScale 属性

XScale、YScale、ZScale 3 种属性的取值都是 linear(缺省值)或 log, 这些属性定义各坐标轴的比例尺度。MATLAB 定义了两种比例尺度, 一种为 linear(线性)坐标尺度, 另一种为 log(对数)坐标尺度。

(6) XTick、YTick、ZTick 属性

XTick、YTick、ZTick 3 种属性的取值都是数值向量, 这些向量记录着各坐标轴刻度位置处的坐标值。每个向量的元素必须是单调增加的, 用户可以设置这些向量, 可以将坐标轴刻度标记在所需的地方。如果不希望坐标轴上有任何刻度, 则将这些向量设置为空向量即可。

(7) XTickMode、YTickMode、ZTickMode 属性

XTickMode、YTickMode、ZTickMode 3 种属性的取值都是 auto(缺省值)或 manual。记

录坐标轴刻度的生成方式。在 auto 模式下, MATLAB 根据各坐标轴的数据范围计算标记刻度的坐标值;如果用户自己定义 XTick、YTick、ZTick 属性,那么 MATLAB 自动地将 XTickMode、YTickMode、ZTickMode 属性置为 manual。

(8) XAxisLocation 属性

XAxisLocation 属性的取值是 top 或 bottom(缺省值),该属性决定 x 轴刻度和说明文字的位置。当它的取值是 bottom 时,就是通常的坐标系;如果它的取值是 top,那么 x 轴就被移到坐标系区域的上方。

(9) YAxisLocation 属性

YAxisLocation 属性的取值是 right 或 left(缺省值),该属性决定 y 轴刻度和说明文字的位置。当它的取值是 left 时,就是通常的坐标系;如果它的取值是 right,那么 y 轴就被移到坐标系区域的右边。

4. 视点属性

在 MATLAB 5.0 以前的版本中,坐标系的视点由 View 属性控制,这个属性只定义了视点的方位,用户不能控制视点与目标点的距离。但实际上,即使方位相同,如果视点离坐标系的距离不同,看到的坐标系的效果也应该是不一样的。为了更好地控制视点,根据照相机原理,MATLAB 5.0 提供了 8 种新属性来控制坐标系的视点位置。

(1) CameraPosition、CameraPositionMode 属性

CameraPosition 属性的取值是 3 元素数值向量 [x,y,z],该属性定义视点位置的坐标值,可以将其比作照相机从该处取景。假设照相机的视角(由 CameraViewAngle 定义)保持不变,那么改变 CameraPosition 就可以变化取景范围,当视点向目标点(由 CameraTarget 定义)靠近时,目标被放大;远离目标点时,目标将缩小。

CameraPositionMode 属性的取值是 auto(缺省值)或 manual。当它的取值是 auto 时,MATLAB 根据 View 属性定义的视点方向,自动计算 CameraPosition 属性值,使得从 CameraPosition 到 CameraTarget 保持固定的距离。如果由用户设置 CameraPosition 属性值,则 MATLAB 将 CameraPositionMode 置为 manual。

(2) CameraTarget、CameraTargetMode 属性

CameraTarget 属性的取值是 3 元素数值向量 [x,y,z],该属性定义目标点位置的坐标值,或说是照相机的对焦点。显然,视点方向在 CameraPosition 和 CameraTarget 两点决定的直线上。

CameraTargetMode 属性的取值是 auto(缺省值)或 manual。当它的取值是 auto 时,MATLAB 自动地将 CameraTarget 点固定在坐标系绘图区域的重心点处。如果用户自定义 CameraTarget 属性值,那么 MATLAB 就置 CameraTargetMode 为 manual。

(3) CameraUpVector、CameraUpVectorMode 属性

CameraUpVector 属性的取值为 3 元素数值向量 [x,y,z],该属性定义坐标系中的自由向量,决定照像机取景时的上方方向。例如,如果该向量为 [0 1 0],表示 y 轴的正方向为照相机取景的上方方向,很明显,设置不同的 CameraUpVector 向量,可以让照相机绕由 CameraTarget 与 CameraPosition 决定的直线旋转。它的缺省值为 [0 0 1],即 z 轴正向为视点上方方向。

CameraUpVectorMode 属性的取值是 auto(缺省值)或 manual。当它的取值是 auto 时, MATLAB 用 [0 0 1] 作为 3 维视点的上方方向, 而用 [0 1 0] 作为 2 维视点的上方方向。当用户自定义 CameraUpVector 属性值时, MATLAB 将 CameraUpVectorMode 置为 manual。

(4) CameraViewAngle、CameraViewAngleMode 属性

CameraViewAngle 属性的取值是范围为 $0^\circ \sim 180^\circ$, 该属性定义视点或照相机的视角。它的取值影响坐标系中图形对象的大小, 值越大, 取景范围越大, 那么坐标系中的同一图形就越小。

CameraViewAngleMode 属性的取值是 auto(缺省值)或 manual。当它的取值是 auto 时, MATLAB 将 CameraViewAngle 设置为能使照相机取到坐标系全景的最小视角。当用户自定义 CameraViewAngle 属性值时, MATLAB 将 CameraViewAngleMode 置为 manual。

(5) View 属性

View 属性的取值是 2 元素数值向量 [x, y], 它定义视点方向, 即视点方向的经度和纬度。在 MATLAB5.0 中, 可以用上述的 8 种照相机属性代替 View 的作用。

5. 比例属性

(1) DataAspectRatio、DataAspectRatioMode 属性

DataAspectRatio 属性的取值是 3 元素数值向量 [dx, dy, dz], 该向量控制 x、y、z 轴单位长度的相对比例。例如, 如果该属性值为 [1 2 3], 则表示坐标系 x 轴上的数据单位长度等于 2 倍的 y 轴数据单位长度, 等于 3 倍的 z 轴数据单位长度。

DataAspectRatioMode 属性的取值是 auto(缺省值)或 manual。当它的取值是 auto 时, MATLAB 自动设置 DataAspectRatio 属性值; 如果用户自定义 DataAspectRatio 属性值, 则 MATLAB 置 DataAspectRatioMode 属性值为 manual。

(2) PlotBoxAspectRatio、PlotBoxAspectRatioMode 属性

PlotBoxAspectRatio 属性的取值是 3 元素数值向量 [px, py, pz], 该向量控制坐标系长方体区域在 x、y、z 方向的相对比例。例如, 当它的取值为 [1 1 1] 时, 坐标系呈正方体形式。

PlotBoxAspectRatioMode 属性的取值是 auto(缺省值)或 manual。当它的取值是 auto 时, MATLAB 自动设置 PlotBoxAspectRatio 属性值; 如果用户自定义 PlotBoxAspectRatio 属性值, 则 MATLAB 置 PlotBoxAspectRatioMode 属性值为 manual。

(3) Projection 属性

Projection 属性的取值为 orthographic(缺省值)或 perspective, 它决定坐标系在平面(如计算机屏幕平面)上的投影方式。对观察者来说, 在 orthographic 模式下, 是保持图形的相对宽度, 例如数据决定的平行线在屏幕上画为平行线; 在 perspective 模式下, 3 维图形投影为 2 维图形时, 则保持景深效应, 即图形离观察者越远, 看上去越小。

6. 图形刷新模式

(1) DrawMode 属性

DrawMode 属性的取值是 normal(缺省值)或 fast。当坐标系所在的图形窗口 Renderer 属性值为 painters 时, 该属性控制 MATLAB 对坐标系中图形进行刷新的方式。在 normal 模式下, MATLAB 根据当前的视点方向, 按从后向前的顺序刷新或输出图形, 体现出图形

在 3 维空间中的前后层次关系;而在 fast 模式下, MATLAB 按绘图命令的顺序刷新或输出图形,而不考虑图形对象在 3 维空间中的顺序关系,在这种模式下输出的图形没有图形的物理层次。例如,同一数学曲面 peaks 在两种刷新模式下的图形如图 3-3 所示。

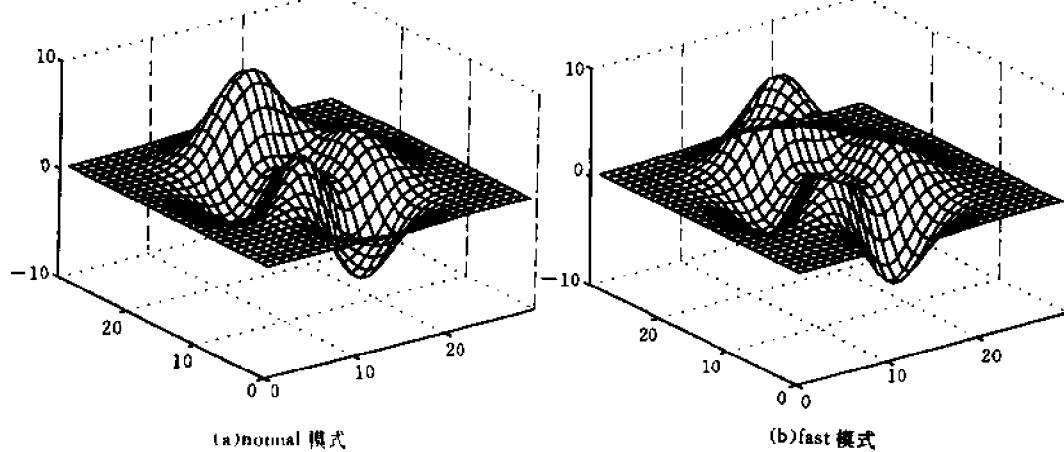


图 3-3 DrawMode 属性的作用

(2) NextPlot 属性

NextPlot 属性的取值是 add、replace(缺省值)或 replacechildren。这个属性决定了 MATLAB 的高层图形函数图形输出的方式。如果当前坐标系的 NextPlot 属性值是 add, 则表明作图形输出时, 保持原有的子图像不变, 加入新的图形; 如果它的取值是 replace, 则除了坐标系的 Position 属性外, 所有的属性都设置为缺省值, 并且在输出图形之前, 删除已存在的所有图形子对象; 如果它的取值是 replacechildren, 那么只是在新图形输出之前, 删除坐标系中已存在的所有图形子对象, 而不改变坐标系其他的属性值。参见图形窗口对象的 Nextplot 属性。

7. 颜色属性

(1) AmbientLightColor 属性

AmbientLightColor 属性的取值为 MATLAB 系统定义颜色的字符或 RGB 颜色值, 该属性定义坐标系中环境照明光的颜色。环境照明光是无方向的光源, 它均匀地散射在坐标系中的图形上。但是如果在坐标系中没有定义光源对象, 则该属性不起作用。如果有光源对象存在, 那么 AmbientLightColor 的作用就加到光照效应中去。

(2) CLim、CLimMode 属性

CLim 属性的取值是 2 元素数值向量 [cmin, cmax], 它决定着 MATLAB 将曲面对象和区域片对象的 CData 数据映射到其图形窗口 Colormap 色谱矩阵行索引号的方式。cmin 值映射到色谱矩阵的第一行, cmax 值映射到色谱矩阵的最后一行, 介于 cmin 与 cmax 之间的数值按线性方式映射到色谱矩阵的各行。在 CData 中超出 [cmin, cmax] 范围的数值, 则不对应任何颜色。

CLimMode 属性的取值是 auto(缺省值)或 manual, MATLAB 取 CData 数据中的最小值为 cmin, 最大值为 cmax。当用户自己设置 CLim 属性值时, MATLAB 将 CLimMode 属性设为 manual。

性置为 manual。在 manual 模式下,CDATA 数据发生变化,对 CLim 属性没有影响。

(3)Color 属性

Color 属性的取值为 none(缺省值),或由 MATLAB 系统定义颜色的字符或 RGB 颜色值,它定义坐标系区域背景颜色。当它的值为 none 时,坐标系区域是透明的,即坐标系区域的颜色为图形窗口的背景色。

(4)ColorOrder 属性

ColorOrder 属性的取值是 $m \times 3$ 的 RGB 颜色值矩阵,其缺省值是系统预定义的 7 种颜色。它定义在坐标系中用 plot、plot3 等函数同时绘制多条曲线时,曲线着色的颜色顺序。当坐标系 NextPlot 属性值为 replace(缺省值),函数 plot 在决定要使用的颜色时,将 ColorOrder 设置为系统的缺省值。如果希望使用自定义的颜色顺序,则要将坐标系的 NextPlot 属性设置为 replacedata。

(5)XColor、YColor、ZColor 属性

XColor、YColor、ZColor 3 种属性的取值都是 MATLAB 系统定义颜色的字符或 RGB 颜色值,它们的缺省值都是 w(白色)。它们分别定义坐标系的坐标轴、坐标轴刻度、坐标轴刻度标记以及坐标系网格线的颜色。

3.3.3 属性应用技巧

1. 添加文字对象技巧

坐标系是输出图形的地方,除了可以在坐标轴和坐标系标题栏给出图形说明文字外,MATLAB 提供了两种方式,可在坐标系的任何位置上加入说明文字。如果掌握好这种技巧,就能设计出图文并茂的图形输出。下面通过几个例子来说明添加文字对象的一些技巧。

请看下面的子程序:

```

>> mesh_handle = mesh(peaks(30));
>> x = get(mesh_handle, 'XData');
>> y = get(mesh_handle, 'YData');
>> z = get(mesh_handle, 'ZData');
>> minz = min(min(z));
>> maxz = max(max(z));
>> set(gca,'fontsize',11,'xlim',[0 30],'ylim',[0 30])
>> [i,j] = find(minz == z);
>> h1=text(x(j),y(i),z(i,j),['The Minimum Value Is: ',...
    num2str(minz)],'VerticalAlignment','top',...
    'HorizontalAlignment','right');
>> set(h1,'fontsize',11)
>> [i,j] = find(maxz == z);
>> h2=text(x(j),y(i),z(i,j),['The Maximum Value Is: ',...
    num2str(maxz)],'VerticalAlignment','bottom')
```

```
>> set(h2,'fontsize',11)
>> grid on
```

由这段子程序生成的图形如图 3-4 所示。

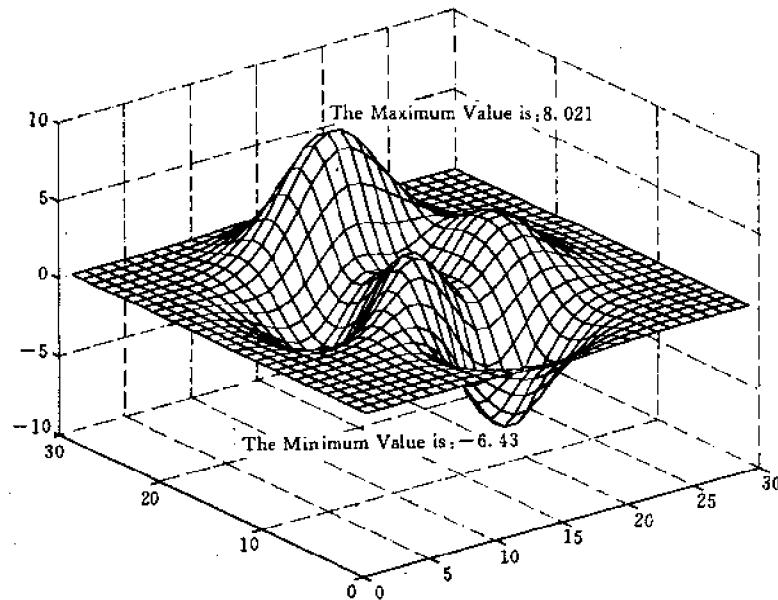


图 3-4 在最高点与最低点处加说明文字

这段子程序的第一条语句在当前坐标系中创建曲面对象,第二至四条语句取得曲面对象的网格数据,第五和第六条语句求出曲面上最高点与最低点的 z 轴坐标值,第八条语句求出最小值点数据矩阵的行列指标,第九条语句在最小值点处写说明文字,并将说明文字向左偏移,同理,第十与十一条语句的作用类似。

一般, text 函数根据指定的坐标值将说明文字置于坐标系内,为了在某个坐标系外写出说明文字,除了直接按超出坐标系 XLim、YLim、ZLim 属性值的坐标值调用函数 text 外,也可以运用多坐标系统技巧。例如,可以创建一个坐标系,占据整个图形窗口,再定义较小的坐标系。如果位置恰当,在大的坐标系中写出的说明文字,看上去就像是贴在小坐标系之外一样。这样,可以达到在另一个坐标系区域外写说明文字的效果。例如:

```
>> h = axes('Position',[0 0 1 1],'Visibility','off');
>> axes('Position',[0.25 0.1 0.7 0.8]);
>> t = 0 : 900;
>> plot(t,0.25 * exp(-0.005 * t))
```

第一条语句创建的坐标系充满整个图形窗口,并且是不可见的;第二条语句再创建一个坐标系;第四条语句在第二个坐标系输出图形。可以在第一个坐标系中写如下说明文字:

```
>> str(1) = 'Plot of the function:';
>> str(2) = 'y = A * e^(-alpha * t)';
>> str(3) = 'With the values:';
```

```

>> str(3) = ' A = 0.25';
>> str(4) = ' alpha = 0.005';
>> str(5) = ' t = 0 : 900 $';
>> set(gcf,'currentaxes',h)
>> text(0.025,0.6,str,'FontSize',12)

```

这样,在小坐标系的左侧,就写上了几行说明文字。

2. 坐标轴刻度控制技巧

对于坐标系中每条坐标轴上的刻度,MATLAB会根据图形数据范围生成等距的刻度点。如果需要在一些特别点处标记坐标轴刻度,就必须修改坐标轴刻度属性。例如:

```

>>t=0 : 900;
>>plot(t,0.25 * exp(-0.05 * t))
>>set(gca,'xlim',[0 100])
>>set(gca,'ylim',[0 0.25])
>>set(gca,'YTick',[0 0.05...
    0.075 0.1 0.15 0.2 0.25])

```

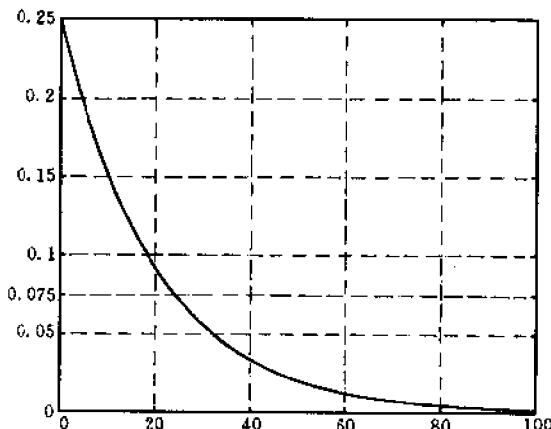


图 3-5 自定义坐标轴刻度

图像呢?

首先看一个简单的例子:

```

>> load clown
>> ax1=subplot(221);image(X);
>> ax2=subplot(222);image(X);
>> axes(ax1), colormap(map)
>> MapGray = gray;
>> NewColorMap = [map; MapGray];
>> axes(ax2), image(X+size(map,1))
>> colormap(NewColorMap)

```

这组语句生成的图形如图 3-5 所示。

也可以用文字代替数值刻度标记,例如:

```

>> set(gca,'YTickLabel',...
    '0|0.05|Cutoff|0.1|...
    0.15|0.2|0.25')

```

3. 多色谱技巧

MATLAB 为每个图形窗口定义了一个着色色谱 Colormap,该图形窗口内每个坐标系中的图形输出时,都是用这个色谱进行着色。如果要在同一个图形窗口中输出两幅图像,比如一幅是彩色的,而另一幅是黑白的,那么怎样才能在一个图形窗口中同时正确地显示这两幅

这组语句首先在两个不同的坐标系内显示两幅同样的图像，并使用相同的色谱。第五条语句生成与图像色谱长度相同的灰色色谱，然后将图像的色谱与这个灰色色谱拼成一个大的色谱。最后两条语句的作用是用新色谱的后面一半色谱，即灰色来显示同一幅图像，但是，此时图像被着色成灰色。这是因为 `image` 函数用图像数据矩阵 `X` 的值直接作为图形窗口 Colormap 色谱的索引值，为了引用后一半色谱，必须将图像数据加上 `size(map,1)`。

但是对于 `imagesc` 函数，就不能这样使用多色谱，因为它与 `mesh` 和 `patch` 等函数一样，是利用坐标系 CLim 属性值，将图形数据线性地映射到整个 Colormap 色谱上。由于坐标系 CLim 属性决定坐标系使用的颜色范围，所以可以为每个坐标系定义一个特定的 CLim 属性值，使得该坐标系只使用图形窗口 Colormap 色谱矩阵中的部分颜色。于是，问题归结为：为坐标系计算出新的 CLim 属性值，使得在该 CLim 属性值下，坐标系中图形（曲面等）的 CData 属性值中的最小值映射到长度为 CmLen 色谱的第 BeginSlot 行，而最大值映射到第 EndSlot 行。为此定义一个函数 M 文件实现这个目的，M 文件 `newclim.m` 如下：

```
function CLim = newclim(BeginSlot,EndSlot,CDmin,CDmax,CmLen)
PBeginSlot = (BeginSlot - 1) / (CmLen - 1);
PEndSlot = (EndSlot - 1) / (CmLen - 1);
PCmRange = PEndSlot - PBeginSlot;
DataRange = CDmax - CDmin;
ClimRange = DataRange / PCmRange;
NewCmin = CDmin - (PBeginSlot * ClimRange);
NewCmax = CDmax + (1 - PEndSlot) * ClimRange;
CLim = [NewCmin,NewCmax];
```

那么，下面的语句组可以在同一个图形窗口中显示彩色与灰色的曲面：

```
>> ax1 = subplot(121); mesh(peaks)
>> ax2 = subplot(122); mesh(peaks)
>> clim1 = get(ax1,'clim');
>> clim2 = get(ax2,'clim');
>> newmap = [get(gcf,'colormap');gray];
>> cmlength = size(newmap,1);
>> set(ax1,'clim',newclim(1,cmlength/2,...
    clim1(1),clim1(2),cmlength))
>> set(ax2,'clim',newclim(cmlength/2+1,cmlength,',...
    clim2(1),clim2(2),cmlength))
```

请读者试一试运行效果。

3.4 线段对象

线段对象（Line）是最简单的图形对象，它是坐标系对象的子对象。在 MATLAB 中，曲线是用线段对象表示的。高层图形函数，如 `plot`、`plot3`、`contour`、`contour3` 等等都可以创建线

段对象。线段对象既可以定义在 2 维坐标系中，也可以定义在 3 维坐标系中。

3.4.1 线段对象创建函数

线段对象的低层创建函数是 line，它的命令形式有下列几种：

```
>> line(X,Y)
>> line(X,Y,Z)
>> line(X,Y,Z,'属性名',属性值,...)
>> line('属性名',属性值,...)
>> h = line(...)
```

每种命令形式都能在当前坐标系中创建一个或多个线段对象，最后一种形式还返回创建线段对象的句柄值。如果不给出“属性名/属性值”对，MATLAB 就用线段对象的缺省值创建该线段对象。在这些命令形式中，对变量 X、Y、Z 的解释与高层线段函数 plot 和 plot3 是一样的，当 X、Y、Z 等都是矩阵时，按列的对应方式创建多个线段对象。在这种多个线段对象的创建中，线段对象、线段类型与着色分别由坐标系 LineStyleOrder 属性和 ColorOrder 属性共同决定，即对 LineStyleOrder 定义的每种线段类型，依次用 ColorOrder 定义的颜色作线段，如此循环。

line 创建函数与 plot 等高层创建函数的重要区别在于，line 函数不检查当前坐标系的 NextPlot 属性值，它总是直接向当前坐标系中添加新的线段对象，并自动更新坐标系的 XLim、YLim、ZLim 等属性，使得新加入的线段对象完全落在坐标系中，而高层函数 plot 等调用另一个函数 newplot 来决定当前坐标系 NextPlot 属性的状态，除非 NextPlot 的属性值为 add。plot 等在输出图形之前，会先删除当前坐标系中已存在的图形对象。

下面是一个作图的例子：

```
>> t = 0 : pi/20 : 2 * pi;
>> hline1 = plot(t,sin(t),'k');
>> hline2 = line(t+.06,sin(t),'LineWidth',4,'Color',[.8 .8 .8]);
>> set(gca,'Children',[hline1, hline2])
```

最后一条语句的作用是将第一条线段提到图形子对象的最上层。

3.4.2 线段对象属性

与对其他的图形对象一样，MATLAB 为线段对象定义了各种属性，并通过这些属性来管理线段对象。每个线段对象共有 25 种属性，分述如下（公共属性除外）。

1. 外形属性

(1) Color 属性

Color 属性的取值是某些颜色的预定义字符或颜色 RGB 数值，它决定线段对象输出时的颜色。可以用 set 函数设置该属性值，将线段对象按指定颜色着色。

(2) LineStyle 属性

LineStyle 属性的取值是—(实线,缺省值)、—(虚线)、:(点线)、-.(点虚线)或 none(无线),这是 MATLAB 预定义的线段类型,参见表 2-1。该属性决定线段对象输出(显示)时所用的线段类型。

(3) LineWidth 属性

LineWidth 属性的取值是以磅为单位的数值,缺省值为 0.5 磅。它定义线段对象输出(显示)时线段的宽度。增大 LineWidth 属性值,可以得到较粗的线段(曲线)。

(4) Marker 属性

Marker 属性的取值可以是+ (加号顶点标记)、o (圆圈顶点标记)、* (星号顶点标记)、. (实点顶点标记)、× (乘号顶点标记)、square (方块顶点标记)、diamond (钻石形顶点标记)、^ (三角形顶点标记△)、v (三角形顶点标记▽)、> (三角形标记▷)、< (三角标记◁)、pentagram (五角星标记★)、hexagram (六角星标记*)、none (空顶点标记,缺省值)。这是 MATLAB 预定义的顶点标记符号,参见表 2-1。

在 MATLAB 中,曲线是通过一些顶点的折线表示的,该属性定义这段折线顶点处的标记,则线段看上去会更美观。缺省属性值是 none,表示无顶点标记。Marker 属性与 LineStyle 属性无关,但可以通过这两者的不同组合得到不同式样的线段。例如,如果只想用顶点标记表达线段对象,那么只需将 LineStyle 属性设置为 none 即可。

(5) MarkerEdgeColor 属性

MarkerEdgeColor 属性的取值是 auto(缺省值)、none 或某些颜色的预定义字符,或颜色 RGB 数值。该属性的取值如果是颜色值,那么它定义顶点标记的颜色,或某些有边界的顶点标记(圆圈标记、方块标记、钻石形标记、五角形标记、六角形标记,以及 4 种三角形标记)的边界颜色。如果取值为 none,则对有边界的顶点标记来说,边界没有颜色,边界也不可见,而对其他的顶点标记,其效果是顶点标记不可见。如果 MarkerEdgeColor 取值为 auto,那么将使用线段对象 Color 属性定义的颜色对标记或标记边界进行着色。

(6) MarkerFaceColor 属性

MarkerFaceColor 属性的取值是 auto、none(缺省值),或某些颜色的预定义字符或颜色 RGB 数值。该属性的取值如果是颜色值,那么它定义某些有边界的顶点标记(圆圈标记、方块标记、钻石形标记、五角形标记、六角形标记,以及四种三角形标记)的内部区域填充颜色。如果取值为 none,则有边界的顶点标记内部无颜色,并且内部区域是透明的。如果 MarkerFaceColor 取值为 auto,那么内部区域用坐标系 Color 属性定义的颜色填充。如果坐标系 Color 属性值为 none,则使用图形窗口 Color 属性定义的颜色填充。对无边界的顶点标记,该属性不起作用。

(7) MarkerSize 属性

MarkerSize 属性的取值是以磅为单位的数值,缺省值为 6 磅。该属性决定顶点标记的大小。

(8) XData、YData、ZData 属性

XData、YData、ZData 3 种属性的取值都是数值向量,这 3 个向量的对应元素构成线段对象各顶点在坐标系中的 x、y、z 坐标值,线段对象就是通过这些顶点定义的。线段对象创建函数的最低层形式为:

```
>> line('XData',x,'YData',y,'ZData',z,'属性名',属性值,...)
```

2. 辅助属性

(1) EraseMode 属性

EraseMode 属性的取值是 normal(缺省值)、none、xor 或 background。该属性定义 MATLAB 系统使用何种方式在屏幕上画线段对象和擦去线段对象,这些技术在生成动画序列时特别有用。在 normal 模式下,当画线段或者擦去线段时,重新刷新整个图形,相当于重新生成该对象。在有遮挡的情况下,还要进行 3 维分析,保证原有的图形对象恢复到正确状况。虽然在这种模式下,重新刷新的图形对象保持原有的着色,但是这种模式速度太慢,并且要消耗大量内存。在 none 模式下,当线段对象被删除时,该线段对象原来的图形仍保留在屏幕上。在 xor 模式下,当在屏幕上画线段对象或者擦去线段对象时,用线段的颜色与线段下屏幕上应有的颜色值作“异或”运算,所以最后输出的线段图形的颜色依赖于屏幕上原有的颜色。在 background 模式下,为了擦去线段,用坐标系的背景颜色画线段,这样做的结果会破坏线段下原有的图形对象的着色。

3.4.3 属性应用技巧

1. 顶点标记控制技巧

MATLAB 4.2 版将顶点标记定义为线型,MATLAB 5.0 版对此作了区分。在 MATLAB 4.2 版中,线段对象的 LineStyle 属性的取值可以是—(实线)、—(虚线)、:(点线)、—·(点虚线)、+(加号)、○(圆圈)、*(星号)、×(乘号)等,没有规定 Marker 属性;而 MATLAB 5.0 版新增加了 Marker 属性,上述的后 4 种取值不再属于 LineStyle 属性,而属于 Marker 属性的取值范围。这一点请读者注意。请看下例(在 MATLAB 4.2 版系统下运行):

```
>> h=line('xdata',[1,2,3],'ydata',[1,4,2],'lineStyle','○');
>> axis([-0.5 3.5 -0.5 4.5])
>> set(h,'MarkerSize',20)    %增大圆圈
>> set(h,'ButtonDownFcn',['size_h=get(h,"MarkerSize");',...
    'set(h,"MarkerSize",size_h-1)'])
```

那么,在 3 个圆圈上每按一次鼠标,圆圈标记就减小一点。在本例中第一条语句生成仅有 3 个圆圈顶点标记的线段对象,但这条语句在 MATLAB 5.0 中不能正确执行。因为在 MATLAB 5.0 中,○不再是 LineStyle 属性的合法取值。要得到与上述第一条语句同样的效果,即只生成 3 个圆圈,在 MATLAB 5.0 必须用下列语句实现:

```
>> h=line('xdata',[1,2,3],'ydata',[1,4,2],'lineStyle','none',...
    'Marker','○');
```

请读者注意这条语句与下面语句的区别:

```
>> h=line('xdata',[1,2,3],'ydata',[1,4,2],'Marker','○');
```

此时 h 是具有实线将圆圈连接的线段。

2. 线段对象的动态生成技巧

线段对象的 EraseMode 属性十分有用,如果能巧妙应用之,可以得到图形动态生成的方法。这个属性用于控制图形刷新时,MATLAB 重新绘制图形对象的方式。先看下面这个例子,该例以动态的方式生成两条随机曲线,每条曲线由运动的曲线头、中间部分及曲线尾部 3 部分组成:

```

>> clg                      %清图形窗口
>> nstep=400;                %曲线顶点数
>> y=randn(nstep,2)          %曲线纵坐标值
>> y=cumsum(y);             %序列和
>> x=1:400;
>> head_1=line, body_1=line; tail_1=line;
>> head_2=line, body_2=line; tail_2=line;
>> set(head_1,'color','c','LineStyle','○')
>> set(head_2,'color','c','LineStyle','+')
>> set(head_1,'erase','xor','xdata',x(1),'ydata',y(1,1));
>> set(head_2,'erase','xor','xdata',x(1),'ydata',y(1,2));
>> set(body_1,'color','g','LineStyle','—','erase','none')
>> set(body_2,'color','r','LineStyle','—','erase','none')
>> set(tail_1,'color','y','LineStyle','—','erase','none')
>> set(tail_2,'color','w','LineStyle','—','erase','none')
>> for ii=2:40
    num=ii-1:ii;
    Set(head_1,'xdata',x(ii),'ydata',y(ii,1))
    Set(head_2,'xdata',x(ii),'ydata',y(ii,2))
    Set(body_1,'xdata',x(num),'ydata',y(num,1))
    Set(body_2,'xdata',x(num),'ydata',y(num,2))
    drawnow
end
>> for ii=2:360
    num1=ii-1:ii;
    num2=ii+39:ii+40
    Set(head_1,'xdata',x(ii+40),'ydata',y(ii+40,1))
    Set(head_2,'xdata',x(ii+40),'ydata',y(ii+40,2))
    Set(body_1,'xdata',x(num2),'ydata',y(num2))
    Set(body_2,'xdata',x(num2),'ydata',y(num2,2))
    Set(tail_1,'xdata',x(num1),'ydata',y(num1,1))
    Set(tail_2,'xdata',x(num1),'ydata',y(num1,2))
    drawnow
end

```

```

>> for ii=361:400
    num.=ii-1:ii
    Set(tail_1,'xdata',x(num),'ydata',y(num,1))
    Set(tail_2,'xdata',x(num),'ydata',y(num,2))
    drawnow
end

```

3.5 曲面对象

曲面对象(Surface)是图形对象的一种,它是坐标系对象的子对象。在 MATLAB 中,数学曲面是用曲面对象表示的。高层图形函数,如 mesh、surf、surfc、surfl 等等都可以创建曲面对象。曲面对象定义在 3 维坐标系中,而坐标系可以在任何视点模式下。

3.5.1 曲面对象创建函数

曲面对象的低层创建函数是 surface,它的命令形式有下列几种:

```

>> surface(Z)
>> surface(Z,C)
>> surface(X,Y,Z)
>> surface(X,Y,Z,C)
>> surface(...'属性名',属性值,...)
>> h = surface(...)

```

每种命令形式都在当前坐标系中创建一个曲面对象,最后一种形式还返回创建曲面对象的句柄值。如果不给出“属性名/属性值”对,MATLAB 就用曲面对象的缺省值创建该曲面对象。在这些命令形式中,对变量 X、Y、Z、C 的解释与高层曲面函数 mesh 和 surf 等的解释是一样的,其中 X 和 Y 可以是向量,通常,变量 C 称为颜色数据矩阵。参见 2.2.3 节。

surface 创建函数与 mesh 等高层创建函数的重要区别在于,surface 函数不检查当前坐标的 NextPlot 属性值,它总是直接向当前坐标系中添加新的曲面对象,并不更新坐标系的 XLim、YLim、ZLim 等属性,使新加入的曲面对象完全落在坐标系中。而高层函数 mesh 等要调用另一个函数 newplot 指令决定当前坐标系 NextPlot 属性的状态,除非 NextPlot 属性值为 add。mesh 等在输出图形之前,会先删除当前坐标系中已存在的图形对象。

通常,如果变量 X、Y、Z 都是矩阵,那么它们是同维数的矩阵,但是颜色数据矩阵 C 可以与 Z 是不同维的,这时 FaceColor 属性值(参见下一小节)应该是 texturemap。下面是一个作图的例子:

```

>> load clown
>> surface(peaks,flipud(X),'FaceColor','texturemap',...
    'EdgeColor','none')
>> colormap(map)

```

```
>> view([-37.5 50])
```

当没有坐标系存在时, surface 函数创建的坐标系的视点为 [0 90], 即 2 维坐标, 上面的最后一条语句将其转换到 3 维坐标系, 生成的图形如图 3-6 所示。

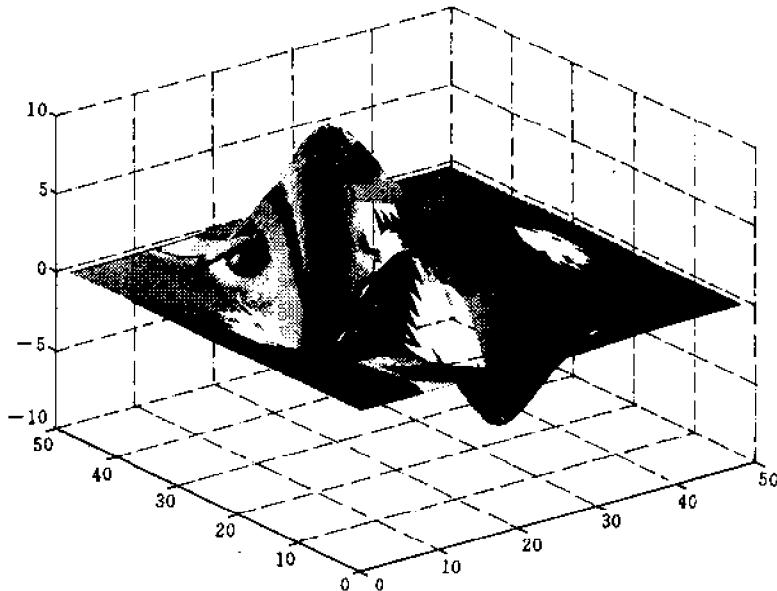


图 3-6 曲面的 texture 着色方式

3.5.2 曲面对象属性

与对其他的图形对象一样, MATLAB 为曲面对象定义了各种属性, 并通过这些属性来管理曲面对象。每个曲面对象共有 40 种属性, 分述如下(公共属性除外)。

1. 外形属性

(1) CData 属性

CData 属性的取值是数值矩阵或全真颜色 RGB 3 维数组。在第一种情况下, 称 CData 为索引式颜色数据矩阵; 在第二种情况下, 称 CData 为 RGB 颜色数组, 统称为颜色数据矩阵。它是曲面数据 ZData 对应的曲面各顶点处的颜色数据。通常, CData 矩阵的维数与 ZData 矩阵维数相同, 但是如果 FaceColor 属性值为 texturemap, 那么这两个矩阵的维数可以不必相同。在这种情况下, MATLAB 将 CData 矩阵线性地映射为与 ZData 同维数的矩阵。

如果 CData 为索引式颜色数据矩阵, 那么根据 CDataMapping 属性的取值情况, 或者将 CData 的数据按通常的线性映射方式映射到图形窗口 Colormap 色谱矩阵的索引号中, 或者将 CData 的数据直接解释为 Colormap 色谱矩阵的索引号。

如果 CData 为 RGB 颜色数组, 此时如果坐标数据, 如 XData 的维数是 $m \times n$, 那么 CData 的维数必须是 $m \times n \times 3$ 。如果计算机系统不支持全真颜色, MATLAB 就用

Colormap 和 Dithermap 对全真颜色进行模拟。

(2) CDataMapping 属性

CDataMapping 属性的取值是 scaled(缺省值)或 direct, 该属性定义对索引式颜色数据矩阵的解释方式。如果 CData 定义的是全真颜色数据, 这个属性无任何作用。当该属性值为 scaled 时, MATLAB 根据坐标系 CLim 属性值, 按线性方式将 CData 数据映射到 Colormap 索引号中; 如果它的属性值为 direct, 那么 MATLAB 将 CData 数据直接解释为 Colormap 索引号。因此, 在这种情况下, CData 数据通常为整数。如果数值小于 1, 就解释为 1, 数值大于 length(colormap), 就解释为 length(colormap)。对于非整数数值, 按其整数部分截断。

(3) EdgeColor 属性

EdgeColor 属性的取值是 MATLAB 合法的颜色值以及 none、flat 或 interp。取缺省值时为黑色。该属性定义曲面网格线的颜色或着色方式。在 MATLAB 中, 曲面都是用网格曲面表示的, 对网格线划分的小曲面片进行着色后就可得到一张视觉良好的实曲面, 参见 2.2.3 节。

如果 EdgeColor 属性值是 RGB 颜色值或 MATLAB 表示的预定义颜色的字符, 那么曲面网格线就用这种颜色着色, 取缺省值为 k, 即网格线为黑色。但是网格线上顶点处的标记不受该值影响; 如果 EdgeColor 属性值为 none, 则在曲面上不画网格线; 如果 EdgeColor 属性值为 flat, 则用 CData 第(i, j)项数值定义的颜色分别对 x 方向网格线段(x(i, j), y(i, j), z(i, j)) (x(i+1, j), y(i+1, j), z(i+1, j)) 和 y 方向的网格线段(x(i, j), y(i, j), z(i, j)) (x(i, j+1), y(i, j+1), z(i, j+1)) 涂色, 这里, x, y, z 分别是曲面顶点的坐标值; 如果 EdgeColor 属性值为 interp, 则根据网络线段两顶点的颜色值按线性插值方式对两顶点之间的线段着色。

(4) FaceColor 属性

FaceColor 属性的取值是 MATLAB 合法的颜色值以及 none、flat(缺省值)或 interp, 该属性定义曲面上网格片的着色方式。

如果 FaceColor 属性值是 RGB 颜色值或 MATLAB 表示的预定义颜色的字符, 则用该颜色对整个曲面着色, 曲面网格线除外; 如果 FaceColor 属性值为 none, 那么不对曲面片进行着色, 此时曲面是由网格线构成的透明曲面, 而 mesh 函数生成的网格曲面的曲面片用黑色着色。虽然在屏幕上看上去, mesh 函数生成的曲面与 FaceColor 属性值为 none 时定义的曲面是一样的, 但它们之间是有区别的。在透明的情况下, 可以看到曲面背后的图形对象。例如, 读者可以从下面的例子中看出两者区别:

```
>>mesh(peaks(20))
>>set(gcf, 'color', 'r')
>>h=mesh(peaks(20))
>>set(gcf, 'color', 'r')
>>set(h, 'facecolor', 'none')
```

如果 FaceColor 属性值为 flat(缺省值), 那么用 CData 第(i, j)项数值定义的颜色对由四顶点(x(i, j), y(i, j), z(i, j)), (x(i+1, j), y(i+1, j), z(i+1, j)), (x(i+1, j+1), y(i+1, j+1), z(i+1, j+1)), (x(i, j+1), y(i, j+1), z(i, j+1)) 决定的网格片进行着色。

这里, x 、 y 、 z 分别是曲面顶点的坐标值;如果 FaceColor 属性值为 interp, 则用每个曲面片 4 顶点对应的颜色值按双线性插值的方式定义曲面片的颜色;如果 FaceColor 属性值为 texturemap, 则 MATLAB 将 CData 数据线性地映射到整个曲面上, 通常 CData 的维数远大于坐标数据矩阵例如 ZData 的维数。例如, 如果 CData 的数据来自一幅图像, 那么在这种着色模式下, 可以将该图像镶在一张曲面上。

(5) LineStyle 属性

LineStyle 属性的取值是—(实线, 缺省值)、…(虚线)、:(点线)、-.(点虚线)或 none(无线), 这是 MATLAB 预定义的线段类型, 参见表 2-1。该属性决定曲面对象上网格线所使用的线段类型。

(6) LineWidth 属性

LineWidth 属性的取值是以磅为单位的数值, 缺省值为 0.5 磅, 它定义曲面对象上网格线的宽度。增大 LineWidth 属性值, 可以得到较粗的网格线。

(7) Marker 属性

参见线段对象 Marker 属性, 两者意义相同。

该属性定义曲面顶点上的标记, 缺省属性值是 none, 表示无顶点标记。Marker 属性与 LineStyle 属性无关, 可以通过这两者的不同组合得到不同式样的曲面网格线。例如, 如果只想用顶点标记表示曲面, 那么只要将 LineStyle 属性和 FaceColor 属性设置为 none 即可。

(8) MarkerEdgeColor 属性

与线段对象的 MarkerEdgeColor 属性的取值相同。如果 MarkerEdgeColor 取值为 auto, 则使用曲面对象 EdgeColor 属性定义的颜色对标记或标记边界着色。

(9) MarkerFaceColor 属性

与线段对象 MarkerFaceColor 属性的取值相同。如果取值为 none, 则有边界的顶点标记内部无颜色, 并且内部区域是透明的;如果 MarkerFaceColor 取值为 auto, 则那么内部区域用该顶点的 CData 数据定义的颜色填充。

(10) MarkerSize 属性

MarkerSize 属性的取值是以磅为单位的数值, 缺省值为 6 磅, 该属性决定顶点标记的大小。

(11) MeshStyle 属性

MeshStyle 属性的取值是 both(缺省值)、row 或 column, 该属性定义画曲面网格线的方式。如果取值为 both, 表示画出 x 和 y 方向的所有网格线;如果其取值为 row, 只画 x 方向的网格线;如果取值为 column, 只画 y 方向的网格线。

(12) XData、YData、ZData 属性

XData、YData、ZData 3 种属性的取值都是数值向量或矩阵, 它们的对应元素构成曲面对象各顶点在坐标系中的 x 、 y 、 z 坐标值, 曲面对象就是这些顶点定义的。下面是常用的创建曲面对象的语句:

```
>> surface('XData',x,'YData',y,'ZData',z,'属性名,属性值,...)
```

如果输入的数据 x 和 y 分别是向量, 那么 MATLAB 将这些向量扩展为矩阵时, x 作为行向量扩展为 XData, 且行数与 ZData 的行数相同; y 作为列向量扩展为 YData, 且列数与 ZData

的列数相同。

2. 光照效应属性

(1) AmbientStrength 属性

AmbientStrength 属性的取值是介于 0 到 1 之间的数值, 它定义曲面上均匀分布的照明光强度。只有当坐标系中存在有可见的光源对象时, 该属性定义的照明光才会增加曲面上的光照效应。而照明光的颜色由坐标系 AmbientColor 属性确定。

(2) BackFaceLighting 属性

BackFaceLighting 属性的取值是 unlit、lit 或 reverselit(缺省值)。MATLAB 5.0 为曲面上的顶点定义了法向方向, 该属性定义对那些顶点法向背离视点的曲面片(称为背向曲面片)的明暗着色方式。如果此属性取值为 unlit, 那么背向曲面片上无光照效应; 如果取值为 reverselit, 则背向曲面片上也有光照效应; 如果取值为 lit, 其效果基本上与 reverselit 模式类似, 只是对闭曲面(例如球面)其效果有点区别。对于闭曲面, 在 reverselit 模式下, 边界点上也有光照效应, 这不符合实际情况, 而 lit 模式即可避免这个问题。该属性对于在视觉上区分曲面的内侧与外侧是有用的。

(3) DiffuseStrength 属性

DiffuseStrength 属性的取值是介于 0 到 1 之间的数值, 它定义光照落在曲面上的散射成分的强度, 散射光来自光源对象。

(4) EdgeLighting 属性

EdgeLighting 属性的取值是 none(缺省值)、flat、gouraud 或 phong, 该属性定义计算光源对象对曲面网格线作用效果的算法。在 none 模式下, 光源对象对曲面网格线无影响; 在 flat 模式下, 光源对象在网格线线段上的光照效果是均匀一致的; 在 gouraud 模式下, 根据网格线线段两端点的光照效应作线性插值决定线段上的光照效应; 在 phong 模式下, 先根据线段两端点的法向沿线段作线性插值, 再计算线段上每点(像素点)的反射系数, 据此决定线段上的光照效应。这样计算的结果在视觉上效果最好, 但花费时间较长。

(5) FaceLighting 属性

FaceLighting 属性的取值是 none(缺省值)、flat、gouraud 或 phong, 该属性定义计算光源对象对曲面片作用效果的算法。在 none 模式下, 光源对象对曲面片无影响; 在 flat 模式下, 光源对象在曲面片上的光照效果是均匀一致的; 在 gouraud 模式下, 根据计算曲面片顶点的光照效应, 再作线性插值决定曲面片上的光照效应; 在 phong 模式下, 先根据曲面片顶点的法向在曲面片上作线性插值, 再计算曲面片上每点(像素点)的反射系数, 据此决定曲面片上的光照效应。这样计算的结果在视觉上效果最好, 但花费时间较长。

(6) NormalMode 属性

NormalMode 属性的取值是 auto(缺省值)或 manual, 该属性定义曲面上顶点法向向量的生成方式。如果它的取值为 auto, 则 MATLAB 会自动根据曲面坐标数据计算法向向量; 如果用户自定义法向向量, 那么 MATLAB 将该属性值置为 manual。

(7) SpecularColorReflectance 属性

SpecularColorReflectance 属性的取值是介于 0 到 1 之间的数值, 该属性定义镜面反射光颜色。如果它的值是 0, 那么镜面反射光颜色就同时依赖于曲面的颜色和光源对象的颜

色;如果它的值是 1,镜面反射光颜色就仅与光源 Color 属性有关;在取其他值的情况下,MATLAB 就在曲面颜色与光源对象颜色之间按线性比例进行调节。

(8)SpecularExponent 属性

SpecularExponent 属性的取值是大于或等于 1 的数值,该属性定义镜面反射点的大小。

(9)SpecularStrength 属性

SpecularStrength 属性的取值是介于 0 到 1 之间的数值,该属性定义光源在曲面上的镜面反射强度。

(10)VertexNormals 属性

VertexNormals 属性的取值是数值向量或矩阵,该属性记录着曲面顶点的法向数据。一般情况下,MATLAB 会自动生成 VertexNormals 属性值,用户也可以自定义法向数据。MATLAB 生成曲面光照效应时会使用 VertexNormals 数据。这样,就可以达到各种光照效应下的曲面。

3. 辅助属性

(1)EraseMode 属性

与线段对象 Erase Mode 属性的取值及意义相同。

3.6 区域片对象

区域片对象(Patch)是图形对象的一种,它是坐标系对象的子对象。区域片对象的图形是由平面或空间中某些顶点构成的 3 维立体,例如长方体就可以看成是具有六个面的区域片对象。每个面是由一些顶点组成的多边形,多边形可能是凹的或自相交的。可以将曲面对象看成为一个特殊的区域片对象,因为曲面对象的每个面就是由 4 个顶点构成的多边形区域。MATLAB 5.0 对区域性的概念略有更新,区域先被认为是由面构成的几何体。

3.6.1 区域片对象创建函数

区域片对象的低层创建函数是 patch,它的命令形式有下列几种:

```
>> patch(X,Y,C)
>> patch(X,Y,Z,C)
>> patch(...'属性名',属性值,...)
>> patch('属性名',属性值,...)
>> h = patch(...)
```

每种命令形式都可在当前坐标系中创建一个区域片对象,最后一种形式还返回创建区域片对象的句柄值。如果不给出“属性名/属性值”对,MATLAB 就用区域片对象的缺省值创建该区域片对象。

在这些命令形式中,变量 X、Y、Z 为向量或矩阵,它们的维数相同,对应的元素定义顶点的坐标值。若它们都是矩阵,则 MATLAB 按列定义区域片对象的面,变量 C 称为颜色数据,它的解释较复杂,与区域片的其他属性值有关,共同定义区域片上的着色方式。假设 X、Y、Z

的维数都是 $m \times n$, 那么当 C 的取值是数值(作为 Colormap 的索引值)或全真颜色 RGB 单色值时, MATLAB 对所有的区域都着这种颜色; 当 C 的取值是 $1 \times n$ 向量(作为 Colormap 的索引值)或 $n \times 3$ (或 $1 \times n \times 3$)RGB 全真颜色值时, MATLAB 对每个不同面将着不同的颜色; 当 C 的取值是 $m \times n$ 矩阵(作为 Colormap 索引值)或 $m \times n \times 3$ RGB 全真颜色值时, MATLAB 将 C 定义的颜色对应到每个顶点, 而面可以用 flat 或 interp 模式着色。另外, 在某些情况下, 着色还受 FaceVertexCData 属性值的影响。

注意: 这里说的着色是对面的内部的着色, 区域片边界的颜色由 EdgeColor 等属性定义。

第一种调用形式在平面上定义区域片对象, 第二种形式在空间中定义区域片对象。patch 创建函数与 fill 等高层创建函数的重要区别在于, patch 函数不检查当前坐标 NextPlot 属性值, 它总是直接向当前坐标系中添加新的区域片对象, 而高层区域片生成函数 fill 等需调用另一个函数 newplot 来决定当前坐标系 NextPlot 属性的状态, 除非 NextPlot 属性值为 add, fill 等在输出图形之前, 会先删除当前坐标系中已存在的图形对象。

当没有坐标系存在时, patch 函数创建的坐标系的视点为 [0 90], 即 2 维坐标, 如有必要, 要由用户用 view 等函数将其转换到 3 维坐标系。

有两种方式可以定义区域片对象, 第一种是直接定义顶点坐标和颜色值, 例如:

```
>> x = [0 1; 1 1; 0 0];
>> y = [2 2; 2 1; 1 1];
>> z = [1 1; 1 1; 1 1];
>> id_color = [24 37];
>> patch(x,y,z,id_color)
```

另一种更方便的方式是先定义所有的顶点, 然后定义顶点之间的联系, 即每个面由哪些顶点构成, 最后定义颜色数据, 例如, 下面语句组的作用与上述例子相同:

```
>> vert = [0 1 1; 0 2 1; 1 2 1; 1 1 1];
>> fac = [1 2 3; 1 3 4];
>> id_color = [24 37];
>> patch('faces', fac, 'vertices', vert, 'FaceVertexCData', id_color)
```

vert 矩阵中的每一行数值为顶点 x、y、z 坐标值, 而 fac 矩阵的元素值是 vert 中顶点的序号, 它的每行定义区域片对象的一个面, 当然每个面的顶点个数不必相同, 顶点数较少时, 可以用 NaN 代替 vert 使得 fac 每行元素个数相同。

3.6.2 区域片对象属性

区域片对象属性大体上与曲面对象属性相同, 并通过这些属性来管理区域片对象。每个区域片对象共有 40 余种属性, 分述如下。

1. 外形属性

(1) CData 属性

CData 属性的取值可以是数值或 MATLAB 预定义的颜色字符值, 也可以是向量, 还可以是数值矩阵或全真颜色 RGB 3 维数组, 该属性定义区域片颜色及着色方式。MATLAB

针对不同的 CData 值,来解释如何对区域片对象进行着色。当 CData 属性值的数据是数值时,MATLAB 或者将数据按线性方式映射到整个 Colormap 色谱上,或者将数据直接作为 Colormap 色谱的索引值,例如对整数值就是如此,当然也依赖于 CDataMapping 属性。如果 CData 数据是 RGB 3 维数组,MATLAB 就直接引用 RGB 值定义的颜色,而不必在 Colormap 色谱中作索引,但是当计算机系统不支持全真颜色时,MATLAB 就用 Dithermap 色谱作全真颜色模拟。

CData 属性值矩阵维数与坐标值 XData、YData、ZData 的维数之间有一定的匹配关系。设坐标值矩阵的维数都是 $m \times n$,如果 CData 属性值是数值(或颜色字符),或者 $1 \times 1 \times 3$ RGB 全真颜色值,那么整个区域片用该颜色着色,当然,除区域片边界外,边界颜色是另定义的。如果 CData 属性值是长度为 n 的数值向量,或者是 $1 \times n \times 3$ RGB 全真颜色值,那么区域片的 n 个面分别用这 n 种颜色着色;如果 CData 属性值是与坐标值同维数的矩阵,或 $m \times n \times 3$ RGB 全真颜色值,那么区域片对象的每个顶点对应一种颜色,而每个区域片上的着色方式分为 flat 和 interp 两种方式。

(2) CDataMapping 属性

CDataMapping 属性的取值是 scaled(缺省值)或 direct,该属性定义对索引式颜色数据矩阵的解释方式。如果 CData 定义的是全真颜色数据,则这个属性无任何作用。当该属性值为 scaled 时,MATLAB 会根据坐标系 CLim 属性值,按线性方式将 CData 数据映射到 Colormap 索引号中;如果它的属性值为 direct,则 MATLAB 将 CData 数据直接解释为 Colormap 索引号。因此,在这种情况下,CData 数据通常为整数。如果数值小于 1,就解释为 1,数值大于 length(colormap),就解释为 length(colormap)。对于非整数数值,则按其整数部分截断。

(3) EdgeColor 属性

EdgeColor 属性的取值是 MATLAB 合法的颜色值,以及 none,flat 或 interp,缺省值时为黑色。该属性定义单个区域片对象边界线颜色或着色方式。

如果 EdgeColor 属性值是 RGB 颜色值或 MATLAB 预定义的颜色的字符,那么边界线就用这种颜色着色,取缺省值为 b 时,边界线为黑色,但是边界线上顶点处的标记不受该值影响;

如果 EdgeColor 属性值为 none,则不画边界线。

如果 EdgeColor 属性值为 flat,则边界线上两顶点间线段的颜色与起点的颜色相同,边界方向是由给定坐标值的顺序确定的;如果 EdgeColor 属性值为 interp,则根据两顶点的颜色值按线性插值方式对两顶点之间的线段进行着色。

(4) FaceColor 属性

FaceColor 属性的取值是 MATLAB 合法的颜色值,none, flat 或 interp,缺省值为 w,该属性定义区域片的颜色与着色方式。

如果 FaceColor 属性值是 RGB 颜色值或 MATLAB 表示预定义颜色的字符,则用该颜色对整个区域片着色,区域边界线除外;如果 FaceColor 属性值为 none,则不对区域片进行着色,此时区域片是由边界线构成的透明片;如果 FaceColor 属性值为 flat(缺省值),那么区域片中每个面的颜色就由其第一个顶点的颜色决定;如果 FaceColor 属性值为 interp,则用

每个面片上顶点对应的颜色值按多线性插值的方式定义面上的颜色。

(5) Faces 属性

Faces 属性的取值是 $m \times n$ 数值矩阵, 称为连接矩阵。它定义了用 Vertices 属性定义的顶点的连接关系。连接矩阵的元素都是 Vertices 中顶点的序号, 或 NaN, 每一行的顶点构成区域片对象的面。Faces 属性和 Vertices 属性一起提供了一种更经济地定义区域片对象的方法。

(6) FaceVertexCData 属性

FaceVertexCdata 属性的取值为矩阵, 它决定了由 Faces 属性和 Vertices 属性定义的区域片及顶点的颜色与着色方式。

FaceVertexCData 属性值的解释依赖于它的维数, 对于索引式颜色数据, FaceVertexCdata 属性可以取单个颜色值, 它定义整个区域片的颜色; 可以是 $n \times 1$ 向量, 这里 n 是 Faces 属性值的行数, 它定义区域片每个面的颜色; 也可以是 $n \times 1$ 向量, 这里 n 是 Vertices 属性值的行数, 它定义每个顶点的颜色。

对于全真颜色 RGB 值, FaceVertexCdata 属性的值可以是 1×3 矩阵, 它定义整个区域片的颜色; 可以是 $n \times 3$ 矩阵, 其中 n 是 Faces 属性值的行数, 它定义区域片每个面的颜色; 也可以是 $n \times 3$ 矩阵, 这里 n 是 Vertices 属性值的行数, 它定义每个顶点的颜色。

(7) LineStyle 属性

LineStyle 属性的取值是一(实线, 缺省值)、- - (虚线)、: (点线)、-. (点虚线)或 none(无线), 这是 MATLAB 预定义的线段类型, 参见表 2-1。该属性决定区域片边界线所用的线段类型。

(8) LineWidth 属性

LineWidth 属性的取值是以磅为单位的数值, 缺省值为 0.5 磅, 它定义区域片边界线的宽度。增大 LineWidth 属性值, 可以得到较粗的边界线。

(9) Marker 属性

Marker 属性的取值与线段对象的 Marker 属性的取值相同。参见表 2-1。

该属性定义区域片顶点上的标记, 缺省属性值是 none, 表示无顶点标记。Marker 属性与 LineStyle 属性无关, 可以通过这两者的不同组合得到不同式样的区域片边界线。

(10) MarkerEdgeColor 属性

MarkerEdgeColor 属性的取值及意义与线段对象 MarkerEdgeColor 属性相同。如果 MarkerEdgeColor 取值为 auto, 则用区域片对象 EdgeColor 属性定义的颜色对标记或标记边界进行着色。

(11) MarkerFaceColor 属性

MarkerFaceColor 属性的取值及意义与线段对象 MarkerFaceColor 属性相同。如果 MarkerFaceColor 取值为 auto, 则内部区域用坐标系背景色或图形窗口背景色填充。

(12) MarkerSize 属性

MarkerSize 属性的取值是以磅为单位的数值, 缺省值为 6 磅, 该属性决定顶点标记的大小。

(13) Vertices 属性

Vertices 属性的取值是 $n \times 3$ 矩阵, 它定义区域片各顶点的坐标值。每一行对应一个顶点, 其值为 x, y, z 坐标值。

(14) XData、YData、ZData 属性

XData、YData、ZData 3 种属性的取值都是数值向量或矩阵, 它们的对应元素构成区域片对象各顶点的 x, y, z 坐标值, 区域片对象就是这些顶点定义的。下面是常用的创建区域片对象的语句:

```
>> patch('XData',x,'YData',y,'ZData',z,'属性名',属性值,...)
```

输入数据 x, y, z 的列对应着定义区域片对象某个面上顶点的坐标值。

2. 光照效应属性

控制光照效应的属性共有 10 个, 它们的属性名、属性取值、属性的含义与曲面对象的光照效应属性完全相同, 在此不再重复。参见第 3.5 节光照效应属性。

3. 辅助属性

EraseMode 属性

EraseMode 属性的取值是 normal(缺省值)、none、xor 或 background, 该属性定义 MATLAB 系统使用何种方式在屏幕上画区域片对象或擦去区域片对象, 这些技术在生成动画序列时特别有用。在 normal 模式下, 当画区域片或者擦去区域片时, 会重新刷新有变化的图形部分, 必要时还要进行 3 维分析, 保证所有的图形对象被正确着色。虽然在这种模式下, 重新刷新的图形对象还保持原有的着色和层次, 但是这种模式速度太慢, 并且要消耗大量内存; 在 none 模式下, 当区域片对象被删除时, 该区域片对象原来的图形仍保留在屏幕上; 在 xor 模式下, 当在屏幕上画区域片或者擦去区域片对象时, 用区域片每个像素的颜色与区域片下屏幕上像素应有的颜色值作“异或”运算, 所以最后输出的图形的颜色依赖于屏幕上原有的颜色; 在 background 模式下, 为了擦去区域片, 要用坐标系的背景颜色画区域片, 这样做的结果会破坏区域片下原有的图形对象的着色。

3.7 图像对象

MATLAB 通过创建图像对象的方法显示一幅图像, 图像对象也是图形对象的一种, 它是坐标系对象的子对象, 即第三级子对象。图像对象本身没有子对象。

3.7.1 图像对象创建函数

图像对象的创建函数是 image, 它既是高层创建函数, 也是低层创建函数, 视其调用形式而定。调用 image 函数的形式有如下几种:

```
>> image(C)
>> image(x,y,C)
>> image(...,'属性名',属性值,...)
>> image('属性名',属性值,...)
>> handle = image(...)
```

前 3 种调用形式是其高层调用,第四种形式是其低层调用,而第五种调用形式则返回创建图像对象的句柄值。低层调用直接在当前坐标系中显示图像对象,而高层形式调用 newplot 函数决定坐标系并检查坐标系 NextPlot 属性值,再决定加入图像对象方式。这些语句都是用于创建图像对象,将图像数据 C 按其数据形式或者解释为图形窗口 Colormap 的索引值,或者直接解释为 RGB 全真颜色值。

图像数据 C 可以是索引式颜色值,也可以是 RGB 全真颜色数组值。取索引式颜色值时,C 数据作为图形窗口 Colormap 色谱的索引来引用颜色对图像对象的相应点进行着色;取 RGB 全真颜色数组值时,MATLAB 直接用 RGB 定义的颜色对图像对象的相应点进行着色。在不支持全真颜色的计算机系统上,用 Dithermap 色谱中定义的颜色作模拟图像颜色,图像数据保存在图像对象 CDATA 属性中,所以它或者是 $m \times n$ 数值矩阵,或者是 $m \times n \times 3$ 3 维 RGB 颜色数组。

在 MATLAB 5.0 中,图像数据有两种数据格式,即 8 位数据格式和 64 位数据格式(双精度)。通常,MATLAB 在内存中对任何数据都使用双精度格式(不管它的显示格式是什么),由于图像数据的数量很大,为了节省内存,MATLAB 提供了 8 位数据格式。在显示图像时,MATLAB 对这两种数据格式的数据使用不同的解释。对于索引式颜色数据矩阵 C,如果它是 64 位数据格式的,那么其元素值通常在范围 [1, length(colormap)] 内。当元素值为 1 时,就映射到 Colormap 色谱的第一行,如此等等。如果它是 8 位数据格式,那么其元素值在范围 [0 255] 中,那么作索引映射时必须偏移一位,即数值 0 映射到 Colormap 色谱第一行,如此等等。对于 3 维 RGB 颜色数组 C,在 64 位数据格式下,各元素之值落在 [0 1] 范围内;而在 8 位数据格式下,各元素之值落在 [0 255] 范围内。

3.7.2 图像对象属性

每个图像对象具有 20 个属性,它们的定义与内容所述如下。

1. 外形属性

(1) CDATA 属性

CDATA 属性的取值可以是数值矩阵或全真颜色 RGB 3 维数组,该属性定义图像对象的颜色数据。函数 image(C) 将变量 C 的值传送给 CDATA 属性,MATLAB 针对不同的数据 CDATA,按不同的方式进行解释:如果 CDATA 是数据矩阵,则或者将其值直接作为 Colormap 色谱的索引值,或者按线性方式映射到整个 Colormap 色谱上,这取决于 CDATAMapping 属性值;如果 CDATA 数据是 RGB 3 维数组,MATLAB 就直接引用 RGB 值定义的颜色,而不必在 Colormap 色谱中作索引,但是当计算机系统不支持全真颜色时,MATLAB 就用 Dithermap 色谱作全真颜色模拟。

(2) CDATAMAPPING 属性

CDATAMAPPING 属性的取值是 scaled 或 direct(缺省值),该属性定义对索引式颜色数据矩阵的解释方式。如果 CDATA 定义的是全真颜色数据,这个属性无任何作用。当该属性值为 scaled 时,MATLAB 根据坐标系 CLIM 的属性值,按线性方式将 CDATA 数据映射到 Colormap 索引号中;如果它的属性值为 direct,那么 MATLAB 将 CDATA 数据直接解释为

Colormap 索引号。因此，在这种情况下，CData 数据通常为整数。如果数值小于 1，就解释为 1；数值大于 length(colormap)，就解释为 length(colormap)。对于非整数数值，则按其整数部分截断。

(3) XData、YData 属性

XData、YData 两种属性都是 2 元素向量。通常，XData 的取值是 [1, size(C, 2)]，YData 的取值是 [1, size(C, 1)]，它们分别定义了图像占用的 x、y 坐标范围。

2. 辅助属性

EraseMode 属性

EraseMode 属性的取值是 normal(缺省值)、none、xor 或 background，该属性定义 MATLAB 系统使用何种方式在屏幕上显示和擦去图像对象，这些技术在生成动画序列时特别有用。在 normal(缺省)模式下，当画或者擦去图像时，重新刷新有变化的图形部分，必要时还要进行 3 维分析，保证所有的图形对象被正确着色。虽然在这种模式下，重新刷新的图形对象保持原有的着色和层次，但是这种模式速度太慢，并且要消耗大量内存。在 none 模式下，当图像对象被删除时，该图像对象的图形仍保留在屏幕上；在 xor 模式下，当在屏幕上画或者擦去图像对象时，MATLAB 用图像每个像素的颜色值与图像下屏幕上像素现有的颜色值作“异或”运算，所以最后输出的图形的颜色依赖于屏幕上原有的颜色；在 background 模式下，为了擦去图像，MATLAB 用坐标系的背景颜色重新显示图像。这样做的结果会破坏图像下原有的图形对象的颜色。

3.8 文字对象

文字对象主要用于在坐标系中写说明文字。MATLAB 5.0 对文字对象作了重大改进，在文字对象中还可以使用希腊字母、数学符号及公式等，这就是 MATLAB 处理文字对象时对 LATEX 文本(LATEX 是一种用于数学文章的排版语言及排版系统。)的解释功能。这样就可以在坐标系中添加更形象的说明文字。

3.8.1 文字对象创建函数

文字对象的低层创建函数是 text，交互式的高层文字对象创建函数是 gtext。函数 text 可以有下列几种命令形式：

```
>> text(x,y,'字符串')
>> text(x,y,z,'字符串')
>> text(...,'属性名',属性值...)
>> h = text(...)
```

第一种命令形式是在 2 维坐标系，或者以 z 坐标值为 0，在 3 维坐标系的指定位置上写出给定的字符串(文字)；第二种命令形式是在 3 维坐标系的指定位置上写出给定的字符串(文字)；第三种命令形式是根据指定位置和属性值写出说明文字；第四种命令形式返回创建的文字对象的句柄值。由于 text 是低层函数，所以它只是向当前坐标系添加文字对象，而不改

变坐标系的属性。如果给出的坐标值超出了当前坐标系的范围，则文字对象是不可见的，但所创建的文字对象是存在的。

变量 x, y, z 是同长度向量，而字符串变量可以是字符矩阵，其行数与坐标变量向量长度相同，也可以是字符串块数组或用竖线“|”隔开的字符串，其块个数与坐标变量向量长度相同。此时，MATLAB 将创建多个文字对象。注：MATLAB 5.0 以前版本的 text 函数仅支持单行文字说明。

坐标值定义文字对象字符串定位的相对点，文字对象 VerticalAlignment 属性和 HorizontalAlignment 属性决定文字对象的最后位置。坐标值按照当前坐标系的数值单位解释。

3.8.2 文字对象属性

每个文字对象具有 30 个属性，它们的定义与内容如下所述。

1. 基本属性

(1) Color 属性

Color 属性的取值是 RGB 颜色值或 MATLAB 预定义的颜色名称，它定义文字对象的显示颜色，其缺省值为 w(白色)。

(2) Interpreter 属性

Interpreter 属性的取值是 latex(缺省值)或 none，该属性控制 MATLAB 对 String 属性中字符及控制字符串的解释方式。在 latex 模式下，MATLAB 按 LATEX 意义解释 String 属性中的控制符。LATEX 是一种十分流行的数学公式排版工具，在 LATEX 意义解释下，MATLAB 可以在说明文字加入数学公式以及希腊字母等；在 none 模式下，则按 ASCII 码解释控制字符串。

(3) String 属性

String 属性的取值是字符串、字符串矩阵或字符串块数组，它记录着文字对象的说明文字内容。在 MATLAB 5.0 中，文字对象既可以是单行说明文字，也可以是多行说明文字，后者由字符串矩阵或字符串块数组定义。在字符串中，除了标准的 ASCII 字符外，还可以有以 LATEX 命令形式出现的控制字符串。在 latex 模式下，MATLAB 将控制字符串解释为数学符号或希腊字母等。目前，MATLAB 5.0 支持全部的大小写希腊字母及 47 个数学符号，参见表 3-1。

为了嵌入特殊符号和字母(如俄文字符等)，可以使用其他系统字库。控制字符串 “\fontname{字库名}” 表示使用给定的字库。利用 LATEX 的变换字体的命令，还可以改变说明文字中部分字符的字体，这些命令包括 \bf(黑体)、\it(斜体)、\sl(斜体的一种，很少使用)、\rm(正体)。对于在这些命令的作用范围之外的字符，仍使用字体属性定义的字体。

(4) Units 属性

Units 属性的取值是 pixels(像素)、normalized(相对单位)、inches(英寸)、centimeters(厘米)、points(磅)、data(坐标数据单位，缺省值)，该属性定义 Extent 属性和 Position 属性的度量单位。坐标数据单位是指按坐标系各坐标单位进行解释。其他单位都是从坐标系矩形的左下角开始度量的。

表 3-1 控制字符串及与其对应的符号

控制字符串	符号	控制字符串	符号	控制字符串	符号
\alpha	α	\beta	β	\gamma	γ
\delta	δ	\epsilon	ε	\zeta	ζ
\eta	η	\theta	θ	\vartheta	ϑ
\iota	ι	\kappa	κ	\lambda	λ
\mu	μ	\nu	ν	\xi	ξ
\pi	π	\rho	ρ	\sigma	σ
\varsigma	ς	\tau	τ	\upsilon	υ
\phi	φ	\chi	χ	\psi	ψ
\omega	ω	\Gamma	Γ	\Delta	Δ
\Theta	Θ	\Lambda	Λ	\Xi	Ξ
\Pi	Π	\Sigma	Σ	\Upsilon	Υ
\Phi	Φ	\Psi	Ψ	\Omega	Ω
\sim	~	\leq	≤	\infty	∞
\clubsuit	♣	\diamondsuit	♦	\heartsuit	♥
\spadesuit	♠	\leftrightarrow	↔	\leftarrow	←
\uparrow	↑	\rightarrow	→	\downarrow	↓
\circ	◦	\forall	∀	\exists	∃
\ni	∋	\cong	≅	\approx	≈
\Re	ℝ	\oplus	⊕	\cup	∪
\subseteq	⊆	\in	∈	\pm	±
\geq	≥	\propto	∞	\partial	∂
\bullet	•	\div	÷	\neq	≠
\aleph	ℵ	\wp	℘	\oslash	∅
\supseteq	⊇	\subset	⊂	\circ	◦

2. 字体属性

(1) FontAngle 属性

FontAngle 属性的取值是 normal(缺省值)、italic 或 oblique, 该属性定义文字对象使用的某种系统字库, normal 表示使用系统的正体字库, italic 和 oblique 在通常的计算机系统上都表示使用系统的斜体字库。

(2) FontName 属性

FontName 属性的取值是字符串, 它给出文字对象使用的字库类名, 缺省值是

Helvetica。

(3)FontSize 属性

FontSize 属性的取值是整数值,它定义文字对象使用的字体大小,其单位由 FontUnits 属性定义。缺省值是 10,以磅为单位。

(4)FontWeight 属性

FontWeight 属性的取值是 light、normal(缺省值)、demi 或 bold,该属性决定文字对象字体的粗细,MATLAB 据此从系统字库中挑选一种适当的字库。通常取 demi 和 bold 时,都表示黑体。

(5)FontUnits 属性

FontUnits 属性的取值是 points(磅:缺省值)、normalized(相对单位)、inches(英寸)、centimeters(厘米)或 pixels(像素),该属性定义 FontSize 属性的度量单位。相对单位将 FontSize 属性值解释为坐标系高度的比例。

3. 位置属性

(1)Extent 属性

Extent 属性的取值为 4 元素数值向量 [left, bottom, width, height],该属性只可读,用户不能改写。它定义了文字对象所占据的矩形区域的大小,(left, bottom)是该区域左下角相对坐标系矩形左下角距离的值,而 width 和 height 是区域宽度和高度值,其单位由 Units 属性定义。

(2)HorizontalAlignment 属性

HorizontalAlignment 属性的取值是 left(缺省值)、center 或 right,它定义文字对象在水平方向相对于 Position 点的对齐方式。

(3)Position 属性

Position 属性的取值是 2 元素或 3 元素数值向量 [x,y] 或 [x,y,z],如果略去 z,则在系统内以 0 代替 z,缺省值为 [0,0,0]。它定义坐标系中的一个参考点,文字对象以该点定位,并根据 HorizontalAlignment 和 VerticalAlignment 属性值决定文字对象的最后位置。

(4)Rotation 属性

Rotation 属性的取值是数量值,缺省值为 0,它定义文字对象的旋转角度,取正值时表示逆时针方向旋转,取负值时表示顺时针方向旋转。

(5)VerticalAlignment 属性

VerticalAlignment 属性的取值是 top、cap、middle(缺省值)、baseline 或 bottom,该属性定义文字对象的纵向定位方式。top 表示将文字对象的顶端在纵向上与 Position 参考点对齐;cap 表示将文字对象中的大写字母的顶端与 Position 参考点对齐,这种方式大体上与 top 方式是一样的;middle(缺省方式)表示将文字对象的纵向中点与 Position 参考点对齐;baseline 表示将文字对象字符串的基线点与 Position 参考点对齐;bottom 表示将文字对象的底端与 Position 参考点对齐。

4. 辅助属性

(1)EraseMode 属性

EraseMode 属性的取值是 normal(缺省值)、none、xor 或 background,该属性定义

MATLAB 系统使用何种方式在屏幕上写文字对象或擦去文字对象。在 normal(缺省)模式下,当写或者擦去说明文字时,MATLAB 重新刷新有变化的图形部分,必要时还要进行 3 维分析,保证所有的图形对象被正确着色;在 none 模式下,当文字对象被删除时,该文字对象的图形仍保留在屏幕上;在 xor 模式下,当在屏幕上写或者擦去文字对象时,用文字上每个像素颜色值与文字下屏幕上像素原有的颜色值作“异或”运算,所以最后输出的图形的颜色依赖于屏幕上原有的颜色;在 background 模式下,为了擦去说明文字,MATLAB 用坐标系的背景颜色重写说明文字,这样做的结果会破坏文字下原有的图形对象的着色。

3.9 光源对象

光源对象是 MATLAB 5.0 引进的新的图形对象类型。光源对象仅对区域片对象和曲面对象产生光照效应,利用光源对象可以生成更逼真的 3 维实体模型和曲面。在 MATLAB 4.2 版本中,虽然也可以生成具有光照效应的曲面,但除了光源的方向外,用户不能对光源进行控制。现在用户不仅可以控制光源的方向,还可以控制光源的位置、颜色、强度等。光源对象在坐标系中是不可见的,它的存在是通过它对区域片对象和曲面对象的作用效果来反映的。

3.9.1 光源对象创建函数

创建光源对象的函数是 light,它具有下列两种命令形式:

```
>> light('属性名', 属性值, ...)  
>> h = light('属性名', 属性值, ...)
```

它们的作用都是在当前坐标系中创建光源对象,除非对 Parent 属性给定特定的坐标系句柄值。第二种形式还返回所创建的光源对象的句柄值。

光源对象对区域片对象和曲面对象产生光照效应,其效果是由区域片对象和曲面对象的 AmbientStrength、DiffuseStrength、SpecularStrength、SpecularExponent、SpecularColorReflectance、VertexNormals 等属性共同作用的结果。

3.9.2 光源对象属性

由于光源对象对用户来说是不可见的,因此,许多图形对象的公共属性对光源对象就毫无意义。事实上,光源对象只使用如下基本属性。

(1) Color 属性

Color 属性的取值是 RGB 颜色值或 MATLAB 预定义的颜色名称,它定义光源对象发出的光的颜色,缺省颜色为白色。

(2) Style 属性

Style 属性的取值是 infinite(缺省值)或 local,它定义光源对象的类型。在 infinite 模式

下, MATLAB 将光源点置于无穷远处, 相当于平行光束; 在 local 模式下, 光源点被置于坐标系的特定位置上, 此时光线是从该点发出的散射光束。该属性值决定了 Position 属性的解释方式。

(3) Position 属性

Position 属性的取值是 3 元素数值向量 [x,y,z], 它定义了光源对象的位置。如果 Style 属性值为 local, 那么光源对象置于由该向量定义的坐标位置上, 它发出的光线是散射光束; 如果 Style 属性值为 infinite, 那么该向量将决定无穷远光源对象发出的平行光束的方向。

3.10 缺省属性及其设置

3.10.1 缺省属性值

为了应用上的方便, MATLAB 对所有图形对象的属性都规定有缺省的属性值, 又称为 factory 定义值。MATLAB 也允许用户定义自己的缺省属性值。从定义处开始, 用户定义的缺省属性值将覆盖 factory 定义值, 随后的图形对象生成函数按用户定义的缺省值来设置未指定的属性值。

任何一个图形创建函数启动时, 对于没有提供属性值的那些属性, MATLAB 都将使用属性缺省值。缺省属性值的寻找过程是这样的: 首先从当前的图形对象开始逐步向上检查它的高层图形对象类, 直到找到用户定义的缺省值或 MATLAB 预定义的缺省值, 即 factory 属性值。用户定义的缺省值优先级最高。在使用缺省值时, 以当前被创建图形对象的最邻近高层图形对象类中用户定义的缺省值优先, 而 factory 定义值的优先级最低, 也就是 factory 定义值的定义级别最高, 但它的使用级别最低。当用户定义了新的缺省属性值后, MATLAB 将保留着 factory 定义值的备份, 必要时可以恢复 factory 定义值。

新的缺省属性值一经定义后,会在创建该类图形对象时随即发挥作用,除非在低层创建函数中用“属性名/属性值”指定了特殊值。但是,新的缺省值对已存在的该类图形对象没有影响。

3.10.2 设置缺省属性值

1. 定义缺省属性值

除了最低级别的图形对象外, MATLAB 中的每种类型的图形对象都为其各子级图形对象的属性定义了缺省属性名, 借此来设置子级图形对象的缺省属性值。缺省属性名由 3 部分组成, 第一部分是 default, 第二部分是图形对象类型名, 第三部分是该类型图形对象的属性名。例如, DefaultLineWidth, 这个缺省属性名的值就是线段对象的线段宽度的缺省值。

MATLAB 图形对象缺省值的定义方法是:在其高层图形对象上用 set 函数将相应的缺省属性名定义为所期望的缺省值, 其一般命令形式为

```
>> set(handle, '缺省属性名', 属性值)
```

其中, handle 是高层图形对象的句柄值。那么在这个高层图形对象的范围内,所有相应的子图形对象就使用所定义的缺省值。定义缺省值的高层图形对象级别越高,相应的缺省属性值的作用范围就越广。

例如,为了在当前的窗口内用白色画线段对象,只要将线段对象的 Color 属性的缺省值设置为白色,即

```
>> set(gcf, 'DefaultLineColor', 'w')
```

随后,无论在这个图形窗口的哪个坐标系中画线段对象,都用白色对线段对象进行着色。由于线段对象是坐标系对象的子对象,因而也可以在坐标系设置线段对象属性的缺省值。只有在该坐标系中输出线段对象时,才使用用户定义的缺省值。例如:

```
>> set(handle_axes, 'DefaultLineColor', 'w')
```

由于线段对象是坐标系对象的子对象,又是图形窗口对象的子对象的子对象,因此, DefaultLineColor 是线段对象的高层对象,即坐标系对象、图形窗口对象和根对象的缺省属性,而不是线段对象自身的属性。线段对象是最低层的图形对象,MATLAB 对最低层的图形对象没有定义缺省属性名。又如,如果要将图形窗口对象的缺省背景色黑色改为蓝色,那么必须在图形窗口对象的父对象,即根对象中定义相应的缺省属性:

```
>> set(0, 'DefaultFigureColor', 'b')
```

此后,再创建的图形窗口的背景颜色就是蓝色。

如果用户对某些属性总是有自己的特殊要求,那么可以在根对象层上设置所期望的缺省属性值,并把这些设置语句写入到 startup.m 文件中,这样 MATLAB 系统启动后就在根对象层上定义了用户的缺省属性值。

2. 缺省属性值的使用

在创建图形对象时,没有被指定的属性,将使用用户定义的缺省属性值或 factory 定义值。MATLAB 还提供了一种方法,可以将已存在的图形对象的任何属性值设定成用户定义的缺省值或 factory 定义值。在 MATLAB 中,任何属性都接受 default 和 factory 值输入。属性值“default”含义是:将在根据缺省值搜索过程用首先遇到的缺省值作为相应的属性值。例如,下面的语句设置曲面的 EdgeColor 属性为绿色:

```
>> h = surf(peaks);  
>> set(0, 'DefaultSurfaceEdgeColor', 'g');  
>> set(h, 'EdgeColor', 'Default')
```

第一条语句对创建的曲面网格线着黑色,这是因为系统定义的曲面对象 EdgeColor 的缺省属性值为 k(黑色),即 factory 定义值为 k;第二条语句是在根对象层上定义曲面对象 EdgeColor 的缺省属性值为 g(绿色);而第三条语句的作用是将已存在的曲面对象 EdgeColor 属性值设置为用户定义的缺省值,即 g。虽然 EdgeColor 的缺省属性值是在根对象层上定义的,但是它的定义级别低于 factory 定义值的定义级别,所以执行第三条语句后, MATLAB 就使用根对象层上定义的缺省属性值。

如果 EdgeColor 属性的缺省值已经在坐标系对象或图形窗口对象中存在,则它们将首先被使用,代替在根对象层中设置的缺省值。

属性值 factory 的含义是将相应的属性值设置成 factory 定义值。例如,下面的语句将曲面 h 的 EdgeColor 属性重新设置成黑色(factory 定义值):

```
>> h = surf(peaks);
>> set(0, 'DefaultSurfaceEdgeColor', 'g')
>> set(h, 'EdgeColor', 'Default') %网格线成为绿色
>> pause
>> set(h, 'EdgeColor', 'factory') %网格线还原为黑色
```

值得注意的是,如果对于某个属性没有用户定义的缺省属性值存在,那么 default 和 factory 的作用是相同的。

3. 用户定义缺省值的删除

一旦用户定义了自己的缺省属性值,MATLAB 总是试图使用用户定义的缺省属性值。在其后的过程中,有可能不再希望使用已定义的缺省属性值,这时就需要删除用户定义的缺省属性值。为此,MATLAB 对所有的缺省属性都定义了属性值 remove,其含义是删除用户定义的缺省值,例如,语句

```
>> set(0, 'DefaultSurfaceEdgeColor', 'remove')
```

的作用就是删除在根对象层中定义的曲面 EdgeColor 属性的缺省值。

3.10.3 例子

下面给出一个较复杂的例子,来说明在不同层上设置缺省值的方法:

```
>> set(gcf, 'DefaultAxesBox', 'on')
>> subplot(121)
>> set(gca, 'DefaultLineStyle', ':')
>> plot(sin(0 : pi/20 : 2 * pi))
>> hold on
>> plot(cos(0 : pi/20 : 2 * pi))
>> text('Position', [20, 0.4], 'String', 'sin');
>> text('Position', [15, -0.3], 'String', 'cosine',...
    'HorizontalAlignment', 'right')
>> subplot(122)
>> set(gca, 'DefaultTextRotation', 90)
>> plot(sin(0: pi/20 : 2 * pi))
>> hold on
>> plot(cos(0: pi/20 : 2 * pi))
>> text('Position', [20, 0.4], 'String', 'sine')
>> text('Position', [15, -0.3], 'String', 'cosine',...
    'HorizontalAlignment', 'right')
```

可以看出,同样的语句将生成不同的效果,如图 3-7 所示。这主要是由于在坐标系对象

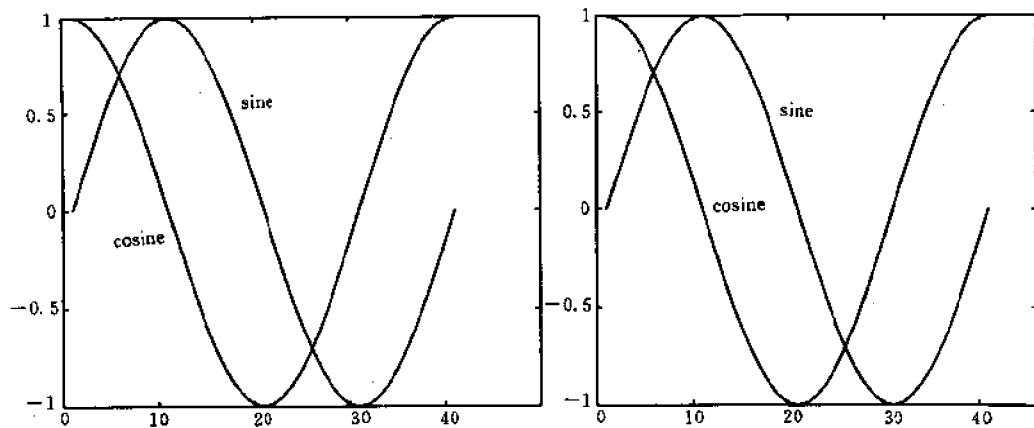


图 3-7 缺省值的作用效果

层设置了缺省属性值。由于 Box 属性的缺省值是在图形窗口层设置的，所以 MATLAB 生成的坐标系都有坐标外框。

第四章 MATLAB 的接口

MATLAB 是一个开放的系统,它具有多种接口功能,使得用户可以十分方便地与其他的应用程序交换数据和信息。它的接口主要有:数据输入输出接口,子程序库调用接口,图形图像输入输出接口,动态链接接口等。本章主要介绍数据输入输出接口和子程序调用接口。

4.1 MATLAB 的数据接口

MATLAB 的数据对象是数组。许多科学计算任务都需要处理大批量的数据。如何将这些数据输入到 MATLAB 的工作空间呢?在 MATLAB 环境中,要视具体情况而定。MATLAB 接受两种形式的数据:一种是 ASCII 码的文本数据文件,这种文件每一行的数据项数是相同的,每个数据项由一个或多个空格分隔开,其内容恰好构成一个矩阵;另一种数据形式是 MATLAB 定义的 MAT 型数据,即所谓的 MAT 文件。首先介绍 MATLAB 系统对 MAT 数据文件的处理方法,而后介绍 MATLAB 的数据文件操作功能。

4.1.1 数据结构

了解 MATLAB 的内部数据结构和数据对象,对于进行数据交换操作是十分必要的。MATLAB 5.0 只支持单一的数据对象类型——数组(Array)。MATLAB 5.0 定义了如下几种变量类型:矩阵(包括数量、向量),字符串,块数组和结构。这些变量类型在 MATLAB 内部都是用数组来存储和管理的。

在 C 语言的意义下,变量类型 mxArray 定义 MATLAB 数组的内部数据结构,即任何一个 MATLAB 变量定义为 C 语言意义下的 mxArray 结构类型。mxArray 结构包含以下的结构域:

- MATLAB 变量名 NAME,这是指向一个字符串的指针,字符串的最大长度由 MATRIX.H 头文件中的常数 mxMAXNAM 规定。这个字符串就是 MATLAB 的变量名字;
- 变量的维数 Dimensions,定义各维数大小。例如数量、向量和矩阵都视为二维的;
- 变量的类型 ClassName,这是一个标识值,指明变量被显示时是数值变量还是字符型变量,即将变量元素看成是字符的 ASCII 码;

- 变量的实或复数类型。如果变量包含有复数值作为其元素,那么该变量对应的 MATLAB 内部数组就包含有实部向量和虚部向量;
- 变量的存储属性 Storage,即是否是稀疏矩阵。指明变量的存储类型,即变量是按满矩阵还是按稀疏矩阵的方式存储。它的取值是 Full 或 Sparse。

数据矩阵定义为 $M \times N$ 数组,它是 MATLAB 语言的基本操作对象。矩阵的含义与通常的数学意义相同,可以是长方矩阵,也可以是方阵,以及单个数量。矩阵的类型可以是复数型矩阵、实数型矩阵或由字符组成的字符型矩阵。在 MATLAB 系统中,矩阵有两种不同的存储类型,即满矩阵的存储方式或稀疏矩阵的存储方式等。数量、向量、字符串等都是特殊的矩阵,它们都是按矩阵的结构来组织和管理的。

如果 MATLAB 变量是满矩阵,那么 mxArray 数组结构类型包含有参数 pr 和 pi。pr 是指向实部向量的指针,pi 是指向虚部向量的指针。如果指针为空(NULL),则表示矩阵是实数型变量。如果 MATLAB 变量是稀疏矩阵,则 mxArray 除了有 pr 和 pi 参数域外,还有另外 3 个参数:nzmax、ir 和 jc。它们的具体含义如下:

nzmax	定义 pr 和 pi(如果 pi 存在)所指向量的长度,是稀疏矩阵中“非零”元素的最大可能总数
pr	指向长度为 nzmax 的双精度型向量的指针,该向量按列由稀疏矩阵中“非零”元素的实部构成
pi	指向长度为 nzmax 的双精度型向量的指针,该向量按列由稀疏矩阵中“非零”元素的虚部构成,如果指针为 NULL,表示是实数型矩阵
ir	指向长度为 nzmax 的整数型向量的指针,其向量元素值为 pr 和 pi 所指的向量中的对应元素在原矩阵中的行指标
jc	指向长度为 $N+1$ 的整数型向量的指针,包含矩阵元素的列信息。对于 $0 \leq j \leq N-1$, jc[j] 记录着矩阵第 $j+1$ 列的第一个“非零”元素在 pr 和 pi 向量中的位置指标, jc[j+1]-1 是该列最后一个“非零”元素在 pr 和 pi 向量中的位置指标, jc[N] 定义为原矩阵变量中“非零”元素的总数 nnz

例如,执行下面的语句生成一个 3×2 的矩阵对象 A:

```
>> A = [1 2; 3 4; 5 6]
```

那么上述的矩阵对象 A 的各属性值分别如下:

```
NAME = "A"
Dimensions = 3×2
ClassName = double
Storage = Full
pr[0] = 1; pr[1] = 3; pr[2] = 5; pr[3] = 2; pr[4] = 4; pr[5] = 6;
pi = NULL
```

4.1.2 MATLAB 数据输入

用户可以用多种方式向 MATLAB 系统输入数据,最佳的输入方式依赖于用户的数据

格式。下面给出几种可以选用的方法。

1. 显示的输入

针对数据量比较小的情形,可以在 MATLAB 的命令窗口中,从键盘直接输入待输入的数据。这种方法就是 1.1.2 节介绍的矩阵的初等输入法,即使用方括号将矩阵的元素按行的顺序括起来,每行用分号隔开,行内各元素用空格或逗号分开。

2. M 文件形式输入

如果数据量较大,且数据是某种硬拷贝形式,不是以计算机可读形式存在时,最有效的办法是用某种文本编辑器直接编写一个包含数据矩阵的 M 文件。这种方法与上一小节所述的方法是类似的,所不同的是,前者是在命令窗口中实时地写 MATLAB 的矩阵输入语句,后者是编写矩阵输入语句的 M 文件,最后可以通过执行 M 文件达到数据输入的目的。

3. ASCII 码数据文件的输入

MATLAB 可以直接读入 ASCII 码的数据文件。ASCII 码的数据文件中的数据形式必须是一个矩阵,要求数据文件每一行的数据个数必须相同,每行数据对应于矩阵的每一行,每行的元素用空格分开。用户可以将数据编辑成一个 ASCII 码文件,或者由其他的程序将所输出的数据写到一个 ASCII 码文件中。ASCII 码文件中的数据由 MATLAB 的命令 load 装入,命令形式是

```
>> load 文件名(带文件扩展名)
```

该语句在 MATLAB 工作空间中创建一个与文件名(无文件扩展名)相同的变量,该变量表示的矩阵即是 ASCII 码文件的数据组成的矩阵。

4. 低层 I/O 输入方式

MATLAB 提供了文件低层操作函数,可以直接打开文件(open),读文件(fread)。这种方法主要用于装入某种特定格式的数据文件,这种数据文件可能是其他的应用程序生成和创建的。

5. MEX 动态程序输入

如果已经有一些子程序(如 C 子程序或 FORTRAN 子程序)可以用来读取某些特定格式的数据文件,那么可以开发 MATLAB 的动态链接 MEX 子程序,与已有的子程序链接在一起,将数据文件转换成 MATLAB 的 MAT 数据文件,再用 load 函数将有关的数据装入到 MATLAB 系统中。

6. 外部程序转换

如果已有的数据文件格式比较复杂,用户可以开发 FORTRAN 或 C 程序将数据文件直接转换成 MATLAB 的 MAT 数据文件,再用 load 命令装入到 MATLAB 系统中。在这种情况下,用户必须对 MAT 文件结构有较深入的了解。

4.1.3 MATLAB 数据输出

MATLAB 系统输出数据的方式主要有下列的几种。

1. 小型矩阵输出

对于数据量较小的数据矩阵,通过输出的设置,可以用 diary 命令生成命令窗口部分内

容的拷贝文件。该文件是文本文件,记录了命令窗口的命令行,以及 MATLAB 的屏幕输出内容。这样可以将 diary 文件剪接到其他的文件或报告中。

2. ASCII 码数据输出

带-ascii 选项的 save 命令,可以生成一个 ASCII 码的数据文件。在这种方法中,用户可以存储某些指定的变量。例如,下列语句

```
>> A = rand(4, 3)  
>> save temp.dat A -ascii
```

将生成名字为 temp.dat 的 ASCII 码文件,包含矩阵 A 的全部数据,如

0.2113	0.8096	0.4832
0.0824	0.8424	0.6135
0.7599	0.4536	0.2749
0.0087	0.8075	0.8807

3. 低层 I/O 输出

利用 MATLAB 的低层 I/O 函数 fopen 和 fwrite,或者其他低层 I/O 函数,可将数据写入到某个特定格式的文件中。这种方式主要用于将输出的数据写成某些其他应用程序所需要的数据格式。

4. MEX 程序输出

如果已经有某些子程序可以将数据写成特定格式,那么可以开发 MEX 子程序直接调用那些子程序,把最后形成的数据格式作为某些应用程序的输入数据。

5. MAT 格式输出

利用 MATLAB 的 save 命令,可以将数据存储成 MAT 格式文件,再开发一些 C 或 FORTRAN 程序将 MAT 文件转换成某些所需要的特殊格式。例如,将 MATLAB 的图像数据存入到 MAT 文件中,然后利用 C 或 FORTRAN 程序将 MAT 的图像数据转换成一些标准的图像格式文件,如 PCX、TIF、GIF 格式等。

4.1.4 MAT 数据格式

MAT 数据格式是 MATLAB 的数据存储的标准格式。一个 MAT 文件中可以存储一个或多个矩阵数据,矩阵顺序地存在一片连续的磁盘空间上。在每个矩阵的开始处,有一个固定长度的矩阵信息头,这个信息头完整地描述该矩阵的全部特征信息,信息头之后才是矩阵的数据部分,数据占用的磁盘空间字节长度由信息头的长度信息决定。整个这样的结构就构成 MAT 文件中的一个矩阵。

MATLAB 的 save 命令可以将 MATLAB 系统内部数据写为 MAT 文件,而 load 命令可以将磁盘上的 MAT 文件正确地读入到 MATLAB 系统中。除此之外,为了有效地管理 MAT 文件,以及在 MATLAB 外部读取和创建 MAT 文件,MATLAB 提供了一个子程序库,用户可以在 C 或 FORTRAN 程序中直接调用这些子程序来创建和读取 MAT 文件。当然,如果用户了解 MAT 文件的结构,则完全可以开发自己的子程序读写 MAT 文件。但是,使用 MATLAB 提供的子程序库中的标准子程序会更有效。

MAT 子程序库中包含 11 个子程序, 它们的名字和所提供的功能说明如下(以 C 语言库为例):

matOpen	打开 MAT 文件
matClose	关闭 MAT 文件
matGetDir	取 MAT 文件中的变量列表
matGetFp	取 MAT 文件的 C 语言 FILE 句柄
matGetArray	从 MAT 文件中取一个数组
matPutArray	向 MAT 文件中存一个数组
matGetNextArray	从 MAT 文件中取下一个数组
matDeleteArray	从 MAT 文件中删除一个数组
matPutArrayAsGlobal	向 MAT 文件中存一个数组, 使得当用 load 命令装入这个 MAT 文件时, 该数组对应的变量成为 global 变量
matGetArrayHeader	读取 MAT 文件中的 MATLAB 数组头信息
matGetNextArrayHeader	读取 MAT 文件中下一个 MATLAB 的数组头信息

注意: 由于 MATLAB 5.0 将数组结构作为 MATLAB 的基本数据对象, 所以在 MAT 文件中是以数组变量为存储单元。MAT 子程序库中的子程序名中包含“Array”。

有关的程序及目标程序存放在 MATLAB 目录下的子目录 extern 中, 目录 extern 的子目录 include 下是描述数组访问的有关头文件 matrix.h 和 mat.h。在子目录 lib(或 bin)下存有子程序动态链接库文件:

libmat.dll MAT 文件子程序库
libmx.dll MAT 文件数组访问子程序库

在编译用户的程序时, 可以链接这些库中的目标程序。

例如, 在 MATLAB 5.0 环境中, 下面的 C 程序说明了如何使用库中的子程序生成 MAT 文件。源程序如下:

```
#include<string.h>
#include "mat.h"

static double Areal[6] = 1, 2, 3, 4, 5, 6;
main()

    MATFILE *fp;
    mxArray *a;
    fp = matOpen("dataout.mat", "w");
    a = mxCreateDoubleMatrix(3, 2, mxReal);
    memcpy(mxGetPr(a), Areal, 6 * sizeof(double));
    mxSetName(a, "A");
    matPutArray(fp, a);
    matClose(fp);
```

```
mxDestroyArray(a);
```

如果是在 MATLAB 4.2 版本中,那么完成上述任务的 C 语言程序如下:

```
#include<string.h>
#include "matrix.h"
static double Areal[6] = 1, 2, 3, 4, 5, 6;
main()

MATFILE *fp;
Matrix *a;
fp = matOpen("dataout.mat", "w");
a = mxCreateFull(3, 2, Real);
memcpy(mxGetPr(a), Areal, 6 * sizeof(double));
mxSetName(a, "A");
matPutMatrix(fp, a);
matClose(fp);
mxFreeMatrix(a);
```

4.2 文件 I/O 操作

MATLAB 系统具有直接对磁盘文件进行访问的功能。这样,用户不仅可以进行高层的程序设计,在必要时,也可以进行低层次磁盘文件的读/写操作,极大地增强了 MATLAB 程序设计的灵活性。

MATLAB 的文件 I/O 函数是基于 C 语言的文件 I/O 函数的,熟悉 C 语言的读者将会很快地掌握这种方法。即使读者不熟悉 C 语言的文件操作功能,也很容易学会使用 MATLAB 的文件 I/O 操作。

4.2.1 文件的打开与关闭

根据操作系统的要求,在程序要使用或创建一个磁盘文件时,必须向操作系统发出打开文件的命令。文件使用完毕后,还必须告诉操作系统关闭使用过的文件。

在 MATLAB 中,用户可使用 C 语言的同名函数 fopen 打开文件,供系统读或写操作之用。函数 fopen 使用两个参数,第一个参数指明要打开的文件的名字,第二个参数是操作说明字符串。例如,下列命令

```
>> fid = fopen('pen.dat', 'r')
```

用来打开文件 pen.dat,供读数据操作之用。操作说明字符可以是下列的选项:

r 表示对文件进行读数据操作

- w 表示对文件进行写数据操作, 覆盖文件原有内容
- a 表示对文件进行数据追加操作, 在文件尾追加数据
- r+ 表示对文件进行读与写的操作
- w+ 表示创建一个新文件或删除已存在的文件内容, 并进行数据读写操作
- a+ 表示创建一个新文件, 打开一个已存在的文件, 并进行数据追加操作

在有些操作系统, 如 VMS 操作系统上, 要求在操作说明中区分文本和二进制文件。例如, “rb”表示打开一个二进制文件, 进行读数据操作。

正常情况下, 函数 fopen 的返回值是一个非负的整数, 是由操作系统设定的, 作为打开的文件的标识值, 或称为文件句柄值。对于该文件的任何操作, 都是通过这个句柄值来传递的, MATLAB 通过这个句柄值来标识已打开的文件, 实现对该文件的读、写和关闭等操作。由下列命令

```
>> fid = fopen('all')
```

可以取得所有已打开的文件句柄值。

如果文件不能被正确地打开, 例如, 如果文件以“r”方式打开, 但是该文件并不存在, 则会出现文件打开错误。这时, fopen 函数返回 -1 作为该文件的句柄值。程序设计的良好习惯之一是, 每次打开文件时, 都要进行打开操作是否正确的测定。fopen 函数还可以按下列方式调用:

```
>> [fid, message] = fopen('pen.dat', 'r')
```

输出变量 message 中包含了打开文件操作结果的有关信息。例如, 如果文件 pen.dat 不存在, 那么上述语句完成后, fid 的值为 -1, 而 message 的值是字符串

No such file or directory

注意: 错误信息是与操作系统有关的。MATLAB 函数 error 也可以提供有关的错误信息。一旦文件被打开, 该文件就可以被用于读/写等操作。在完成了对文件的读/写操作之后, 必须关闭该文件。关闭文件的方法很简单, 例如:

```
>> status = fclose(fid)
```

用函数 fclose 即可关闭句柄值为 fid 的已打开的文件。为了关闭所有已打开的文件, 只需执行下列命令:

```
>> status = fclose('all')
```

如果关闭文件操作成功, 则 status 的值为 0; 否则 status 的值为 -1。所以, 与打开文件操作一样, 在程序设计中, 用户最好检查一下文件是否正常地关闭。

4.2.2 二进制数据文件的读/写操作

1. 读文件操作

MATLAB 函数 fread(与 C 语言函数同名), 可以用来读二进制文件, 它有多种调用形式。最简单的形式为

```
>> fid = fopen('penny.dat', 'r');
>> A = fread(fid);
```

```
>> status = fclose(fid);
```

将数据文件 penny.dat 中的数据逐字节地按无符号字符型数据读入到矩阵变量 A 中。fread 还有两个可选择的操作参数,利用这两个参数可以控制读入数据的个数以及每个数据的精度。例如,以下语句

```
>> fid = fopen('penny.dat', 'r');
>> A = fread(fid, 100);
>> status = fclose(fid);
```

将 penny.dat 文件中的前 100 个数据读入到列向量 A 中。如果参量 100 用维数向量 [10, 10] 代替,则这 100 个数据将被按列的顺序读入到 10×10 的矩阵 A 中。另外,

```
>> A = fread(fid, Inf);
```

表示读至文件的结尾,生成一个列向量 A。

在 fread 中,还可以加入第三个参数,用来控制读入数据的精度。按数据精度可把数据分为字符型数据、整数型数据和浮点型数据。数据精度与计算机的硬件有关,用户对自己使用的计算机数据精度应有所了解。

MATLAB 支持各种数据精度类型,用户可以指明读入数据时采用的精度类型,有些与 C 语言或 FORTRAN 语言支持的数据精度类型相同。常用的数据精度类型有

char 和 uchar	有符号和无符号字符类型,数据精度为一个字节(8 bits)。对有符号字符类型,数据取值范围是 -128~127;对无符号字符类型,数据取值范围是 0~255
--------------	--

short 和 long	整数类型和长整数类型,数据精度分别为 16 bits 和 32 bits
--------------	--------------------------------------

float 和 double	浮点类型和双精度类型,数据精度分别是 32 bits 和 64 bits
----------------	--------------------------------------

如果 fid 表示的是已打开的浮点类型的数据文件,那么,

```
>> A = fread(fid, 10, 'float')
```

将读入前 10 个浮点型的数据到列向量 A 中。又如,下面的例子用来读包含函数 fread 的使用说明的文本文件 fread.m,然后在命令窗口中打印出来:

```
>> freadid = fopen('fread.m', 'r');
>> F = fread(freadid, Inf, 'uchar');
>> disp(setstr(F'))
>> status = fclose(freadid);
```

上述第二条语句等价于 `F = fread(freadid)`, 即按无符号字符型数据处理。

2. 写文件操作

函数 fwrite 的作用是将一个矩阵的元素按一定的数据精度类型写入到某个打开的文件中,该函数返回写入的数据个数。例如:

```
>> fwriteid = fopen('magic5.bin', 'w');
>> count = fwrite	fwriteid, magic(5), 'integer * 4');
>> status = fclose(fwriteid);
```

将生成 100 字节长度的二进制文件,包含 5×5 个数据,即 5 阶方矩阵的数据,每个数据占用 4 个字节的存储单位,数据类型为整型,输出变量 count 的值为 25。

4.2.3 文件内的位置控制

根据操作系统的规定,在读/写数据时,缺省的方式总是从磁盘文件的开始处顺序向后地在磁盘空间上读/写数据。操作系统控制一个文件指针,指出当前的文件位置。C 或 FORTRAN 语言都有专门的函数来控制和移动文件指针,达到随机访问磁盘文件的目的。MATLAB 的函数 fseek 和 ftell 可以用来设置文件指针的位置,并取得文件指针的当前位置。该位置是下一次读/写操作的起始点。

函数 ftell 给出文件指针相对于文件中某个指定点的偏移量, fseek 函数根据指定的位置来重新设置文件的当前位置。这样,用户就可以跳过部分数据或返回到文件中前面的数据,达到随机访问的目的。这些函数的输入参数包括待操作的文件句柄值,正或负的偏移量和偏移相对点位置。相对点位置可以是文件指针的当前点,用“eof”表示;文件的开始点,用“b0f”表示;文件的结束点,用“eof”表示。正的偏移量表示向文件尾的方向偏移,负的偏移量表示向文件头的方向偏移。偏移量的单位是字节。

下面看一个例子,便可知道如何使用函数 fseek 和 ftell:

```
>> A = [1 : 5];
>> fid = fopen('five.bin', 'w');
>> fwrite(fid, A, 'short');
>> status = fclose(fid);
>> fid = fopen('five.bin', 'r');
>> status = fseek(fid, 6, 'b0f');
>> four = fread(fid, 1, 'short');
>> position = ftell(fid);
>> status = fseek(fid, -4, 'eof');
>> three = fread(fid, 1, 'short');
>> status = fclose(fid);
```

在这个例子中,前面 4 条语句的作用是生成一个包含 5 个数据的文件,每个数据的二进制长度为 2 个字节。第一次调用 fseek 函数时,让文件指针跳过 6 个字节(包含数据 1、2 和 3),以便进一步直接读数据 4。读数据的过程使文件指针向后移动了两个字节,所以 ftell 的结果应该是当前指针指到文件起始位置的与文件首的偏移量,即 8 个字节。第二次调用 fseek 是将文件指针相对当前的位置向文件首的方向回移 4 个字节,即在数据 3 的位置,以便读取数据 3。

4.2.4 格式文件输入和输出

1. 格式输出

MATLAB 的函数 fprintf(与 C 函数同名)的作用是将数据转换成字符串,输出到命令窗口屏幕或写入到一个文件中,通过一个格式说明字符串,说明输出的数据格式。格式说明

控制矩阵数据的输出格式,格式说明由通常的字符和以%开始的格式类型说明符组成。常用的格式类型说明符有:

- %e 数据指数表示形式
- %f 固定小数点位置的数据格式
- %g 在%e 和%f 两种格式中自动选取较短的格式

在格式类型说明中,还包括数据占用的最小宽度和数据精度的说明。例如:

```
>> X = 0 : .1 : 1;
>> Y = [X; exp(X)];
>> fid = fopen('exptable.txt', 'w');
>> fprintf(fid, 'Exponential Function\n\n');
>> fprintf(fid, '%6.2f    %12.8f\n', Y);
>> status = fclose(fid);
```

上面这个例子创建一个文本文件 exptable.txt,包含指数函数的函数值表。第一条 fprintf 语句输出一行标题,随后空两行;第二条 fprintf 语句输出函数值表自身,每组自变量值和函数值占一行,都是固定小数点位置的格式。自变量值占 6 个字符位,而其小数点后的精度是 2 个字符位;函数值占 12 个字符位,小数点后的精度是 8 个字符位。自变量值与函数值之间空两格。

矩阵 Y 的元素按列的顺序转换成格式化的输出,函数反复使用格式说明,直到将矩阵 A 的数据全部转换完毕。

与此函数相关的另一个函数是 sprintf,它的作用是将输出的结果转换成一个字符串。例如:

```
>> roots = sprintf('The square root of %f is %6.4e.\n', 2, sqrt(2));
```

这时,roots 是字符串变量,其内容是“*The square root of 2 is 1.4142.*”。这个函数对于构造字符串十分有用。

2. 格式输入

MATLAB 的文本输入函数是 fscanf,其作用类似于 fprintf。函数 fscanf 以文件句柄值为第一个输入参数,格式说明为第二个输入参数。格式说明由通常的字符串和以%为首的格式类型说明符组成。常用的格式类型说明符有:

- %s 按字符串进行输入转换
- %d 按十进制数据进行转换
- %f 按浮点数据进行转换

在格式说明中,除了单个的空格字符可以匹配任意个数的空格字符外,通常的字符在输入转换时与输入的字符进行一一匹配。函数 fscanf 将输入的文件看作是一个输入流(stream), MATLAB 根据格式说明来匹配输入流,并将在流中匹配的数据读入到 MATLAB 系统中,下面首先看一个文件输入例子:

```
>> fid = fopen('exptable.txt', 'r');
>> title = fscanf(fid, '%s');
>> [table, count] = fscanf(fid, '%d %f');
```

```
>> status = fclose(fid);
```

第一次调用 `fscanf` 时, 文件 `exptable.txt` 的标题行与“%s”匹配, 解释为字符串, 直到遇到回车符为止。第二次调 `fscanf` 时, 读入文件数据表, 每行数据按“%d %f”方式匹配, 即第一个数据按十进制数, 第二个数据按浮点型数据解释, 读入到变量 `table` 中, 直到文件结束。`count` 返回匹配的或输入的数据个数。

另外, `fscanf` 中可以加入一个选项参数, 用它来控制要读入的数据个数。例如, 如果 `fid` 是一个已打开的文件, 包含十进制的数据, 那么,

```
>> A = fscanf(fid, '%5d', 100);
```

将读入 100 个数据到列向量 `A` 中, 而

```
>> A = fscanf(fid, '%5d', [10 10]);
```

将这 100 个数据送到 10×10 的矩阵 `A` 中。

另一个相关的函数是 `sscanf`, 它的作用与 `sprintf` 的作用相反。可以用 `sscanf` 从一个字符串中读取有关的数据。例如:

```
>> rootvalue = sscanf(root2, 'The square root of %f is %f.');
```

将返回包含 2 和其平方根的向量 `rootvalue`。

4.3 MEX 动态连接函数接口

MATLAB 是自成体系的编程系统, 有自己的数据结构。它不仅有如前面几节介绍的外部数据接口, 还具有调用其他子程序的功能, 这些功能被集成在它的应用程序接口(API)中, 其中 MEX 动态链接是它的主要功能。

在 MATLAB 中, 可以调用用户自己开发的 C 或 FORTRAN 子程序, 通过 MATLAB 的 API 函数库将 C 或 FORTRAN 子程序编译成动态链接函数(库), 即 MEX 文件, 以便在 MATLAB 环境中直接调用或链接这些子程序, 达到提高计算效率的目的。

MEX 是一种动态链接的子程序, 如同 MATLAB 的内建函数一样, 能被 MATLAB 的解释器根据调用命令自动地装入和执行。MEX 文件具有以下几个方面的应用:

- 对于某些已有的 C 或 FORTRAN 子程序, 可以通过 MEX 方式在 MATLAB 环境中直接调用, 而不必重新编写相应的 M 文件;
- 对于影响 MATLAB 执行速度的 for 等循环体, 可以编写相应的 C 或 FORTRAN 子程序完成相同功能, 并编译成 MEX 文件, 提高运行速度;
- 对于 A/D 或 D/A 卡, 或其他 PC 硬件, 可以直接用 MEX 文件进行访问, 扩大 MATLAB 的功能;
- 利用 MEX 文件, 还可以使用一些软件, 如 Windows 的用户界面资源等。

当然, 并不是所有的应用都适于用 MEX 文件方式来实现, MATLAB 的主要优越性在于可以节省编程时间。所以在一般情况下, 还是应该以 MATLAB 编程为主。本书中将简单介绍如何开发 MEX 文件。

4.3.1 MEX 文件的使用

MEX 文件是由 C 或 FORTRAN 源程序经过编译生成的 MATLAB 动态链接子程序，它的作用十分类似于 MATLAB 的内建函数。在 Windows 95 系统下，MEX 文件是 32 位 DLL 格式的。

使用 DLL 格式的优势是可以使用户直接访问 Windows 资源环境的各种功能，可以用 DLL 格式的 MEX 文件生成基于 Windows 的用户图形界面，也可以利用 Windows 的动态数据交换(DDE)能力，与其他的 Windows 应用程序交换必要的数据。

MATLAB 的 MEX 文件的扩展名可以是 MEX 和 DLL。在 MATLAB 中，有一个名字为 `ode23.mex` 的文件，这个子程序的运行速度比 `ode23.m` 更快。由于 MATLAB 在调用函数时的顺序是，在同一目录下，先执行 MEX 文件，其次是 DLL 文件，最后是 M 文件，因此，当 MATLAB 系统发出 `ode23` 函数调用命令时，一般是使用 `ode23.mex`。

最好是用 32 位的编译器和 32 位的链接器生成 MEX 文件。目前，支持 32 位运算的编译器有 NDP FORTRAN 386 ver 3.0, MetaWare High C ver 3.0, Borland C++ ver 4.5 和 Watcom C/386 ver 9.0, MATLAB 支持 MicroSoft C Compiler (ver. 7.0)。Windows SDK 被要求用来创建 DLL 格式的 MEX，而不论用户是否要访问 Windows 的用户界面的功能。在 SDK 中，可以从用户的 MEX 文件中访问 Windows 的应用函数。所以，可以在 MEX 中加入一些 Windows 的资源，如对话框、菜单栏、图形动态数据交换功能等。如果在 MEX 文件中使用 Windows 的功能，必须在 MEX 的源程序中加入

```
#include <windows.h>
```

4.3.2 C 语言 MEX 文件

1. 辅助文件

这一小节将讨论如何生成基于 C 语言的 MEX 文件。MEX 文件是通过将用户的 C 源程序与 MATLAB 提供的 API 库链接在一起的方式得到的。

有关生成 MEX 文件的辅助文件及 MATLAB 链接库等文件组织在 MATLAB 目录的子目录\EXTERN 下，在\EXTERN 下分有三个子目录，其结构如下：

```
<MATLAB 安装目录>\EXTERN \INCLUDE  
          \LIB  
          \SRC
```

\INCLUDE 目录下放有有关的头文件：

`mex.h` MEX 文件的函数原型头文件

`matrix.h` 访问矩阵变量的函数原型头文件

`mxArray.h` 访问数组变量的函数原型头文件，供 MATLAB 5.0 版的 API 函数使用

在 MEX 文件的源程序中必须包含头文件 `mex.h`。`\LIB` 目录下存放有 API 函数库文件，主要用于 MEX 文件的链接。`\SRC` 目录下存放有一些例子的源程序文件。

另外,在 MATLAB 安装目录的子目录\BIN 下,CMEX.BAT 是建立 C 语言 MEX 文件的批处理文件。在 MATLAB 5.0 中,不仅可以在 Windows 环境下编译和链接生成 MEX 文件,而且还可以直接在 MATLAB 命令窗口中用 mex 命令编译和链接生成 MEX 文件。

2. MEX 源文件结构与工作原理

MEX 文件是由 MEX 源代码文件经过适当的编译器编译和链接器链接而生成的动态链接子程序。MEX 源代码文件由两部分组成:第一部分称为入口子程序(Gateway Routine),第二部分称为计算功能子程序(Computational Routine)。它们分工明确,入口子程序的作用是在 MATLAB 系统与被调用的外部子程序之间建立通信联系,可以看作是其通信协议。它定义被 MATLAB 调用的外部子程序的入口地址,定义 MATLAB 系统向子程序传递的子程序参数,还定义子程序向 MATLAB 系统返回的结果参数,以及调用计算功能子程序等。而计算功能子程序就是要链接的外部子程序,它用于完成一些特定的计算,它由入口子程序调用。

MEX 源代码文件的两部分既可以分开,也可以组合在一起,但不论怎样,入口子程序的函数名必须是 mexFunction,其构成形式为

```
void mexFunction(
    int nlhs, mxArray * plhs[ ];
    int nrhs, const mxArray * prhs[ ])
{
    /* 必要的 C 代码 ... */
}
```

注意:如果是在 MATLAB 4.2 版本中编译 MEX 文件,那么类型说明 mxArray 应该改为 Matrix。同时,包含语句

```
#include "mex.h"
```

是不可少的。

下面对 mexFunction 函数中的参数作具体说明:

nrhs	整数型变量,记录调用 MEX 文件时的 mxArray(或 Matrix)类型输入参数的个数,即 MATLAB 函数调用的右端变量个数
prhs	指针数组变量,其元素是指向 mxArray(或 Matrix)类型的输入参数变量的指针
nlhs	整数型变量,记录调用 MEX 文件时的 mxArray(或 Matrix)类型的输出参数的个数,即 MATLAB 函数调用的左端变量个数
plhs	指针数组变量,其元素是指向 mxArray(或 Matrix)类型的输出参数变量的指针

这些参数是用来传递 MATLAB 启动 MEX 文件的参数,按 MATLAB 的语法规则,函数(包括 MEX 文件)调用的一般形式是

```
>> [a, b, c, ...] = fun(d, e, f, ...)
```

这里 fun 是 MEX 文件名或 M 文件名,三连点表示参数个数可以是任意的,变量 a, b, c 等称为是输出变量(左变量),而 d, e, f 等称为是输入变量(右变量)。


```

double * x;
unsigned int m, n;

/* 检查变量个数 */
if (nrhs !=1) {
    mexErrMsgTxt("Only one input argument allowed.");
}
else if (nlhs !=1) {
    mexErrMsgTxt("Only one output argument allowed.");
}

m = mxGetM(prhs[0]);
n = mxGetN(prhs[0]);
if (!mxIsNumeric(prhs[0]) || mxIsComplex(prhs[0])
|| mxIsSparse(prhs[0]) || !mxIsDouble(prhs[0])
|| !(m ==1 && n ==1)) {
    mexErrMsgTxt("Input x must be a scalar.");
}

/* 创建矩阵变量作为输出变量. */
plhs[0] = mxCreateFull(m, n, REAL);

/* 赋输入/输出变量指针. */
y = mxGetPr(plhs[0]);
x = mxGetPr(prhs[0]);

/* 调用计算功能子程序. */
squarenumber(y, x);
}

```

从这个例子可以看出：入口子程序起到链接 C 子程序（例如本例中计算平方的函数）与 MATLAB 系统的作用。同时，入口子程序完成大部分的辅助性工作，特别是判别变量类型。例如，在本入口子程序中，有好几条语句用于检测输入/输出变量的个数，以及变量的数据类型，这是必须的步骤。因为 MATLAB 语言不像 C 语言那样提供变量类型的说明，进行必要的类型检查可以避免由于指针与数据不匹配而导致的错误。

为了编译和链接这个子程序，在 DOS 状态下，键入

> cmex squaren.c

或在 MATLAB 5.0 系统的命令窗口中，键入

>> mex squaren.c

就可以生成 MEX 文件，当然这里假设系统的各项参数配置是正确的。之后，在 MATLAB 下就可以调用这个 MEX 文件。例如：

```

>> x = 2;
>> y = squaren(x)
y =
    4

```

MATLAB 5.0 的 API 链接库不仅支持 MATLAB 的最基本数据结构——矩阵，而且支持更广泛的数据结构和数据类型。MATLAB 系统支持的任何数据结构和数据类型都可以用 MEX 子程序传递。下面通过几个例子来说明如何在 MEX 文件中传递其他的数据结构和数据类型。

(1) 字符串变量

下面是计算功能子程序的 MEX 源代码文件，它的作用是将输入的字符串，按反序输出：

```

#include "mex.h"
#include <math.h>
/* 计算功能子程序 */
char *revord(char *buf, int buflen)
{
    int i;
    char *rev_string;
    /* Allocate memory for the reverse order string. */
    rev_string = (char *)mxCalloc(buflen+1, sizeof(char));
    /* Reverse the order of the input string. */
    for (i=0;i<buflen;i++)
        *(rev_string+i) = *(buf+buflen-i-2);
    return(rev_string);
}

```

在这个例子中，用 API 函数 mxCalloc 代替了 C 语言的动态内存分配函数 calloc。函数 mxCalloc 使用 MATLAB 系统的内存管理来分配所需要的内存，在任何需要用 calloc 分配内存的地方，都应该用 mxCalloc 来分配内存。

以下是入口子程序的 MEX 源代码文件：

```

/* 入口子程序 */
void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[])
{
    char *buf, *rev_string;
    unsigned int buflen;
    int status, m, n;
    if (nrhs != 1) {
        mexErrMsgTxt("Only one input argument allowed.");
    }
}

```

```

else if (nlhs != 1) {
    mexErrMsgTxt("Only one output argument allowed.");
}
buflen = (mxGetM(prhs[0]) * mxGetN(prhs[0])) + 1;
/* 分配足够内存 */
buf = mxCalloc(buflen, sizeof(char));
/* 从 prhs[0] 拷贝字符串到 buf */
status = mxGetString(prhs[0], buf, buflen);
if (status == 0)
    mexPrintf("The converted string is %s.\n", buf);
else
    mexErrMsgTxt("Could not convert string data.");
/* 分配足够内存 */
rev_string = mxCalloc(buflen, sizeof(char));
rev_string = revord(buf, buflen);
plhs[0] = mxCreateString(rev_string);
}

```

在入口子程序中,分配的 char 字符型内存 buf 用来将输入的字符串传递给计算功能子程序 revord,函数 revord 返回一个字符型指针,然后 mxCreateString 函数生成返回的字符串 rev_string 的拷贝,并将输出变量 mxArray 指针指向该拷贝值。

(2) 结构型变量

结构(Structure)数据类型是 MATLAB 5.0 定义的一种新的数据类型,像其他的 MATLAB 数据类型一样,结构型数据可以在 C 语言 MEX 文件中传递。

向 MEX 子程序输入结构型数据时,如同传递其他类型的数据,结构的每个域(field)本身是 mxArray 型数据,所以 API 函数 mxGetField 返回一个 mxArray 类型的指针,再将这个指针当作通常的数组类型指针处理。但是,如果希望将结构域值传递到 C 子程序中,则必须调用 API 函数,例如 mxGetString 创建一个字符型指针,并将其指向域值。

MATLAB 5.0 API 函数 mxCreateStructArray 为结构型变量分配内存,但是它不能写入单个域值,而必须用 API 函数 mxSetField。这与字符串内存创建函数 mxCreateString 不同,mxCreateString 函数不仅为字符串分配内存,而且还将 mxArray 指针指向给定值。一般说来,对于较复杂的 MATLAB 数据结构,如块数组和结构,MATLAB 5.0 API 函数不是在同一步里完成内存分配和写入变量之值的任务的。

下面通过一个例子说明如何在 MEX 文件之间传递结构型数据,这个例子是将一批客户数据进行排序后重新输出来。其计算功能子程序如下:

```

#include <matrix.h>
#include <string.h>
#include "mex.h"
#define FMAX 2 /* 结构域的个数 */

```

```

#define NMAX 50 /* 最大客户数. */
void sortf(char * f1[], char * f2[], int n)
{
    int i, j;
    void swap(char **, char **);
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            if (strcmp(f1[i],f1[j])<0)
                swap(&f1[i],&f1[j]);
                swap(&f2[i],&f2[j]);
    }
    return;
}

void swap(char **x, char **y)
{
    char *tmp;
    tmp = *x;
    *x = *y;
    *y = tmp;
    return;
}

```

以下是入口子程序的源代码：

```

void mexFunction( int nlhs, mxArray * plhs[],
                  int nrhs, const mxArray * prhs[])

int i, j, n, numfields, status, buflen, num_dims;
const int * dims;
const int * dims2;
char * buf[NMAX], * buf2[NMAX], * fnames[FMAX];
mxArray * tmp;
char * tmp2;
/* 检查输入的是否为结构 */
if (!mxIsStruct(prhs[0]))
    mexErrMsgTxt("Input is not a structure.");
/* 求结构的维数 */
num_dims = mxGetNumberOfDimensions(prhs[0]);
/* 结构的大小,域的个数,域名 */

```

```
dims = mxGetSize(prhs[0]);
numfields = mxGetNumberOfFields(prhs[0]);
for (i=0; i<numfields; i++)
    fnames[i]=mxGetFieldNameByNumber(prhs[0], i);

/* 求结构的总域数 */
n = mxGetN(prhs[0]) * mxGetM(prhs[0]);
for (i=0; i<n; i++) {
    tmp = mxGetField(prhs[0], i, fnames[0]);
    if (mxIsString(tmp)) {
        buflen= mxGetN(tmp) * mxGetM(tmp)+1;
        buf[i]= mxCalloc(buflen,sizeof(char));
        status= mxGetString(tmp,buf[i],buflen);
        if (status == 0)
            mexPrintf("The converted string is \n%s.\n",buf[i]);
        else
            mexErrMsgTxt("Could not convert name string.");
    }
    else
        mexErrMsgTxt("Element is not a name (string).");

    tmp = mxGetField(prhs[0], i, fnames[1]);
    if (mxIsString(tmp)) {
        buflen = mxGetN(tmp) * mxGetM(tmp)+1;
        buf2[i] = mxCalloc(buflen, sizeof(char));
        status = mxGetString(tmp, buf2[i], buflen);
        if (status == 0)
            mexPrintf("The converted string is \n%s.\n",buf2[i]);
        else
            mexErrMsgTxt("Could not convert address string.");
    }
    else
        mexErrMsgTxt("Element is not an address (string).");
}

/* 排序... */
sortf(buf, buf2, n);

/* 创建输出结构变量 */
```

```

plhs[0] = mxCreateStructArray(num_dims,dims,numfields,fnames);
/* 装入新的结构值 */
for (i=0; i<n; i++) {
    tmp = mxCreateString(buf[i]);
    mxSetField(plhs[0], i, fnames[0], tmp);
    tmp = mxCreateString(buf2[i]);
    mxSetField(plhs[0], i, fnames[1], tmp);
}
}

```

该源代码文件经编译和链接后,就可以用来对结构变量的元素按名字(如果存在的话)域进行排序。

(3) 块数组变量

块数组也是 MATLAB 5.0 的一种新数据结构或数据类型。同样地,块数组变量也可以在 C 语言 MEX 文件中传递。MATLAB 5.0 提供 API 函数 mxGetCell 和 mxSetCell 访问和管理块数组变量,函数 mxCreateCellArray 创建块数组对象。

下面是一个传递块数组变量的例子,它的功能是对于含有字符串的块截去尾部空格。其计算功能子程序如下:

```

#include <matrix.h>
#include "ctype.h"
#include "string.h"
#include "mex.h"
char *rmblocks(char *buf, int buflen)
{
    int i, count=0;
    char ch, ch_old = '0';
    char *hold;
    /* Count the number of trailing blanks. */
    for (i=0; i<buflen; i++) {
        ch= *(buf+i);
        if (isspace(ch) && isspace(ch_old))
            count++;
        if (!isspace(ch) && ch != NULL)
            count=0; ch_old = ch;
    }
    hold = mxCalloc(buflen-count-2, sizeof(char));
    strncpy(hold, buf, buflen-count-2);
    return(hold);
}

```

其入口子程序如下：

```

void mexFunction( int nlhs, mxArray * plhs[],
                  int nrhs, const mxArray * prhs[])
{
    int m, n, count, i, j, buflen, index, status;
    int subs[2];
    int nsubs=2;
    mxArray * cell_elt_pr, * str_pr;
    char * buf, * tmp;
    int dim=2;

    /* 假设为2×2块数组 */
    plhs[0] = mxCreateCellMatrix(dim, dim);
    if (!mxIsCell(prhs[0])) {
        mexErrMsgTxt("Input is not a cell array.");
    }
    m = mxGetM(plhs[0]);
    n = mxGetN(prhs[0]);
    if (m!=2 || n!=2) mexErrMsgTxt("Input is not 2-by-2.");
    /* 访问每个块 */
    for (i=0;i<n;i++) {
        for (j=0;j<m;j++) {
            subs[0]=i;
            subs[1]=j;
            index = mxCalcSingleSubscript(prhs[0], nsubs, subs);
            cell_elt_pr = mxGetCell(prhs[0], index);
            if (mxIsChar(cell_elt_pr)) {
                buflen = mxGetM(cell_elt_pr)*mxGetN(cell_elt_pr) + 1;
                buf = mxCalloc(buflen, sizeof(char));
                if (buf == NULL)
                    mexErrMsgTxt("Not enough heap space to hold string");
                status = mxGetString(cell_elt_pr, buf, buflen);
                if (status!=0)
                    mexErrMsgTxt("Could not convert string.");
            }
        }
    }
}

```

```
mexPrintf("The converted string is\n%s.\n",buf);

    tmp = rmblanks(buf, buflen);
    str_pr = mxCreateString(tmp);
    mxSetCell(plhs[0], index, str_pr);
}
else
    mxSetCell(plhs[0], index, cell_elt_pr);
}
}
}
```

假设上述的源文件经编译和链接后的 MEX 文件名为 delblank.dll,那么对如下的块变量

```
x =
'hi friend' '... bye-bye !!
[2x2 double] [ 11 ]
```

经过 delblank 函数作用后的结果如下:

```
>> y = delblank(x)
y =
'hi friend' '... bye-bye !!
[2x2 double] [ 11 ]
```

由于块数组的元素本身是 mxArray 类型,这说明 mxGetCell 等函数返回一个 mxArray 型指针。于是,可以用任何能够访问 mxArray 型变量的 API 函数,如 mxGetPr、mxGetString 等对单个的块元素进行访问和操作。

(4) 复数型变量

在 C 语言中,为了管理和使用复数,需要将复数型变量分为实数部分和虚数部分。MATLAB 的 API 提供两个函数 mxGetPr 和 mxGetPi 从一个指向双精度型的指针变量中提出对应数据的实部和虚部,这样就可以向计算功能子程序传递所需的复数实部和虚部。

下面是计算两个复数向量卷积的例子 convect.c,其计算功能子程序为:

```
#include "mex.h"
#include <math.h>
#define COLS 1
void convect(double * xr, double * xi, int mx, double * yr,
             double * yi, int my, double * zr, double * zi)
{
    int i, j;
    for (i=0; i< mx; i++)
        for (j=0; j<my; j++) {
```

```

    * (zr+i+j) = * (zr+i+j) + * (xr+i) * * (yr+j) - * (xi+i)
    * * (yi+j);
    * (zi+i+j) = * (zi+i+j) + * (xr+i) * * (yi+j) + * (xi+i)
    * * (yr+j);
}
return;
}

```

下面则是入口子程序：

```

void mexFunction( int nlhs , mxArray * plhs[],
                  int nrhs , const mxArray * prhs[])
{
    double * xr, * xi, * yr, * yi, * zr, * zi;
    int rows, mx, my;

    if (nrhs < 2)
        mexErrMsgTxt("Two input arguments are required.");
    else if (nlhs > 2)
        mexErrMsgTxt("Only two input arguments are allowed.");

    if (mxGetN(prhs[0]) != 1 || mxGetN(prhs[1]) != 1)
        mexErrMsgTxt("Both inputs must be row vectors.");

    /* 计算向量长度 */
    mx = mxGetM(prhs[0]);
    my = mxGetM(prhs[1]);

    /* 取输入向量的实部和虚部 */
    xr = mxGetPr(prhs[0]);
    xi = mxGetPi(prhs[0]);
    yr = mxGetPr(prhs[1]);
    yi = mxGetPi(prhs[1]);

    /* 分配内存作卷积 */
    rows = mx+my-1;
    zr = mxCalloc(rows, sizeof(double));
    zi = mxCalloc(rows, sizeof(double));

    /* 调用计算功能子程序 */

```

```

    convec(xi, xr, mx, yi, yr, my, zr, zi);

    /* 构造输出变量 */
    plhs[0] = mxCreateDoubleMatrix(rows, COLS, mxCOMPLEX);
    mxSetPr(plhs[0], zr);
    mxSetPi(plhs[0], zi);
}

}

```

经编译和链接后，在 MATLAB 环境下，可以计算两个复数向量的卷积。例如：

```

>> x = [3-1i 4+2i 7-3i]';
>> y = [8-6i 12+16i 40-42i]';
>> z = convc(x, y)
z =
    1.0e+002 *
    -0.1800 - 0.2600i
    -0.9600 + 0.2800i
    -1.3200 - 1.4400i
    -3.7600 - 0.1200i
    -1.5400 - 4.1400i

```

这与 MATLAB 函数 conv.m 的计算结果是相同的。

(5) 多维数值变量

多维数组也是 MATLAB 5.0 的新型数据结构。为了与 C 语言 MEX 文件进行多维数组变量的交换，应该将 MATLAB 多维数值数组中的实部和虚部值用 API 函数的 mxGetPr 和 mxGetPi 传递到 MEX 子程序中。反过来，可以用 API 函数的 mxCreateNumericArray 在 MEX 子程序中创建多维数值数组对象，并传递到 MATLAB 系统中去。

下面通过一个例子说明如何实现这种传递。该例中的子程序将一个 $2 \times 3 \times 2$ 的三维数组的各元素平方后送回到 MATLAB。

注意：数组元素为 8 bit 无符号整数类型，其计算功能子程序如下：

```

#define FIRST_DIM 2
#define SECOND_DIM 3
#define THIRD_DIM 2
#define TOTAL_ELEMENTS (FIRST_DIM * SECOND_DIM * THIRD_DIM)
void sqmat(char *x, int l, int m, int n, char *y)
{
    int i, j, k;
    for (i=0; i<FIRST_DIM; i++)
        for (j=0; j<SECOND_DIM; j++)
            for (k=0; k<THIRD_DIM; k++)

```

```

        * (y+i+j+k) = * (x+i+j+k) * * (x-i+j+k);
    return;
}

```

其入口子程序如下:

```

void mexFunction(int nlhs, mxArray * plhs[],
                 int nrhs, const mxArray * prhs[])
{
    int ndim = 3, dims[3]={FIRST_DIM, SECOND_DIM, THIRD_DIM};
    unsigned char data[] = {9, 7, 5, 2, 6, 3, 4, 8, 2, 1, 10, 5};
    unsigned char * start_of_pr;
    unsigned char * start_of_pry;
    mxArray * array_ptr;
    size_t bytes_to_copy;

    /* 调用计算功能子程序 */
    sqmat(data);

    /* 创建三维数组 */
    array_ptr = mxCreateNumericArray(ndim, dims,
                                    mxUINT8_CLASS, mxREAL);
    if (array_ptr == NULL)
        mexErrMsgTxt("Could not create mxArray.\n");

    /* 向创建的数组中写数组值 */
    start_of_pr = (unsigned char *)mxGetPr(array_ptr);
    bytes_to_copy = TOTAL_ELEMENTS * mxGetElementSize(array_ptr);
    memcpy(start_of_pr, data, bytes_to_copy);
    plhs[0] = array_ptr;
}

```

从前面的一些例子可以看出,计算功能子程序按照 C 语言规则编写,处理 C 语言支持的各种数据结构,完成各项计算功能;而入口子程序的功能是作为 MATLAB 系统与 C 语言子程序之间的接口,处理 MATLAB 向 MEX 子程序传递的变量,用 API 函数从这些输入变量中取出 C 语言子程序,即计算功能子程序所需要的各种数据及其数据结构,负责调用计算功能子程序完成必要的计算任务,按 MATLAB 支持的数据结构创建输出变量对象,并将计算功能子程序的计算结果通过所创建的数据对象传递到 MATLAB 系统中去。

4. 调用 MATLAB 函数和自定义函数

在 MEX 源代码文件中,可以调用 MATLAB 函数和 MEX 文件,其调用过程是通过 API 函数 `mxCallMATLAB` 函数实现的。下面的例子说明如何在 MEX 文件中调用

MATLAB 函数：

```
#include "mex.h"
#include <math.h>
void mexFunction(int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[])
{
    double pr[] = {5.2, 7.9, 1.3, 4.2};
    double pi[] = {3.4, 6.5, 2.2, 9.1};
    mxArray *array_ptr;
    int num_out, num_in;
    mxArray *output_array[2], *input_array[2];

    array_ptr = mxCreateDoubleMatrix(2, 2, mxCOMPLEX);
    memcpy(mxGetPr(array_ptr), pr, 4);
    memcpy(mxGetPi(array_ptr), pi, 4);

    num_out = 0;
    num_in = 1;
    input_array[0] = array_ptr;
    mexCallMATLAB(num_out, output_array, num_in, input_array, "disp");

    num_out = 2;
    num_in = 1;
    input_array[0] = array_ptr;
    mexCallMATLAB(num_out, output_array, num_in, input_array, "eig");

    num_out = 0;
    num_in = 1;
    input_array[0] = output_array[0];
    mexCallMATLAB(num_out, output_array, num_in, input_array, "disp");
}
```

这是一个很简单的例子，在这个程序中没有计算功能子程序。当然，完全可以在计算功能子程序中通过 API 函数 `mexCallMATLAB` 调用 MATLAB 函数或 MEX 文件，但要注意的是，只能调用处理双精度型数据的 MATLAB 函数和 MEX 文件。

4.3.3 FORTRAN 语言 MEX 文件

1. MEX 源文件结构与工作原理

FORTRAN 语言 MEX 文件只接受双精度数值变量和字符串变量的输入。像 MATLAB 的 C 语言 MEX 文件一样, FORTRAN 语言 MEX 文件也是由两部分构成。第一部分是计算功能子程序, 完成 MEX 文件的主要计算任务; 第二部分是入口子程序, 是 MATLAB 系统与计算功能子程序之间的接口。其程序的入口点由函数 `mxFunction` 定义, 函数的参数是 `prhs`、`nrhs`、`plhs`、`nlhs`。其中, `prhs` 是指向右端输入变量的 `mxArray` 类型指针; `nrhs` 是右端输入变量的个数; `plhs` 是指向左端输出变量的 `mxArray` 类型指针; 而 `nlhs` 则是左端输出变量的个数。在入口子程序中要调用计算功能子程序, 完成必要的计算任务。

入口子程序和计算功能子程序可以分开, 也可以合并在一起。经过 FORTRAN 编译器编译和链接器连接后, 即可以在 MATLAB 系统中像调用 M 文件一样调用 FORTRAN 语言 MEX 文件。假设有一个名为 `fun` 的 FORTRAN 语言 MEX 文件, 那么下面的 MATLAB 调用语句

```
>> [C, D] = fun(A, B)
```

的工作过程如下:

首先, MATLAB 系统将变量 A 和 B 作为 `fun` 函数的输入参数传递给 `fun` MEX 文件, 也即将指针 `prhs[1]` 和 `prhs[2]` 分别指向 `mxArray` 类型变量 A 和 B, 再用 API 函数的 `mxGetPr`/`mxGetPi` 等通过这些指针取得各项输入值, 并将取得的输入值传递给计算功能 FORTRAN 子程序, 同时在入口子程序中生成必要的 `mxArray` 类型输出变量, 用 API 函数的 `mxCopyPtrTo` 将计算功能子程序的计算结果拷贝到 `mxArray` 类型输出变量中, 最后用输出指针 `plhs(1)` 和 `plhs(2)` 把计算结果传递给 MATLAB 的新变量 C 和 D。

MATLAB API 函数的工作对象仅有 `mxArray` 类型一种。由于在 FORTRAN 77 中没有办法生成新的数据类型和数据结构, MATLAB 只能向 FORTRAN 子程序传递一种标识, 称为指针。在 FORTRAN 子程序内, 用 API 函数从这个指针中获取 `mxArray` 变量的各项信息。

在 FORTRAN 子程序中使用指针时, 有以下两点值得注意:

- `%val` 构造

`%val` 构造是对 FORTRAN 77 和 FORTRAN 90 的扩展。如果用户使用的 FORTRAN 编译器支持 `%val` 构造, 那么可以直接使用一种类型的指针; 即数据指针(例如 API 函数的 `mxGetPr` 或 `mxGetPi` 返回的指针), 可以用 `%val` 将指针的内容直接传递到子程序的类型为双精度矩阵参数变量中; 如果 FORTRAN 编译器不支持 `%val` 构造, 那么用户只能用 API 的 `mxCopy` 类的函数, 例如 `mxCopyPtrToReal8`, 访问指针所指的内容。

- 变量说明

为了正确地使用指针, 必须说明指针的类型。例如在有些系统平台上, 指针必须是 `integer*4`。如果指针的类型说明出现错误, 可能会导致程序崩溃。

入口子程序函数说明形式如下:

```
subroutine mexFunction(nlhs, plhs, nrhs, prhs)
integer plhs(*), prhs(*)
integer nlhs, nrhs
```

C 必要的 FORTRAN 代码 ...

例如,如果要调用一个名为 funmexf 的 MEX 文件,则在 MATLAB 环境中其命令形式为

```
>> [u, v] = funmexf(x, y, z)
```

随后,MATLAB 从 funmexf 的 mexFunction 处开始执行,且参数传递如下所示:

```
nlhs = 2
nrhs = 3
plhs = (指针→'NULL'
         (指针→'NULL')
prhs = (指针→ x
         (指针→ y
         (指针→ z
```

在执行子程序调用之前,由于输出变量 u 和 v 的对象还没有创建,所以参数 plhs 的各指针指向的地址暂未定义。在入口子程序中,MATLAB 根据需要创建 mxArray(或 Matrix)类型输出变量,其地址分别保留在 plhs(1)和 plhs(2)中,再通过两个指针,就可以在计算功能子程序使用这些变量作计算。

为了保证程序能正确地运行,在入口子程序中,必须进行输入变量个数,类型,维数检测等工作。对于不符合子程序输入要求的调用,必须发出警告信息,这可以利用 API 函数的 mexErrMsgTxt 做。

MATLAB 的 API 函数库中集成了大量的 mxArray 变量访问函数,这些函数包括执行从 mxArray 结构中提取数据,向 mxArray 结构中写入数据,以及在 MATLAB 数组和 FORTRAN 数组之间传递数据的函数等。读者可以参考系统的在线帮助。

2. %val 构造

目前,大多数的 FORTRAN 编译器支持%val 构造。用%val 构造传递指针变量时,它传递的是指针变量所指的参数值,而不是参数变量的地址,类似于 C 语言的引用。如果所用的 FORTRAN 语言编译器不支持%val 构造,那么必须用 MATLAB 的 API 函数将 MATLAB 的 mxArray 之值拷贝到 FORTRAN 的数组中。例如,在入口子程序中调用计算功能子程序时,如果是用支持%val 构造的 FORTRAN 编译器进行编译,那么下面的调用是方便的:

```
call submexf(%val(A), %val(B), %val(C), %val(D))
```

上述语句直接将 mxArray 型变量 A 和 B 的值传递到 submexf 子程序。但是,如果 FORTRAN 编译器不支持%val 构造,则子程序的调用过程就要复杂一些,必须将要传递的数据从 mxArray 类型变量中取出后,再传递给 submexf 子程序。例如:

```
real * 8 ar, br, cr, dr
```

C 将指针变量指向的值拷贝到临时局部变量

```
call mxCopyPtrToReal8(A, ar, 1)
call mxCopyPtrToReal8(B, br, 1)
```

```

C
C 用临时局部变量调用子程序
call submexf(ar, br, cr, dr)
C
C 将计算结果传递到输出变量
call mxCopyReal8ToPtr(cr, C, 1)
call mxCopyReal8ToPtr(dr, D, 1)

```

3. 例子

(1) 一个简单例子

首先,从一个简单例子开始说明如何编写 FORTRAN 语言 MEX 文件(子程序)。下面是一个 FORTRAN 子程序:

```

subroutine timestwo(y, x)
real * 8 y, x
y = 2.0 * x
return
end

```

这个子程序将每个输入的数值乘以 2 以后再输出。为了能在 MATLAB 中调用这个 FORTRAN 子程序,必须编写如下的 MEX 文件:

```

subroutine timestwo(y, x)
real * 8 x, y
integer m, n
y = 2.0 * x
return
end

```

C 入口子程序

```

mexFunction(nlhs, plhs, nrhs, prhs)
integer plhs(*), prhs(*)
integer nlhs, nrhs
integer m,n, size, xp, yp
integer mxGetM, mxGetN, mxGetPr, mxCreateFull, mxIsNumeric
real * 8 x, y

```

C 检测输入变量

```

if (nrhs .ne. 1) then
  call mexErrMsgTxt('Only one input argument allowed.')
elseif (nlhs .ne. 1) then
  call mexErrMsgTxt('Only one output argument allowed.')
endif

```

C 取得输入变量的大小值

```

m = mxGetM(prhs(1))
n = mxGetN(prhs(1))
size = m * n
C 检测输入变量的类型
if (mxIsNumeric(prhs(1)) .eq. 0) then
    call mexErrMsgTxt('Input must be a number.')
endif
C 生成输出变量的内存空间
plhs(1) = mxCreateFull(m, n, 0)
yp = mxGetPr(plhs(1))
xp = mxGetPr(prhs(1))
call mxCopyPtrToReal8(xp, x, size)
C 调用计算功能子程序
call timestwo(y, x)
C 向 MATLAB 传递计算结果
call mxCopyReal8ToPtr(y, yp, size)
return
end

```

假设用 FORTRAN 编译器编译好上述子程序,且 MEX 子程序名为 timestwo,那么在 MATLAB 系统中可以直接调用该子程序。例如:

```

>> x = 2;
>> y = timestwo(x)
y =
     4

```

(2) 字符串变量

下面这个例子用于说明如何在 MATLAB 系统和 FORTRAN 语言 MEX 文件之间传递字符串变量,子程序将一个输入字符串反序地输出来:

```

subroutine revord(x, strlen, y)
character x(100), y(100)
integer strlen
C
do 10 i=1,strlen
    y(i) = x(strlen-i+1)
10 continue
return
end

```

其入口子程序如下:

拷贝字符串值。假设上述 MEX 文件编译为 revord, 那么, 便有

```
x = 'hello Gao'
y = revord(x)
y =
    oaG olleh
```

下面再看一看如何传递字符串数组。在 FORTRAN 子程序与 MATLAB 之间传递字符串数组稍稍复杂一点。因为在 MATLAB 系统中, 矩阵变量的元素是按列的顺序存储的, 所以必须保证字符串数组的维数与 FORTRAN 语言 MEX 文件的字符串数组各维数一致。关键是要将字符串数组(字符矩阵)的行列维数反序。

在下面这个例子中, MEX 子程序向 MATLAB 系统传送一个字符串数组(字符矩阵):

```
subroutine mexFunction(nlhs, prhs, nrhs, phrs)
integer nlhs, nrhs
integer plhs(*), prhs(*)
integer ii, stat
integer p_strm
character thestring * 130
character string(5) * 26
C 构造字符串数组
string(1) = 'Dr. Junbin Gao'
string(2) = 'Department of Mathematics'
string(3) = 'HUST'
string(4) = 'Wuhan 430074, Hubei'
string(5) = 'People's Republic of China'
C 拼接成一个字符串
thestring = string(1)
do 10 ii=2,6
    thestring = thestring(:,((ii-1)*26)) // string(ii)
10 continue
C 生成传向 MATLAB 的字符串矩阵, 列数为5, 行数为26
p_strm = mxcreatestring(thestring)
stat = mxsetm(p_strm, 26)
stat = mxsetn(p_strm, 5)
stat = mxsetname(p_strm, 'myaddress')
C 传递到 MATLAB
stat = mexputmatrix(p_strm)
C 在 MATLAB 中求转置
call mexevalstring('myaddress=myaddress';')
end
```

在 MATLAB 系统中,运行这个子程序的结果是在 MATLAB 工作空间中建立一个字符串数组变量 myaddress。

(3) 矩阵变量

在 MATLAB 系统中,可以向 FORTRAN 语言 MEX 文件传递矩阵变量,也可以接受 MEX 文件输出的矩阵变量。一般用 API 函数 mxGetPr 和 mxGetPi 等管理和传递 mxArray 类型的变量。下面的 MEX 例子接受矩阵变量的输入,经过平方计算处理后,向 MATLAB 输出矩阵变量:

```
subroutine matsq(y, x, m, n)
real * 8 x(2,3), y(2,3)
integer m, n
do 20 i=1,m
    do 10 j=1,n
        y(i,j) = x(i,j) * * 2
10    continue
20 continue
return
end
```

它的入口子程序如下:

```
subroutine mexFunction(nlhs, plhs, nrhs, prhs)
integer plhs(*), prhs(*)
integer nlhs, nrhs
integer m, n, size, xp, yp
integer mxGetM, mxGetN, mxGetPr, mxCreateFull, mxIsNumeric
real * 8 x(3,2), y(3,2)
C 检测输入输出变量个数
if (nrhs .ne. 1) then
    call mexErrMsgTxt('Only one input argument allowed.')
elseif (nlhs .ne. 1) then
    call mexErrMsgTxt('Only one output argument allowed.')
endif
C 检查是否为数值矩阵
if (mxIsNumeric(prhs(1)) .eq. 0) then
    call mexErrMsgTxt('Input must be a numeric matrix.')
endif
C 取输入矩阵的维数及大小
m = mxGetM(prhs(1))
n = mxGetN(prhs(1))
size = m * n
```

```

C 生成 MATLAB 数组变量及取输入值
plhs(1) = mxCreateFull(m, n, 0)
yp = mxGetPr(plhs(1))
xp = mxGetPr(prhs(1))
call mxCopyPtrToReal8(xp, x, size)
C 调用子程序
call matsq(y, x, m, n)
C 向 MATLAB 输出变量传送数据
call mxCopyReal8ToPtr(y, yp, size)
return
end

```

API 函数 `mxCopyPtrToReal8` 用于将 MATLAB 的 `mxArray` 类型数据拷贝到 FORTRAN 矩阵变量, 而函数 `mxCopyReal8ToPtr` 将 FORTRAN 矩阵数值传递给 MATLAB 的 `mxArray` 类型的变量。

(4) 复数型变量

在 FORTRAN 中, 由于自定义了复数类型, 所以从 MATLAB 向 FORTRAN 子程序传递复数变量时, 可以不用将复数的实部和虚部分开。先用 API 函数 `mxGetPr` 和 `mxGetPi` 取得输入变量的实部和虚部数据的指针, 再用 API 函数 `mxCopyPtrToComplex16` 就可以将复数据装入到一个 FORTRAN 复矩阵中。

下面的例子是计算两个长度为3的复向量的卷积:

```

subroutine convec(xx, yy, zz, s1, s2)
complex * 16 xx(3), yy(3), zz(5)
integer s1, s2, index
C
zz(1) = (0.0,0.0)
do 20 i=0,s1
    do 10 j=0,s2
        index=i+j+1
        zz(index) = zz(index)+xx(i+1) * yy(j+1)
10    continue
20 continue
return
end

```

入口子程序如下:

```

subroutine mexFunction(nlhs, plhs, nrhs, prhs)
integer plhs(*), prhs(*)
integer nlhs, nrhs
integer m1, n1, m2, n2, sizex, sizey, sizez

```

```

integer mxGetM, mxGetN, mxIsComplex
integer mxCreateFull
complex * 16 x(3), y(3), z(5)
integer xpr, xpi, ypr, ypi, zpr, zpi

C
C 检测输入输出变量个数
if (nrhs .ne. 2) then
    call mexErrMsgTxt('Only two input arguments allowed.')
elseif (nlhs .ne. 1) then
    call mexErrMsgTxt('Only one output argument allowed.')
endif

C 检测变量类型
if (mxIsComplex(prhs(1)) .ne. 1) then
    call mexErrMsgTxt('Input #1 is not complex.')
elseif (mxIsComplex(prhs(2)) .ne. 1) then
    call mexErrMsgTxt('Input #2 is not complex.')
endif

C
m1 = mxGetM(prhs(1))
n1 = mxGetN(prhs(1))
m2 = mxGetM(prhs(2))
n2 = mxGetN(prhs(2))
if (m1 .ne. 1 .and. n1 .ne. 1) then
    call mexErrMsgTxt('Input #1 is not a vector.')
elseif (m2 .ne. 1 .and. n2 .ne. 1) then
    call mexErrMsgTxt('Input #2 is not a vector.')
endif

C
sizex = m1 * n1
sizey = m2 * n2

C 生成输出变量
sizez = sizex + sizey - 1
plhs(1) = mxCreateFull(sizez, 1, 1)
zpr = mxGetPr(plhs(1))
zpi = mxGetPi(plhs(1))
xpr = mxGetPr(prhs(1))
xpi = mxGetPi(prhs(1))
ypr = mxGetPr(prhs(2))
ypi = mxGetPi(prhs(2))

```

```

ypt = mxGetPi(prhs(2))
C 拷贝数据
call mxCopyPtrToComplex16(xpr, xpi, x, sizex)
call mxCopyPtrToComplex16(ypr, ypi, y, sizey)
C 调用计算功能子程序
call convec(x, y, z, sizex-1, sizey-1)
C 拷贝输出数值到 MATLAB 数组
call mxCopyComplex16ToPtr(z, zpr, zpi, sizez)
return
end

```

(5) 稀疏矩阵变量

MATLAB 5.0 提供了一批 API 函数, 用于 MATLAB 系统与 MEX 文件进行稀疏矩阵变量的交换和管理。在 MATLAB 系统的内部管理下, 对每一个稀疏矩阵变量对象定义了几个特殊的参数, 如 ir、jc、nzmax 等。下面给出一个例子, 说明如何利用这些参数来管理稀疏矩阵。这段程序的作用是将一个稀疏矩阵的内容, 打印到一个 ASCII 文件中:

```

subroutine mexFunction(nlhs, plhs, nrhs, prhs)
integer nlhs, nrhs, plhs(*), prhs(*)
integer nzmax, N, pr(10), ir(10), jc(10), xrows(10), xcols(10)
C 检测输入变量保证是稀疏矩阵
if (mxIsSparse(prhs(1)).eq. 0) then
    call mexErrMsgTxt('Input is not a sparse matrix.')
endif
C
C 取各种数据指针
nzmax = mxGetNzmax(prhs(1))
pr = mxGetPr(prhs(1))
ir = mxGetIr(prhs(1))
jc = mxGetJc(prhs(1))
C 拷贝数据到 FORTRAN 矩阵
call mxCopyPtrToReal8(pr, x, nzmax)
call mxCopyPtrToInteger4(ir, xrows, nzmax)
call mxCopyPtrToInteger4(jc, xcols, nzmax)
C 调用计算功能子程序
call printsparse(x, xrows, xcols, nzmax)
return
end
C 计算功能子程序
subroutine printsparse(x, xrows, xcols, nzmax)

```

```

real * 8 xcols(10)
integer nzmax, N, i, j, xrows(10), xcols(10)
C 打开文件供写入稀疏矩阵内容
open(unit=8, file='sparse.dat', status='new')
do 10 i=1,nzmax
    write(8,111), xrows(i), xcols(i), x(i)
111   format('The contents of Row ',I3,' Column ',I3,' are X ',F6.2)
10 continue
close(8)
return
end

```

注意:API 函数的 mxCopyPtrToInteger4 将 ir 和 jc 向量中的数据拷贝到 FORTRAN 矩阵变量中,只有在这种情况下才能调用这个函数。一般说来,当向 FORTRAN 语言 MEX 文件传递整数时,要注意到 MATLAB 是将整数作为双精度的实数存储的,所以通常用函数 mxCopyPtrToReal8 拷贝 MATLAB 整数值到 FORTRAN 语言 MEX 文件中。

(6) MATLAB 函数调用

与 C 语言 MEX 文件一样,在 FORTRAN 语言 MEX 文件中也可以直接调用 MATLAB 的 M 文件或其他的 MEX 文件,只要用 API 函数 mexCallMATLAB 就可以达到这个目的。当然,一般情况下只能调用那些以双精度数据为计算对象的 MATLAB 函数。下面是一个调用 MATLAB 函数的子程序例子。在子程序中,首先构造一个矩阵,然后计算该矩阵的特征值和特征向量。请读者自己分析下列子程序:

```

subroutine mexFunction(nlhs, plhs, nrhs, prhs)
integer nlhs, nrhs, plhs(*), prhs(*)
call EIGS(nlhs, plhs)
return
end
subroutine EIGS(nlhs, plhs)
integer nlhs
integer plhs(*)
integer mxCreateFull, mxGetPr, mxGetPi
C 定义所需变量
integer m, n, imagflag, mlhs, mrhs, mn
C
integer x, xr, xi, lhs(2), rhs
integer evaluer, evaluatei
C 调用子程序
call fill(xreal, ximag)
m = 4

```

```
n = 4
mn = m * n
imagflag = 1
C 定义复矩阵
x = mxCreateFull(m, n, imagflag)
C 取矩阵实部与虚部的指针
xr = mxGetPr(x)
xi = mxGetPi(x)
C
call mxCopyReal8ToPtr(xreal, xr, mn)
call mxCopyReal8ToPtr(ximag, xi, mn)
C 用 MATLAB 函数 EIG 求特征值和特征向量
mrhs = 1
mlhs = 2
call mexCallMATLAB(mlhs, lhs, mrhs, x, 'eig')
C 显示原矩阵
mlhs = 0
if (nlhs .eq. 0) then
    call mexCallMATLAB(mlhs, lhs, mrhs, x, 'disp')
endif
C 显示特征值
rhs = lhs(2)
if (nlhs .eq. 0) then
    call mexCallMATLAB(mlhs, lhs, mrhs, rhs, 'disp')
endif
C 用两种方法计算逆三角阵
evaluer = mxGetPr(lhs(2))
evaluei = mxGetPi(lhs(2))
C
call mxCopyPtrToReal8(evaluer, xreal, mn)
call mxCopyPtrToReal8(evaluei, ximag, mn)
call invertid(xreal, ximag)
C
call mxCopyReal8ToPtr((xreal, xr, mn)
call mxCopyReal8ToPtr((ximag, xi, mn)
if (nlhs .eq. 0) then
    call mexCallMATLAB(mlhs, lhs, mrhs, x, 'disp')
endif
```

C 释放内存

```
call mxFreeMatrix(x)
call mxFreeMatrix(lhs(2))
if (alhs .ne. 0) goto 1000
call mxFreeMatrix(lhs(1))
goto 1010
1000  plhs(1) = lhs(1)
1010  return
      end
```

C

```
subroutine fill(xr, xi)
double precision xr(4,4), xi(4,4), tmp
integer i, j
1000  do 1000 j=1,4
      do 1000 i=1,j
          xr(i,j) = 4+i-j
          xr(j,i) = xr(i,j)
          xi(i,j) = j-i+1
      xi(j,i) = xi(i,j)
```

C

```
do 1050 j=1,2
  do 1050 i=1,4
    tmp = xr(i,j)
    jj = 5-j
    xr(i,j) = xr(i,jj)
1050  xr(i,jj) = tmp
      return
      end
```

C

```
subroutine invertd(xr, xi)
double precision xr(*), xi(*), tmp
integer i
```

C

C 计算复矩阵逆

C

```
do 1000 i=1,16,5
  tmp = xr(i)* * 2+xi(i)* * 2
```

```

        xr(i) = xr(i)/tmp
1000    xi(i) = -xi(i)/tmp
        return
        end
C
subroutine copyrl(x, y, n)
double precision x(*), y(*)
integer n, i
do 1000 i=1,n
1000  y(i) = x(i)
        return
        end

```

4.4 M 文件 Debugger

从程序设计的角度来讲, MATLAB 语言比其他的程序设计语言在说明结构上要简单得多,但是 MATLAB 有自己严密的语法,用户必须按语法的要求来编写 MATLAB 程序,否则同样会产生一些错误。

许多的语法错误可以在程序执行之前查出,这样的错误也容易处理。由于 MATLAB 是运行解释环境,没有独立的编译过程,所以在程序运行过程中,很可能会碰到在编程时不易发现的错误,而且很难按运行的过程去追踪出错的地方。原因之一是 MATLAB 函数调用对 MATLAB 的工作空间是局部的和封闭的,即使要求 MATLAB 输出中间的计算结果,也很难发现是在什么地方出现错误。

MATLAB 提供了 M 文件的 Debugger 功能,可以对 M 文件函数进行调试。MATLAB 的 Debugger 功能帮助用户确定 MATLAB 程序代码中的错误,可以在函数运行期间的任何时刻用 Debugger 查看 MATLAB 工作空间的变量值,查看函数调用的栈管理,以及逐行地运行 M 文件。

Debugger 为用户提供了命令行交互式接口,可以通过命令窗口的菜单进行操作。

4.4.1 Debugger 主要功能

Debugger 主要是分析 MATLAB 程序中的隐含错误,可以发现和更正如下两类错误。

(1) 语法错误

这类错误主要包括函数名拼写和括号遗漏等错误。当然,这类错误在程序运行时, MATLAB 系统自己可以检测到,并且会指出 M 文件中可能出错的行号。

(2) 运行错误

这类错误通常是算法错误,而在语法上是正确的。这类错误会导致非正常的计算结果,而 MATLAB 系统不会发现出错的地方,这就必须使用 Debugger。

一般说来,运行错误往往难于跟踪,因为当被调用的函数由于出错而退出函数体时,函数局部工作空间的变量就全部消失了。可以用下列技巧发现某些运行错误。例如:

在 M 文件中,将某些语句后的分号去掉,迫使 M 文件输出一些中间计算结果,以便发现可能的算法错误;

在 M 文件中,加入 keyboard 语句。keyboard 语句可以设置程序的断点;

将函数 M 文件改为子程序 M 文件,这样可以在 MATLAB 工作空间查看中间计算结果;

最后一种方法是使用 Debugger。Debugger 比其他技巧优越的地方在于它可以深入到函数工作空间,可以设置或清除断点,按行执行程序等。

4.4.2 Debugger 主要命令

MATLAB 的 Debugger 命令主要有以下几种:

dbstop	设置程序断点
dbclear/dbclear all	清除程序(所有)断点
dbcont	恢复程序运行至程序结束或到另一个断点
dbdown/dbstep in	深入下层局部工作空间
dbstack	列函数调用关系
dbstatus	列出所有断点
dbstep	指定执行步数
dbtype	列出行号内的 M 文件
dbup	向上层改变局部工作空间
dbquit	退出 debugger 状态

4.4.3 Debugger 的使用

如果要检测函数 M 文件是否有某些错误,可以对该函数进行 debugging。在函数中设置断点,帮助分析和发现程序的错误原因。在调试时,若函数子程序遇到断点,则将子程序暂时停下来,并将断点处的命令行显示出来,同时显示键盘输入的提示符,用户可以在该提示符下输入任何合法的 MATLAB 命令。

使用 MATLAB 的 Debugger 命令时,应该记住以下几点:

- (a) Debugger 命令只能对函数 M 文件使用,不能对其他的 M 文件使用;
- (b) M 文件的断点是与编译过的 M 文件相关联的,如果 M 文件被 clear 清除或被重新编辑,那么所有的断点即被取消。

4.4.4 例子

(1) 查看函数 M 文件

第一步 编辑两个 M 文件 test.m 和 test1.m，并开始调试：

```
test.m
function a = test(b)
c = sqrt(b) * cos(b);
a = test1(b, c);

test1.m
function a = test1(b, c)
q = cond(b);
[w, e] = eig(c);
a = w * q;
```

第二步 用 dbtype 命令列出有关函数 M 文件：

```
>> dbtype test
1 function a = test(b)
2 c = sqrt(b) * cos(b);
3 a = test1(b, c);
>> dbtype test1
1 function a = test1(b, c)
2 q = cond(b);
3 [w, e] = eig(c);
4 a = w * q;
>> dbtype cond
1 function y = cond(x)
2 %COND Matrix condition number
3 %COND(X) is the ratio of the largest singular value of X
4 % to the smallest, which is the condition number of X in 2--norm
5
6 % See also RCOND and NORM
7
8 % J. N. Little 11-15-85
9 % Revised 3-9-87, JNL, 2-11-92, LS.
10 % Copyright (c) 1985-92 by the MathWorks Inc.
11
12 if length(x) == 0 % handle null matrix
13 Y = NaN;
14 return
15 end
16 if issparse(x)
17 error('Matrix must be nonsparse.')
```

```

18 end
19 s = svd(x);
20 if any(s == 0) % Handle singular matrix
21     disp('Condition is infinite')
22     y = inf;
23     return
24 end
25 y = max(s). / min(s);

```

第三步 存储变量(可以不做这一步):

» hi = 'hello again';

(2) 设置断点

用 dbstop 命令在 test.m 中设置断点:

» dbstop in test

这条语句使函数 test 在第一条可执行语句前暂停下来, 这与下面的语句是等价的:

» dbstop at 2 in test

再在 cond 文件中的第19行处设置断点:

» dbstop at 19 in cond

(3) 启动 M 文件和查看堆栈

第一步 在设置断点之后, 执行 test。根据前面的设置, 该程序的执行停留在第二行。

例如:

```

» test(magic(3));
2 c = sqrt(b) * cos(b);

```

第二步 当函数执行暂停时, 用 dbstack 命令查看函数调用时的堆栈信息。由于函数刚启动, 故 dbstack 回应的信息是

» dbstack

In pathname test.m at line 2

第三步 调用 dbcont 命令, 继续执行函数, 程序将停留在 cond 的第19行上:

» dbcont

19 s = svd(x);

第四步 再用 dbstack 命令, 回应的信息将是

» dbstack

In pathname cond.m at line 19

In pathname test1.m at line 2

In pathname test.m at line 3

(4) 查看工作空间和有关的变量

第一步 检查 cond 的工作空间中的变量:

» who

Your variables are

x y

第二步 检查变量 x 的内容:

>> x

x =

8	1	6
3	5	7
4	9	2

第三步 用 dbstep 执行 cond 中第19行,暂停在第20行:

>> dbstep

20 if any (s == 0) %Handle singular matrix

第四步 检查变量 s:

>> s

s =

15.000	6.9282	3.4641
--------	--------	--------

第五步 再次查看 cond 的工作空间,此时新增了变量 s:

>> who

Your variables are

s x y

(5) 改变工作空间

第一步 转到调用 cond 的函数的工作空间:

>> dbup

In workspace belonging to pathname test1.m.

第二步 查看 test1 的工作空间:

>> who

Your variables are

a b c

第三步 检查变量 b 的内容:

>> b

b =

8	1	6
3	5	7
4	9	2

第四步 检查变量 c 的内容:

>> c

c =

-3.0026	-0.4199	2.4503
-4.1951	-0.8405	2.2478
-4.1854	0.6431	3.5935

第五步 检查变量 a 的内容：

```
>> a  
a =  
[]
```

这是由于没有对变量 a 赋值，故为空矩阵。

第六步 改变工作空间到函数 test 的工作空间，即调用 test1 的函数的工作空间：

```
>> dbup  
In workspace belonging to pathname test.m
```

第七步 改变工作空间到 MATLAB 的工作空间，即调用函数 test 的工作空间：

```
>> dbup  
In base workspace  
>> who  
Your variables are  
hi
```

(6) 生成新的变量

第一步 输入新的变量：

```
>> newvar = 123  
newvar =  
123
```

第二步 改变工作空间到 test 的工作空间，此时用 dbdown，转换成被调用函数的工作空间：

```
>> dbdown  
In workspace belonging to pathname test.m
```

第三步 查看 test 的工作空间：

```
>> who  
Your variables are  
a b c
```

第四步 检查变量 b 的内容：

```
>> b  
b =  
8 1 6  
3 5 7  
4 9 2
```

第五步 改变到 test1 的工作空间，然后到 cond 的工作空间：

```
>> dbdown;  
In workspace belonging to pathname test1.m  
>> dbdown  
In workspace belonging to cond.m
```

(7) 逐步执行函数

第一步 用 dbstep 逐步执行 cond 中的语句, 直到 test1:

```
>> dbstep
25 y = max(s), ./min(s);
>> dbstep
End of M-file function cond.
```

```
>> dbstep
3 [w, e] = eig(c);
```

第二步 列出调用顺序关系:

```
>> dbstack
In pathname test1.m at line 3
In pathname test.m at line 3
```

第三步 步入到 test1 的下一条可执行的语句:

```
>> dbstep
4 a = w * q;
```

第四步 连续执行函数直到下一个断点, 或到函数调用结束, 返回 MATLAB 的工作空间:

```
>> dbcont
ans =
-2.0428 -2.0030-0.7944i -2.0030+0.7944i
-3.6832 0.5115-0.6343i 0.5115+0.6343i
-1.0056 -3.1257-1.9165i -3.1257+1.9165i
```

第五步 查看 MATLAB 的工作空间:

```
>> who
Your variables are
ans    newvar
```

(8) 停止 Debugger

一旦发现程序问题所在, 用户可以在任何时候终止 Debugger。使用 dbquit 命令即可退出 Debugger 状态, 并返回到 MATLAB 工作空间中。

第一步 用 dbstatus 证实断点所在:

```
dbstatus test
Breakpoints are on lines 2
```

第二步 再启动函数:

```
>> test(magic(3))
2 c = sqrt(b) * cos(b);
```

如果用户怀疑程序的问题是由于 test 工作空间中的变量 b 引起的, 可以查看 b 的数值内容, 即

```
>> b
```

b =

8	1	6
3	5	7
4	9	2

第三步 当用户认为已经无任何错误时,就可以使用 dbquit 退出 Debugger 状态,回到 MATLAB 的工作空间中。

注意:dbquit 不清除断点。为了清除断点,在使用 dbquit 之前,要使用 dbclear 命令清除所有的断点。

第五章 MATLAB GUI 程序设计

经典的用户界面定义为用户与计算机之间的交互通信联系的平台。但在最近几年内,这种概念发生了巨大的变化,出现了多种形式的人机交互方式,从命令行的交互方式转变到以图形界面为主的交互形式。现在,图形界面已在人机交互方式中占主导地位,这主要是由于它给用户带来了操作和控制的方便与灵活性。

图形用户界面(GUI)是指由各种图形对象,如由菜单栏、列表框、控制按钮等组成的用户界面。在这种用户界面下,用户的命令和用户对程序的控制是通过“选择”各种图形对象来实现的。

MATLAB 也提供了在 MATLAB 应用程序中加入 GUI 的功能。本章的目的是阐述 MATLAB 的图形用户界面特性,并说明如何在应用程序中加入一些必要的图形用户界面单元,以方便其他用户使用。

本章中阐述的 GUI 特性是 MATLAB 图形句柄系统的子系统。在本书第三章中已经较详细地介绍了 MATLAB 图形句柄系统。为方便起见,再回顾一下与图形对象相关的几个概念:

- (a) 图形对象
- (b) 对象句柄
- (c) 对象属性
- (d) 对象属性值

MATLAB 的 GUI 的基本图形对象分为两类:用户界面控制元对象和用户界面菜单单元对象,简称为控制元和菜单元。设计一个高效的用户界面包含以下的工作:选择恰当的图形对象,并将它们有逻辑地组织起来,使得用户界面容易操作和使用。

5.1 控制元对象及属性

5.1.1 控制元对象类型

控制元对象是这样一类图形界面对象:用户用鼠标在控制元对象上进行操作(如点按鼠标键)时,将会使应用程序作出响应并执行某些预定的功能子程序(Callback)。控制元的操

作结果是可见的,有的可以改变应用程序的初始状态。MATLAB 大约支持 8 种控制元对象,其类别与主要功能如表 5-1 所述。

表 5-1 控制元对象及其功能

控制元名字	功能说明
Push Buttons (按钮)	执行某种特定功能或操作
Check Boxes (检测框)	单个的检测框用来在两种状态之间切换,多个检测框组成一个检测框组时,可使用户在一组状态中作组合式的选择,或称为多选项
Radio Buttons (收音机按钮)	单个收音机按钮用来在两种状态之间切换,多个收音机按钮组成一个收音机按钮组时,用户只能在一组状态中选择单一的状态,或称为单选项
Sliders (滑标条)	在指定的范围内选择一个输入值
Pop-up Menus (弹出式菜单)	一组可选项的列表,用户可以从中选择一项
Static Texts (文字说明)	一组静态的文字说明,例如,可以用它来作为一组控制元的说明标题
Editable Texts (编辑框)	在编辑框中,用户可以用键盘输入某些数据
Frames (边框)	定义一块区域,可以将一组控制元安排在这块区域中,从而对控制元进行逻辑分组
List Boxes (列表框)	这是 MATLAB 5.0 新增加的一个控制元对象。列表框列出字符串表,用户可以在这列表中选取单个列表项或多个列表项。对应于选择,MATLAB 执行相应功能或操作

MATLAB 控制元由与 MATLAB 运行的系统平台相对应的图形对象支持,在不同的操作系统平台上,对应的控制元的名称可能有些不同。

下面将详细地描述 MATLAB 支持的每种控制元,包括每种控制元的应用目的。用户选择按钮、检测框、收音机按钮意味着将鼠标移至该控制元图形,并点按鼠标。在 Windows 系统中,只要点按鼠标左键即可。

1. 按钮 (Push Buttons)

按钮对象是屏幕上的一小块图形对象,在其上简单地标有功能的说明文字,说明文字称为该按钮的标题(Label)。在 MATLAB 系统下,点按按钮对象将使 MATLAB 执行某种预定义的功能或操作,它与 Windows 系统的命令按钮的作用相同。按钮对象如图 5-1(a) 所示。

2. 检测框 (Check Boxes)

检测框的作用是使用户可以在一个或一组备选项中选择一个或多个选项。每个选择框有两种状态,如果检测框被选中,其状态定义为 on,否则其状态定义为 off。检测框组中的每个检测框是相互独立的,可以设置任意多个检测框的状态为 on,构成一个组合式的选择,每组这样的选择对应于应用程序预定义的一种功能或操作。

在 Windows 系统中,检测框是一个小方框加上一条简单说明。小方框中有√符号表示该检测框的状态为 on,否则其状态为 off,如图 5-1(b) 所示。

3. 收音机按钮 (Radio Buttons)

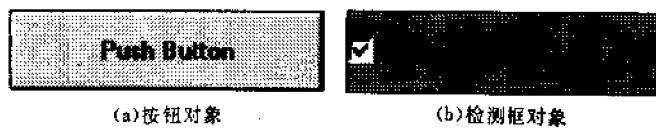


图5-1 控制元对象图形1

单个的收音机按钮的作用与单个的检测框的作用相同。收音机按钮组使得用户在一组可选择项中选择一项。每个收音机按钮可以在两种状态(分别称为 on 与 off 状态)之间切换。每种选择使得应用程序执行某种预定义的功能或操作。

一般说来,收音机按钮组是在多种选择项中唯一地选择一项,亦即在收音机按钮组中,只有一个收音机按钮的状态是 on,其他的收音机按钮的状态是 off。这是收音机按钮组与检测框组的主要区别。

如果在应用程序中需要多组收音机按钮组,为了界面的美观,可以从功能逻辑上将每组分开,使得每组收音机按钮组织在一个边框对象中。

在 Windows 系统中,收音机按钮是一个圆圈加上一条简单说明,称为收音机按钮的标题(Label)。圆圈中有一个圆点时,表示该收音机按钮的状态是 on;如果圆圈中没有圆点,则表示该收音机按钮的状态为 off,如图5-2(a)所示。



图5-2 控制元对象图形2

4. 滑标条 (Sliders)

用户使用滑标条能够输入指定范围内的数量值。为了输入数量值,可以通过移动滑标条的指示钮来设置想要输入的数值。MATLAB 的应用程序根据不同的输入数值,执行预定义的功能和操作。

每个滑标条由3个部分组成:表示允许值范围的滑槽;一个可移动的指示钮,表明在允许值范围内的当前值的相对位置;滑标条两端的箭头。

滑标条的长宽大小比例决定它的外观方向。如果滑标条的宽度大于它的高度,那么滑标条横置在图形窗口中,否则将被竖置在图形窗口中。在 Windows 系统中,滑标条的两端总是带有箭头,如图5-2(b)所示。

用户可以用下列的方法设置或改变滑标条的输入值:

- 在滑槽上用鼠标拖动指示钮(按住鼠标左键移动鼠标),直到所希望的输入数值的相对位置处,然后释放鼠标按钮;
- 用鼠标点按指示钮两端的滑槽部分,每点按一次,指示钮就会向点按的方向移动,移动的距离大约是滑槽长度的10%;
- 用鼠标点按滑标条两端的箭头之一,每点按一次,指示钮就会向箭头指示的方向

移动,移动的距离大约是滑槽长度的1%。

多数情况下,滑标条的输入数值是希望值的近似,用户可以用编辑框来输入必要的精确值。

5. 弹出式菜单 (Pop-up Menus)

弹出式菜单可让用户从一组列项中选择一项作为参数输入。弹出式菜单与后面要介绍的菜单对象是有区别的,弹出式菜单出现在图形窗口内,而菜单对象出现在图形窗口的主菜单栏中。弹出式菜单还可以用来从一组预定义的数值或选项中选择一项输入,而菜单对象往往是对应一些功能选项,确定某种功能操作。

未打开的弹出式菜单只是显示当前的选项(缺省的选项)。当用户打开一个弹出式菜单时,其可选项就会在一个列表栏中出现,然后可以用鼠标点按想要选择的选项,于是被选项被着色。选择好选项后,弹出式菜单自动关闭,并将所作的选择项显示在菜单框中。如果选项列表很长,在 Windows 系统中,选项列表的右端会出现一个滚动条,可以用鼠标滚动条,直到找到想要选择的选项。图 5-3(a)所示的是未打开的弹出式菜单外形,其中有缺省选项;图 5-3(b)所示的是打开的弹出式菜单的列表外形。用户选择某项后,弹出式菜单自动关闭,回到未打开时的外形,此时被选项在菜单栏中。

6. 文字说明 (Static Texts)

文字说明控制对象仅仅用于显示单行的说明文字。例如,用文字说明对象来作为一组其他控制元的标题,给用户必要的提示。又如,标明滑标条的取值范围。用户不能在执行应用程序的过程中改变文字说明对象,故而将其称为静态对象。图 5-4 中的说明文字“Static Text”和“Please Input”都是由文字说明对象生成的。



图 5-3 控制元对象图形

7. 编辑框 (Editable Texts)

用户可以在编辑框中用键盘输入字符串值(字符和数据)。编辑框分为单行与多行编辑框两种。单行的编辑框对象仅仅包含一行字符,多行编辑框可以包含多行字符。用户可以对编辑框中的内容进行编辑、删除、替换等操作。在图 5-4 中,说明文字“Please Input”右端的白色区域为编辑框,“initial input”为编辑框中的缺省输入值,用户可以用键盘修改输入值。

8. 边框 (Frames)

用户可以用边框对象在图形窗口中圈出一块区域,而将一些有关联的控制元组织在这块区域中。边框对象的主要作用是组织出外观更加友好的 GUI。图 5-4 所示的便是用边框对象组成的一个整体。

图 5-4 文字说明、边框、编辑框
控制元对象图形

5.1.2 控制元创建函数

创建控制元对象的基本方法是使用 MATLAB 的函数 `uicontrol`, 该函数的调用形式为
`>> h=uicontrol(hfig,'属性名',属性值,...)`

设计程序时, 可根据需要向函数 `uicontrol` 提供必要的属性名和属性值参数对。MATLAB 系统不区别属性字符串中的大小写字母。例如, '`style`' 和 '`Style`' 都表示是属性 `Style` 的名字。

由于控制元对象都是图形窗口对象的子对象, 所以函数 `uicontrol` 调用中, 第一个参数 `hfig` 应该是某个图形窗口句柄值。所创建的控制元对象将是该图形窗口的子对象, 并出现在该图形窗口中。如果省略这个句柄值参数, 则生成的图形对象将是当前的图形窗口的子对象。如果此时无图形窗口存在, MATLAB 系统将自动创建一个图形窗口, 并在其中创建相应的控制元对象。

`h` 是所创建控制元对象的句柄值。MATLAB 通过这个句柄值来管理该控制元对象, 如同 MATLAB 任何类型的图形对象一样。在 MATLAB 图形句柄系统中, 控制元对象也是一种图形对象, 因此, 对于图形对象的操作方法都能运用于控制元对象。例如, 可以用 `get` 函数取得控制元对象的某些属性的当前值, 用 `set` 函数可以改变和重新设置控制元对象的某些当前的属性值。

与控制元对象相关联的功能操作是通过控制元对象的 `Callback` 属性来定义的, 具体使用方式在 5.1.4 节中介绍。

5.1.3 控制元对象的属性

在 MATLAB 的图形句柄系统中, 可使用属性来对对象的外型和特性进行描述。用户可以在创建控制元对象时, 设定这些属性的属性值, 对未指定的属性值, MATLAB 将使用系统提供的缺省属性值。可以用设置图形对象缺省属性值的方法设置控制元对象的缺省属性, 参见 3.10 节。MATLAB 的控制元对象使用相同的属性类型, 但是这些属性对于不同类型的控制元对象, 其含义不尽相同。

控制元对象的属性分为两大类: 第一类是所有的控制元对象都具有的公共属性; 第二类是把控制元对象作为图形对象所具有的公共属性。下面分别予以介绍。

1. 公共属性

(1) Children 属性

`Children` 属性的取值为空矩阵, 因为控制元对象自己没有子对象。

(2) Parent 属性

`Parent` 属性的取值是某个图形窗口对象的句柄值, 因为控制元对象总是图形窗口对象的子对象, 该句柄值标明了控制元对象所在的图形窗口。如果用 `set` 函数将 `Parent` 属性设置为另一个图形窗口的句柄值, 相当于将控制元对象移到另一个图形窗口中。

(3) Tag 属性

Tag 属性的取值是字符串。它定义了该控制元的一个标识值。定义了 Tag 属性后，在任何程序中都可以通过这个标识值找出该控制元对象。

(4) Type 属性

Type 属性的取值总是 uicontrol，这个属性值标明图形对象的类型。对于控制元对象，其类型就是 uicontrol，用户不能改写这个属性。

(5) UserData 属性

UserData 属性的取值是一个矩阵，缺省值为空矩阵。用户可以在这个属性中保存与该控制元对象相关的重要数据或信息，借此可以达到传递数据或信息的目的。可以用 set 和 get 函数访问该属性。

(6) Visible 属性

Visible 属性的取值是 on(缺省值)或 off。如果该属性值为 off，那么控制元对象在图形窗口中是不可见的，但是该控制元对象仍存在，可以用 set 和 get 函数访问它的各个属性。

2. 基本控制属性

(1) BackgroundColor 属性

BackgroundColor 属性的取值是某些颜色的预定义字符或颜色 RGB 数值，它定义控制元对象区域的背景色，它的缺省颜色是系统定义的浅灰色。

(2) Callback 属性

Callback 属性的取值是字符串，可以是某个 M 文件名或一小段 MATLAB 语句。当用户激活控制元对象(例如，在控制元对象图标上点按鼠标或移动滑动槽标尺)时，应用程序就运行 Callback 属性定义的子程序(callback 调用)，但是 Frames 和 Static Text 控制元对象不定义 Callback 属性。

(3) Enable 属性

Enable 属性的取值是 on(缺省值)、inactive 和 off。当它的取值是 on 时，无论在什么时候激活控制元对象，MATLAB 都执行 Callback 属性定义的 callback 子程序，但如果鼠标是在控制元对象图标区域外的 5 个像素宽的边界范围内按下时，就执行由 ButtonDownFcn 属性定义的 callback 子程序；若 Enable 属性的取值是 inactive，那么当用户在控制元对象图标及 5 个像素边界的范围内点按鼠标时，MATLAB 执行 ButtonDownFcn 定义 callback 子程序；当 Enable 属性的取值是 off 时，控制元对象的标题字是有阴影的，它的 callback 执行功能与 inactive 情形相同，即运行 ButtonDownFcn 定义的 callback 子程序。这样定义是有好处的，例如可以通过定义 ButtonDownFcn 的 callback 子程序来移动控制对象。

(4) Extent 属性

Extent 属性的取值是四元素向量 [0, 0, width, height]，记录控制元对象标题字符的位置大小尺寸，其度量单位由 Units 属性定义。只能读取该属性值，不能改写。

(5) ForegroundColor 属性

ForegroundColor 属性的取值是某些颜色的预定义字符或颜色 RGB 数值，这个属性定义控制元对象标题字符的颜色。标题字符的缺省颜色是黑色。

(6) Max、Min 属性

Max 和 Min 属性的取值都是数值，其缺省值分别是 1 和 0。这两个属性值对于不同的控

制元对象类型,其意义是不同的。以下分别予以介绍:

当 Radio buttons 控制元对象被激活时,它的 Value 属性值为 Max 属性定义的值;当控制元对象处于非活跃状态时,它的 Value 属性值为 Min 属性定义的值;

当 Check boxes 控制元对象被激活时,它的 Value 属性值为 Max 属性定义的值;当控制元对象处于非活跃状态时,它的 Value 属性值为 Min 属性定义的值;

对于 Sliders 控制元对象,Max 属性值必须比 Min 属性值大,Max 定义滑动条的最大值,Min 定义滑动条的最小值;

对于 Editable text 控制元对象,如果 $\text{Max} - \text{Min} > 1$,那么对应的编辑框接受多行字符输入;如果 $\text{Max} - \text{Min} \leq 1$,那么编辑框仅接受单行字符输入;

对于 List boxes 控制元对象,如果 $\text{Max} - \text{Min} > 1$,那么在列表框中允许多项选择;如果 $\text{Max} - \text{Min} \leq 1$,那么在列表框中只允许单项选择;

另外,Frames、Pop-up menus 和 Static Texts 控制元对象不使用 Max 和 Min 属性。

(7) String 属性

String 属性的取值是字符串矩阵或块数组,它定义控制元的标题或选项内容。特别地,对于 Pop-up menus 和 List boxes 这样的多选择的控制元对象,它的 String 属性可以接受多种形式的输入,如字符串块数组、字符串矩阵和用“|”分隔的字符串向量;对于 Editable text 和 Static text 这样的多行文字型控制元对象,String 矩阵的每行,块数组的每块及字符串中的“\n”字符定义文字的行分隔符,即从该处分行。对于 Editable text, String 属性值可以由键盘输入。

(8) Style 属性

Style 属性的取值可以是 pushbutton(按钮:缺省值)、radiobutton(收音机按钮)、checkbox(检测框)、edit(编辑框)、text(文字说明)、slider(滑动条)、frame(边框)、listbox(列表框)和 popupmenu(弹出式菜单)。这个属性定义控制元对象的类型,由相应的值决定。

(9) Units 属性

Units 属性的取值可以是 pixels(像素:缺省值)、normalized(相对单位)、inches(英寸)、centimeters(厘米)和 points(磅)。除了 normalized(相对单位)以外,其他都是绝对单位。这个属性值作为解释 Extent 和 Position 属性值的度量单位。所有单位的度量都是从图形窗口的左下角处开始,在相对单位下,图形窗口的左下角点对应(0, 0),而右上角点对应(1.0, 1.0)。

(10) Value 属性

Value 属性的取值可以是向量值,也可以是数值。它的含义及解释依赖于控制元对象的类型。对于收音机按钮和检测框对象,当它们处于活跃状态时,Value 属性值由 Max 属性值定义,反之由 Min 属性值定义;对于滑动条对象,Value 属性值处于 Min 与 Max 属性值之间,由滑动条标尺位置对应的值定义;对于弹出式菜单对象,Value 属性值是被选项的序号,1 对应第一项。所以由 Value 的值,可知弹出式菜单的选项;同样,对于列表框对象,Value 属性值定义了列表框中高亮度选项的序号,其他的控制元对象不使用这个属性值。

3. 修饰控制属性

(1) FontAngle 属性

FontAngle 属性的取值是 normal(缺省值)、italic 和 oblique, 这个属性值定义控制元对象标题等的字体体态。其值为 normal 时, 选用系统缺省的正字体; 而其值为 italic 或 oblique 时, 使用方头斜字体。

(2) FontName 属性

FontName 属性的取值是控制元对象标题等使用字体的字库名, 必须是系统支持的各种字库。缺省字库是系统的缺省字库。

(3) FontSize 属性

FontSize 属性的取值是数值, 它定义控制元对象标题等字体的字号。字号单位由 FontUnits 属性值定义。缺省值与系统有关。

(4) FontUnits 属性

FontUnits 属性的取值是 points(磅: 缺省值)、normalized(相对单位)、inches(英寸)、centimeters(厘米)或 pixels(像素), 该属性定义字号单位。相对单位将 FontSize 属性值解释为控制元对象图标高度百分比, 其他单位都是绝对单位。

(5) FontWeight 属性

FontWeight 属性的取值是 normal(缺省值)、light、demi 或 bold, 它定义字体字符的粗细。

(6) HorizontalAlignment 属性

HorizontalAlignment 属性的取值是 left、center(缺省值)或 right, 其属性定义控制元对象标题等的调整方向, 即标题在控制元对象图标上居左 left、居中 center、居右 right。

4. 辅助属性

(1) ListboxTop 属性

ListboxTop 属性的取值是数量值, 这个属性只用于 listbox 控制元对象, 它定义了位于列表框中的最上方的字符串在 String 属性中的序号。非整数值被截断为大于该值的最小整数。

(2) SliderStep 属性

SliderStep 属性的取值是 2 元素向量 [minstep, maxstep], 这个属性只对 Slider 控制元对象有定义。maxstep 定义了在滑动槽内点按鼠标时, 滑动槽标尺应该移动的距离相对于滑动槽长度的百分比; minstep 定义了在滑动槽两端箭头上点按鼠标时, 滑动槽标尺应该移动的距离相对于滑动槽长度的百分比。它的缺省值是 [0.01 0.10]。

(3) Selected 属性

Selected 属性的取值是 on 或 off(缺省值)。当 Selected 属性值是 on 时, 如果 SelectionHighlight 属性值也是 on, 那么 MATLAB 就显示一个对象被选中的标记。如果将ButtonDownFcn 定义为设置这个属性为 on, 那么当选中控制元对象时, 控制元对象就会有一个被选中的方框。

(4) SelectionHighlight 属性

SelectionHighlight 属性的取值是 on(缺省值)或 off, 该属性决定控制元对象被选中时, 是否显示被选中的状态。这个属性必须与 Selected 属性配合使用, 共同控制控制元对象被选中时的状态。

5. Callback 管理属性

(1) BusyAction 属性

BusyAction 属性的取值是 `cancel` 或 `queue`(缺省值), 该属性决定 MATLAB 采取的控制中断执行控制元对象的 `callback` 调用的方式。在 MATLAB 环境中, 如果有一个 `Callback` 调用在运行, 那么随后的 `Callback` 调用总是试图中断正在运行的 `Callback` 调用。如果此时 `Interruptible` 属性值为 `on`, 那么 MATLAB 在处理事件队列时, 会中断正在运行的 `Callback` 调用; 如果 `Interruptible` 属性值为 `off`, 那么 `BusyAction` 属性就决定 MATLAB 处理事件的方式: 如果它的值是 `cancel`, 就从事件队列中取消企图运行第二个 `Callback` 调用的事件; 如果它的值是 `queue`, 就将企图运行第二个 `Callback` 调用的事件加入事件队列, 直到当前的 `Callback` 调用完成为止。

(2) ButtonDownFcn 属性

`ButtonDownFcn` 属性的取值是字符串, 一般是某个 M 文件名或一小段 MATLAB 语句。当用户在围绕控制元对象的 5 个像素宽的边界内按下鼠标键时, 程序执行该属性定义的 `Callback` 调用。控制元对象一般是矩形区域, 如果控制元对象的 `Enable` 属性值是 `inactive` 和 `off`, 则当在该区域以及 5 像素宽的边界内按下鼠标键时, 就执行 `ButtonDownFcn` 定义的 `Callback`。如果这个 `Callback` 定义为表达式, 那么计算结果便属于 MATLAB 工作空间。

(3) CreateFcn 属性

`CreateFcn` 属性的取值是字符串, 一般是某个 M 文件名或一小段 MATLAB 语句, 即当 MATLAB 生成控制元对象时, MATLAB 事先应该执行的程序段。这个属性只能按设置缺省值的方式定义(参见 3.10), 例如:

```
>>set(0,'DefaultUicontrolCreateFcn',...
      'set(gcf,"IntegerHandle","off")')
```

从根对象层设置控制元对象的缺省值。对于已经存在的控制元对象, 改变该属性之值, 对该对象无任何影响。MATLAB 在生成控制元对象, 完成所有缺省属性值的设置后, 再执行由 `CreateFcn` 定义的 `Callback`。

如果一个控制元对象 `CreateFcn` 属性定义的 `Callback` 正在运行, 那么该控制元对象的句柄值不可访问, 必须用函数 `gcbo` 取得。

(4) DeleteFcn 属性

`DeleteFcn` 属性的取值是字符串, 一般是某个 M 文件名或一小段 MATLAB 语句。当删除控制元对象时, MATLAB 在清除该对象属性之前, 要执行由 `DeleteFcn` 属性定义的 `Callback`。如果一个控制元对象 `DeleteFcn` 属性定义的 `Callback` 正在运行, 那么该控制元对象的句柄值不可访问, 必须用函数 `gcbo` 取得。

(5) HandleVisibility 属性

`HandleVisibility` 属性的取值为 `on`(缺省值)、`callback` 或 `off`, 这个属性定义控制元对象句柄的可访问权限, 决定其句柄值是否在父对象的 `Children` 属性中出现。如果 `HandleVisibility` 属性值是 `on`, 那么它的句柄值总是可见的; 如果属性值为 `callback`, 那么该句柄值只在该对象的 `Callback` 调用以及 `Callback` 调用的函数内是可见的, 但是命令行内调用的函数则不能访问这样的控制元句柄, 也就是说, 这样的句柄对于该控制元对象自己的

Callback 调用是私有的,这对于保护交互式的程序是十分有用的;如果 HandleVisibility 属性值被设置为 off,那么控制元对象的句柄在任何时刻都是不可见的。如果句柄是不可见的,那么任何查询句柄的函数都不能返回它的值,这些函数包括 get、findobj、gcf、gca、gco、newplot、cla、clf、close 等。当 HandleVisibility 属性值是 callback 或 off 时,这样的控制元对象不会出现在图形窗口对象的 Children 和 CurrentObject 属性中,也不会出现在根对象 CallbackObject 属性中,但是可以将根对象的 ShowHiddenHandles 属性设置为 on,临时地使得所有句柄都是可见的,而不管其 HandleVisibility 属性值是什么。被隐藏的句柄仍是有有效的,如果知道这样的句柄值,一切关于句柄的操作都是合法的。

(6) Interruptible 属性

Interruptible 属性的取值是 on 或 off(缺省值),这个属性决定控制元对象的 Callback 是否可以被随后的 Callback 调用中断。只有由 ButtonDownFcn 和 Callback 属性定义的 Callback 受 Interruptible 属性值的影响。在运行程序时,只有遇到 drawnow、figure、getframe 和 pause 等命令,MATLAB 系统才检查可以中断程序运行的 Callback 调用事件。

5.1.4 例子

1. 创建按钮对象

下面这个例子用来说明如何创建一个按钮对象,与该按钮对象相关联的操作是运行一个函数 M 文件 myplot.m,按钮的文字说明(Label)是 Start Plot,它的位置和大小尺寸的单位是 pixel(像素):

```
>> pbstart=uicontrol(gcf,'Style','pushbutton',...
    'Position',[10 10 100 25],...
    'String','Start Plot',...
    'CallBack','myplot');
```

Style 属性值 pushbutton 指明该控制元对象是按钮,Position 指示创建的按钮对象在当前的图形窗口中的位置及大小,属性 String 的值就是该按钮对象上的说明文字(Label),CallBack 属性定义了当用户用鼠标点按该按钮对象时应用程序应执行的功能和操作。

以下的例子也是创建一个按钮,它的作用是在命令窗口中回应滑标条的输入值,滑标条的句柄值变量是 sli8;

```
>> pbgetval=uicontrol(gcf,'Style','pushbutton',...
    'Position',[10 120 120 25],...
    'String','Get Slider Value',...
    'CallBack','disp(get(sli8,"Value"))');
```

这个例子也说明,属性 CallBack 的值可以是一个 M 文件的名字,也可以是一条由 MATLAB 语句组成的字符串,而 MATLAB 语句中自身的字符串用双引号,这是 MATLAB 的语法要求。

2. 创建检测框

下面这个例子用来说明如何生成两个检测框,用来设置当前坐标系的某些属性值,其

中,文字说明对象给出了控制元的一个标题,控制元对象的位置和大小的尺寸单位是相对单位(normalized),CallBack 定义的操作是设置一些属性值:

```
%  
=>txtaxes = uicontrol(gcf, 'Style', 'text', ...  
    'Position', [1 4 25 1], 'Units', 'normalized', ...  
    'String', 'Set Axes Properties');  
  
%  
=>cbbbox = uicontrol(gcf, 'Style', 'checkbox', ...  
    'Position', [1 3 25 1], 'Units', 'normalized', ...  
    'String', 'Box = on', ...  
    'CallBack', ['set(gca, "Box", "off"),', ...  
        'if get(bbbox, "Value") == 1, ', ...  
        'set(gca, "Box", "on"), ', ...  
        'end']);  
  
%  
=>cbtdir = uicontrol(gcf, 'Style', 'checkbox', ...  
    'Position', [1 2 25 1], 'Units', 'normalized', ...  
    'String', 'TickDir = out', ...  
    'CallBack', ['set(gca, "TickDir", "in"),', ...  
        'if get(cbtdir, "Value") == 1, ', ...  
        'set(gca, "TickDir", "out"), ', ...  
        'end']);
```

读者可以试着运行一下上述这段程序,看看其运行后的效果。

3. 创建收音机按钮

下面的例子用来说明如何创建具有两个收音机按钮的收音机按钮组,用来设置当前的坐标系坐标轴刻度标记的方向,即刻度标记是朝坐标系内还是朝坐标系外,两者居其一。每个收音机按钮检查另一个收音机按钮的状态,保证只有一个按钮的状态是 on:

```
%  
=>txttdir = uicontrol(gcf, 'Style', 'text', ...  
    'String', 'Select Tick Direction', ...  
    'Position', [200 100 150 20]);  
  
%  
=>tdin = uicontrol(gcf, 'Style', 'radio', ...  
    'String', 'In', ...  
    'Position', [200 75 150 25], ...  
    'Value', 1, ...  
    'CallBack', ['set(tdin, "Value", 1), ', ...  
        'set(tdout, "Value", 0), ', ...
```

```

    ' set(gca, "TickDir", "in"),']);
>>tdout = uicontrol(gcf, 'Style', 'radio', ...
    'String', 'out', ...
    'Position', [200 50 150 25], ...
    'CallBack', ['set(tdout, "Value", 1),',...
        'set(tdin, "Value", 0),',...
        'set(gca, "TickDir", "out")']);

```

CallBack 执行的结果保证只有一个收音机按钮的状态为 on, 因为收音机按钮的 Value 属性是这样定义的: 如果收音机按钮的状态是 on, 那么属性 Value 的值是收音机按钮的另一个属性 Max 的属性值, 该属性值的缺省值是 1; 如果状态是 off, 那么属性 Value 的值是收音机按钮的另一个属性 Min 的属性值, 它的缺省值是 0。这样, 通过对属性 Value 设定不同的值, 就可以得到收音机按钮的不同状态。请读者运行上述的语句, 观察所创建的收音机按钮的状态。

4. 创建滑标条

下面的例子用来说明如何创建两个滑标条, 它们分别用于设置3维图形(或坐标系)视点的经度和纬度值, 利用文字说明对象标出滑标条的数值范围以及当前值:

```

>> fig = gcf;
>> clf;
%
>> sliazm = uicontrol(fig, 'Style', 'slider', ...
    'Position', [50 50 120 20], ...
    'Min', -90, 'Max', 90, 'Value', 30, ...
    'CallBack', ['set(azmcur, "String",',...
        'num2str(get(sliazm, "Value"))),',...
        'set(gca, "View",',...
        '[get(sliazm, "Value"),...
        get(slielv, "Value")])']);
>> slielv = uicontrol(fig, 'Style', 'slider', ...
    'Position', [240 50 120 20], ...
    'Min', -90, 'Max', 90, 'Value', 30, ...
    'CallBack', ['set(elvcur, "String",',...
        'num2str(get(slielv, "Value"))),',...
        'set(gca, "View",',...
        '[get(sliazm, "Value"),...
        get(slielv, "Value")])']);
%
>> azmmin = uicontrol(fig, 'Style', 'text', ...
    'Position', [20 50 30 20], ...

```

```

'String', num2str(get(sliazm, 'Min'))));
>> elvmin = uicontrol(fig, 'Style', 'text', ...
    'Position', [210 50 30 20], ...
    'String', num2str(get(slielv, 'Min')));
%
>> azmmax = uicontrol(fig, 'Style', 'text', ...
    'Position', [170 50 30 20], ...
    'String', num2str(get(sliazm, 'Max')), ...
    'HorizontalAlignment', 'right');
>> elvmax = uicontrol(fig, 'Style', 'text', ...
    'Position', [360 50 30 20], ...
    'String', num2str(get(slielv, 'Max')), ...
    'HorizontalAlignment', 'right');
%
>> azmlabel = uicontrol(fig, 'Style', 'text', ...
    'Position', [50 80 65 20], ...
    'String', 'Azimuth');
>> elvlabel = uicontrol(fig, 'Style', 'text', ...
    'Position', [240 80 65 20], ...
    'String', 'Elevation');
%
>> azmcur = uicontrol(fig, 'Style', 'text', ...
    'Position', [120 80 50 20], ...
    'String', num2str(get(sliazm, 'Value')));
>> elvcur = uicontrol(fig, 'Style', 'text', ...
    'Position', [310 80 50 20], ...
    'String', num2str(get(slielv, 'Value')));

```

5. 创建弹出式菜单

用户可以创建一个列表式的弹出式菜单, 菜单的可选项只需在 String 属性中设置, 每项之间用竖线字符“|”隔开, 并用单引号将所有的选项括起来。

弹出式菜单对象的 Value 属性中的值是弹出式菜单列表中选项的序号。如果用户选列表中的第二项, 那么 Value 的属性值就是2。

下面的例子用来说说明如何创建一个弹出式菜单, 其列表中包含一组可供选择的颜色。当某种颜色被选中时, CallBack 就将图形窗口的背景色设置为该种颜色:

```

>> popcol = uicontrol(gcf, 'Style', 'popup', ...
    'String', 'red|blue|green|yellow', ...
    'Position', [100 100 100 80], ...
    'CallBack', ['cocol = ["R", "B", "G", "Y"];',...

```

```
' set(gcf,'Color', cbcoll(get(popcol, ...
    "Value")));']);

```

6. 创建编辑框

下面这个例子用来说明如何创建一个多行的编辑框对象, 初始由4行文字组成, 此时 Max 属性可以是任何大于1的整数值, 这是由于属性 Min 的缺省值是0。属性 Max 与属性 Min 之差小于或等于1时, 为单行编辑框:

```
>> edmulti = uicontrol(fig, 'Style', 'edit', ...
    'String', 'change|these|four|lines', ...
    'Position', [10 200 75 100], 'Max', 2);
```

在下列操作之后, MATLAB 将执行编辑框对象的 callback 属性定义的功能或操作:

- (a) 用户改变编辑框中的输入值, 并将鼠标移出编辑框控制对象;
- (b) 对单行编辑框, 用户键入 Return 键, 而不论编辑框中的值是否被改变;
- (c) 对单行和多行编辑框对象, 用户按下 Ctrl 键, 再按回车键(对 Windows 系统), 而不论编辑框中的值是否被改变; 在多行编辑框中, 用户按下回车键, 可以输入下一行字符。

7. 创建边框对象

当需要用边框对象组织控制元对象时, 必须在定义控制元对象之前创建边框对象, 或者说边框对象必须覆盖该组中所有的控制元对象。为了留出边框的边界, 边框对象所占用的区域至少要比该组中所有控制元对象占用的区域大。一般情况下, 每边要多出2个像素单位(pixels), 这完全由程序设计者自己决定。

下面的例子是创建一个边框对象, 用来组织前面创建的收音机按钮组:

```
>> ftdir = uicontrol(gcf, 'Style', 'frame', ...
    'Position', [195 15 110 110]);
```

在这个例子中, 边框对象的位置向量是这样计算的: 设每个控制元对象为100个像素宽, 最下面的控制元的下边界离图形窗口的底边界的距离为20个像素, 而最上面的控制元的上边界离图形窗口的底边界的距离为120个像素。为了在控制元组的每边多加5个像素的宽度, 所以边框对象的长和宽都是110个像素。这种设置使得边框的边界是可见的。从图形窗口的左边界开始195个像素的位置为边框对象左边界的起点, 向右移动5个像素, 才是各个控制元的最左边界(200像素处)。边框对象的底边界离图形窗口的底线的距离为15个像素。

5.2 菜单对象

菜单对象(或称下拉式菜单对象)可以让用户在运行应用程序时, 从一批功能选择项中浏览和选择某项功能。在 MATLAB 的每个图形窗口上, 有一个主菜单栏, 所有的主菜单都列在菜单栏上。菜单的标题或名字简单地描述了该菜单的功能。

在 Windows 系统中, 菜单栏在图形窗口的标题栏下面。File、Edit、Windows 和 Help 菜单是 MATLAB 图形窗口中的缺省主菜单。如果要取消图形窗口的缺省的主菜单, 可以将图形窗口的MenuBar 属性事先设置为 none, 然后再创建用户自己要定义的主菜单。


```
'Callback', 'ltype = ":";');

>> dashed = uimenu(plotlt, 'Label', 'Dashed line', ...
    'Callback', 'ltype = "—";');
```

3. 创建子菜单对象

在创建子菜单对象的 `uimenu` 函数语句中, 必须指明父菜单的句柄值, 并返回子菜单对象的句柄值, 供定义子菜单对象的菜单项之用。下面的例子定义菜单对象 Plot Options 的 3 个菜单项, 每一个菜单项都是子菜单, 用分隔条将每个子菜单分开:

```
>> plotlt = uimenu(plotopt, 'Label', 'Line Type');
>> plotsym = uimenu(plotopt, 'Label', 'Plot Symbol', 'Separator', 'on');
>> plotcol = uimenu(plotopt, 'Label', 'Color', 'Separator', 'on');
```

4. 创建带检测标记的菜单项对象

MATLAB 对菜单项对象规定了一个属性 `Checked`, 其属性值可以是 `on` 或 `off`, 由此定义菜单项对象的两种不同的状态。当菜单项的状态为 `on` 时, 该菜单项名字的左端具有一个检测的标记, 可利用这个标记指示出对该菜单项所作出的选择。

例如, 下面的语句使得当用户选择 Solid Line 菜单项时, 就在该菜单项的左端作一个标记。由于一次只能选择一个菜单项, 因此, 标记只出现在一个菜单项上:

```
>> solid = uimenu(plotlt, 'Label', 'Solid Line', ...
    'CallBack', ['ltype = "-"; ...

        set(solid, "Checked", "on"), ...
        set(dotted, "Checked", "off"), ...
        set(dashed, "Checked", "off")']);
```

5. 创建切换式菜单项对象

MATLAB 可以让创建的菜单项对象在两种状态之间进行切换, 而菜单项的不同状态是通过菜单项不同的标题来标识的。

下面的例子定义的菜单项具有 `on` 和 `off` 两种状态, 由菜单项的标题的改换来表明菜单项所处的状态, 两种状态的标题分别是 On Now 和 Off Now。当菜单项被选择时, 菜单的名字就会改变, 根据不同的状态来调用不同的子程序:

```
>> toggle = uimenu(topmenu, 'Label', 'On Now', ...
    'CallBack', ['if strcmp(get(toggle, "Label"), '...
        "'On Now'), ...
        set(toggle, "Label", "Off Now"), ...
        fnon, ...
        'else, ...
        set(toggle, "Label", "On Now"), ...
        'fnon, ...
        'end']);
```

5.2.2 菜单对象属性

1. 公共属性

(1) Children 属性

Children 属性的取值为空矩阵或是句柄值向量,因为菜单对象可以有自己的子菜单对象。对于菜单项来说,Children 的取值就是空矩阵,改变向量元素的顺序,可以改变菜单对象的子菜单及菜单项的顺序。

(2) Parent 属性

Parent 属性的取值是对象句柄值,因为菜单对象总是图形窗口对象的子对象或是另一个菜单对象的子菜单,该句柄值标明了菜单对象所在的图形窗口或其父菜单。如果用 set 函数将 Parent 属性值设置为另一个图形窗口的句柄值,则相当于将菜单对象移到另一个图形窗口中。

(3) Tag 属性

Tag 属性的取值是字符串,它定义了该菜单对象的一个标识值。定义了 Tag 属性后,在任何程序中都可以通过这个标识值找出该菜单对象。

(4) Type 属性

Type 属性的取值总是 uimenu,这个属性值标明图形对象的类型。对菜单对象,其类型就是 uimenu,用户不能改写这个属性。

(5) UserData 属性

UserData 属性的取值是一个矩阵,缺省值为空矩阵,用户可以在这个属性中保存与该菜单对象相关的重要数据或信息,借此可以达到传递数据或信息的目的。可以用 set 和 get 函数访问该属性。

(6) Visible 属性

Visible 属性的取值是 on(缺省值)或 off。如果该属性值为 off,那么菜单对象是不可见的,但是该菜单对象仍存在,可以用 set 和 get 函数访问这个菜单对象的每个属性。

2. 基本控制属性

(1) Accelerator 属性

Accelerator 属性的取值是字符,只对 Children 属性值为空的对象,即菜单项对象才有定义,它定义该菜单项的热键。在 Windows 环境下,用 Ctrl 键加该字符,即可激活该菜单项的 callback 功能。

(2) Callback 属性

Callback 属性的取值是字符串,可以是某个 M 文件名或一小段 MATLAB 语句。当用户激活菜单对象时,如果菜单对象没有子菜单就运行 Callback 属性定义的子程序(callback 调用)。如果是子菜单对象,那么先运行 Callback 属性定义的子程序,再显示子菜单对象的子项。

(3) Checked 属性

Checked 属性的取值是 on 或 off(缺省值),该属性为菜单项定义一个指示标记,可以用

这个特性指明某种选择状态。

(4) Enable 属性

Enable 属性的取值是 on(缺省值)或 off, 这个属性控制菜单对象的可选择性。如果它的值是 off, 则此时不能使用该菜单。此时, 该菜单标题的外观是阴影。

(5) Label 属性

Label 属性的取值是字符串, 它定义菜单对象的名字。可以在字符串中加入“&”字符, 那么在该菜单标题上, 跟随“&”字符后的字符有一条下划线, “&”字符本身不出现在标题中。对于这种有带下划线字符的菜单, 可以用 Alt 键加该字符键来激活。

(6) Position 属性

Position 属性的取值是数值, 它定义主菜单对象在菜单条上的相对位置, 或子菜单对象与菜单项对象在菜单组内的相对位置。例如, 对主菜单, Position 属性值为 1, 表示该菜单位于图形窗口菜单条的可用位置的最左端。

(7) Separator 属性

Separator 属性的取值是 on 或 off(缺省值)。如果该属性值为 on, 则在该菜单项上方添加一条分隔线, 可以用分隔线将各菜单接功能分开。

3. Callback 管理属性

(1) BusyAction 属性

BusyAction 属性的取值是 cancel 或 queue(缺省值), 该属性决定 MATLAB 采取的控制中断执行菜单对象的 callback 调用的方式。在 MATLAB 环境中, 如果有一个 Callback 调用在运行, 那么随后的 Callback 调用总是试图中断正在运行的 Callback 调用。如果此时 Interruptible 属性值为 on, 那么 MATLAB 在处理事件队列时, 中断正在运行的 Callback 调用; 如果 Interruptible 属性值为 off, 那么 BusyAction 属性就决定 MATLAB 处理事件的方式: 如果它的值是 cancel, 就从事件队列中取消企图运行第二个 Callback 调用的事件; 如果它的值是 queue, 就将企图运行第二个 Callback 调用的事件加入事件队列, 直到当前的 Callback 调用完成为止。

(2) CreateFcn 属性

CreateFcn 属性的取值是字符串, 一般是某个 M 文件名或一小段 MATLAB 语句。当 MATLAB 生成菜单对象时, 即 MATLAB 事先应该执行的程序段。这个属性只能按设置缺省值的方式定义(参见 3.10), 例如, 下列语句

```
>> set(0,'DefaultUicontrolCreateFcn',...
    'set(gcf,"IntegerHandle","off")')
```

从根对象层设置菜单对象的缺省值。对于已经存在的菜单对象, 改变该属性之值, 对该对象无任何影响。MATLAB 在生成菜单对象, 完成所有缺省属性值的设置后, 再执行 CreateFcn 定义的 Callback。

如果一个菜单对象 CreateFcn 定义的 Callback 正在运行, 那么该菜单对象的句柄值不可访问, 必须用函数 gcbo 获取。

(3) DeleteFcn 属性

DeleteFcn 属性的取值是字符串, 一般是某个 M 文件名或一小段 MATLAB 语句。当删

除菜单对象时, MATLAB 在清除该对象属性之前, 执行由 DeleteFcn 属性定义的 Callback。如果一个菜单对象 DeleteFcn 定义的 Callback 正在运行, 那么该菜单对象的句柄值不可访问, 必须用函数 gcbo 获取。

(4) HandleVisibility 属性

HandleVisibility 属性的取值为 on(缺省值)、callback 或 off, 这个属性定义菜单对象句柄的可访问权限, 决定其句柄值是否在父对象的 Children 属性中出现。如果 HandleVisibility 属性值是 on, 那么它的句柄值总是可见的; 如果属性值为 callback, 那么该句柄值只在该对象的 Callback 调用以及 Callback 调用的函数内是可见的, 但是命令行内调用的函数则不能访问这样的菜单句柄值, 也就是说, 这样的句柄对于该菜单对象自己的 Callback 调用是私有的, 这对于保护交互式的程序是十分有用的; 如果 HandleVisibility 属性值被设置为 off, 那么菜单对象的句柄值在任何时刻都是不可见的。如果句柄是不可见的, 那么任何查询句柄的函数都不能返回它的值, 这些函数包括 get、findobj、gcf、gca、geo、newplot、cla、clf、close 等。当 HandleVisibility 属性值是 callback 或 off 时, 这样的菜单对象不会出现在图形窗口对象的 Children 和 CurrentObject 属性中, 也不会出现在根对象 CallbackObject 属性中, 但是可以将根对象的 ShowHiddenHandles 属性设置为 on, 临时地使得所有句柄都是可见的, 而不管其 HandleVisibility 属性值是什么。被隐藏的句柄仍是有用的, 如果知道这样的句柄值, 则一切关于句柄的操作都是合法的。

(5) Interruptible 属性

Interruptible 属性的取值是 on 或 off(缺省值), 这个属性决定着菜单对象的 Callback 是否可以被随后的 Callback 调用中断。只有由 ButtonDownFcn 和 Callback 属性定义的 Callback 受 Interruptible 属性值的影响。在运行程序时, 只有遇到 drawnow、figure、getframe 和 pause 等命令时, MATLAB 系统才检查可以中断程序运行的 Callback 调用事件。

5.3 应用例子

本节将通过一个简单的应用程序设计实例进一步说明如何利用前面已介绍的交互设计技术来设计用户图形界面。下面设计一个应用程序, 它的图形界面由控制元对象和菜单对象两部分图形元构成, 其中, 菜单项可以模拟大部分控制元对象的操作功能。

要设计的应用程序的用户图形界面如图 5-5 所示, 在这个图形界面窗口上, 用户可以通过对各种控制元或菜单的操作, 达到下列目的:

- (a) 用户可以从某个初始时间, 或者是系统的时间, 或者是用户输入的特定的初始时间启动时钟窗口, 收音机按钮组 Initialize 中的收音机按钮向用户提供这种设置。用户可以使用系统时间作为初始时间, 也可以通过选择收音机按钮 Start at, 然后在它旁边的编辑框中(空白部分)输入用户想要设定的初始时间。等价地, 用户可以从 Options 主菜单中通过其子菜单 Set Start Time 来设置一种初始时间输入方式;
- (b) 让用户在两种时钟形态中选择一种。在收音机按钮组 Type 中, 用户可以任意选

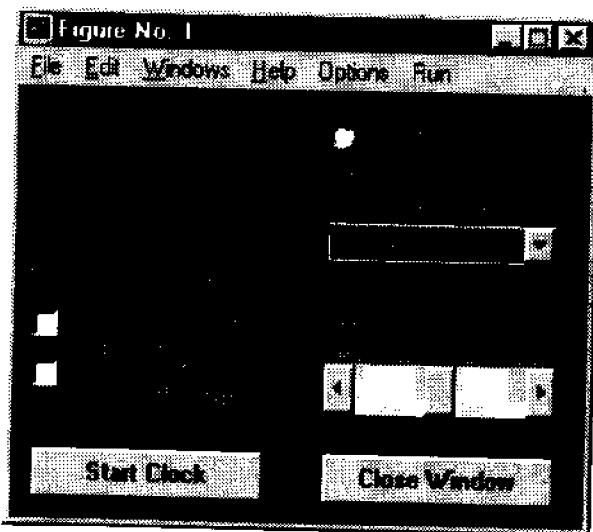


图5-5 GUI应用程序

择一个按钮的状态为 on, 得到时钟的某种显示类型。同时, 用户也能从 Options 主菜单的子菜单 Clock Types 中选择所要的时钟类型;

- (c) 可以显示格林威治时间, 以及几个大城市的当地时间。在图形界面上, 打开弹出式菜单 Show Local Time, 应用程序会列出一系列可供选取的城市名。选取一项后, 时钟的时间就是相应城市的当地时间了, 缺省的选择值是格林威治时间。另外, Run 主菜单的子菜单 Local Time 也提供这种选择功能;
- (d) 可以显示秒钟, 也可以用强弱可调的“滴答”声报时。为了显示秒钟, 用户可将 Show Seconds 检测框的状态设置为 on; 为了调整滴答声, 可以将 Audio Ticks 检测框的状态设置为 on, 然后, 可以用 Ticker Volume 滑标条来设置声音的强弱。当然, 用户也可以从 Options 主菜单的 Setting 子菜单中选择。

根据所设置的选项, 用户可以点按 Start Clock 按钮启动时钟, 或从 Run 主菜单的 Start Clock 菜单项启动时钟。若要结束时钟应用程序, 用户只需点按 Close Window 按钮或从 Run 主菜单中选 Close Clock 菜单项即可。

5.3.1 按钮的设计

在上述的应用程序中, 我们使用两个控制按钮:一个用于启动时钟, 另一个用于关闭时钟窗口。

1. 启动时钟

启动时钟按钮的标题是 Start Clock, 当用户激活该按钮, 即用鼠标点按该按钮时, 应用程序按两种方式之一来设置初始时间, 并启动时钟:一种方式是利用系统时间作为初始时间, 另一种方式是由用户设置初始时间。此外, 时钟的外形有两种选择:一种是数字模式, 一种是3维时钟模式。

启动时钟按钮的 callback 调用 M 文件 initime.m, 该文件的主要任务是设置初始时间, 即或者取系统时间, 或者取用户输入的时间。然后, 根据收音机按钮的选择状态, 调用不同的 M 文件 clkdig.m 或 clk3d.m, 分别启动数字式模拟时钟或 3 维模式模拟时钟。我们不打算写出这些 M 文件, 因为要调用 Windows 的图形资源, 程序较复杂, 这不是本例子的目的。

下面的语句是用于创造标题为 Start Clock 的按钮:

```
% Start clock pushbutton
% rb(1) is the Digital Clock radio button
% rb(2) is the 3D Clock radio button
pbstart = uicontrol(gcf, 'Style', 'push', ...
    'String', 'Start Clock', ...
    'Position', [10 10 120 25], ...
    'CallBack', ['initime,', ...
        'if get(rb(1), "Value") == 1,',...
        'clkdig,', ...
        'elseif get(rb(2), "Value") == 1,',...
        'clk3d,', ...
        'end']);
```

2. 关闭时钟窗口

关闭时钟窗口按钮的标题是 Close Window, 它的作用是关闭当前的图形窗口, 用下面的语句创建:

```
% Close Window pushbutton
pbclose = uicontrol(gcf, 'Style', 'push', ...
    'String', 'Close Window', ...
    'Position', [160 10 120 25], ...
    'CallBack', 'close(gcf);');
```

3. 检测框的设计

在我们要设计的应用程序中, 包含了一个检测框组。检测框组的作用是设置时钟是否报出“滴答”声, 以及是否显示秒钟的各种状态。Options 边框对象将各个检测框组织在一起, 下面的语句可生成有关的边框对象、文字说明对象, 以及检测框对象等:

```
% Frame that contains Options group
frameopt = uicontrol(gcf, 'Style', 'frame', ...
    'Position', [5 55 130 80]);
% Text control that labels Options group
txtopts = uicontrol(gcf, 'Style', 'text', ...
    'String', 'Options', 'Position', [10 110 120 20]);
% Check box that sets audio ticks
cktick = uicontrol(gcf, 'Style', 'checkbox', ...
    'String', 'Audio Ticks', ...)
```

```

'Position', [10 85 120 25]);
% Check box that sets display of seconds
cksecs = uicontrol(gcf, 'Style', 'checkbox', ...
    'String', 'Show Seconds', ...
    'Position', [10 60 120 25]);

```

虽然这些检测框对象都没有设置 Callback, 但是它们的状态(由其 Value 属性值定义)将被启动时钟的有关 M 文件检测和使用。例如, 下面的语句决定 Audio Ticks 是否是 on 状态:

```
>> get(cktick, 'Value')
```

如果返回值是1 (当 Max 属性被定义, 此时 Value 属性值应为 Max 的属性值), 则表示该检测框的状态为 on; 如果返回值为0 (当 Min 属性被定义, 此时 Value 属性值应为 Min 的属性值), 则表示检测框的状态为 off。

5.3.2 收音机按钮的设计

应用程序的收音机按钮组用来设置时钟窗口的初始时间, 以及初始化时钟。收音机按钮组被组织在 Initialize 边框对象中, 收音机按钮组的状态决定以何种方式设定初始时间。选择 Start at 收音机按钮, 则表示由用户设定初始时间, 初始时间从该收音机按钮右边的编辑框中输入; 如果选择 System time 收音机按钮, 则表示使用系统时间, 这也是该应用程序缺省的时间设置方式。

```

% Radio button that control setting the initial time
rbstartat = uicontrol(gcf, 'Style', 'radio', ...
    'String', 'Start at', ...
    'Position', [160 180 80 25], ...
    'CallBack', ['if get(rbstartat, "Value") == 1, ', ...
        'set(rbsystime, "Value", 0), ', ...
        'else set(rbsystime, "Value", 1), ', ...
        'end']);
rbsystime = uicontrol(gcf, 'Style', 'radio', ...
    'String', 'System time', ...
    'Position', [160 155 120 25], ...
    'Value', 1, ...
    'CallBack', ['if get(rbsystime, "Value") == 1, ', ...
        'set(rbstartat, "Value", 0), ', ...
        'else set(rbstartat, "Value", 1), ', ...
        'end']);

```

在这个例子中, 可以通过点按收音机按钮来作选择。callback 保证只有一个收音机按钮的状态是 on。

5.3.3 滑标条的设计

下面的应用程序使用一个滑标条,用它来控制时钟“滴哒”声的强弱,该滑标条对象的范围值是缺省值[0, 1]。

```
% Text that labels the slider
txtsli = uicontrol(gcf, 'Style', 'text', ...
    'String', 'Ticker Volume', ...
    'Position', [160 100 120 20]);
% Text that labels the slider range
txtvol = uicontrol(gcf, 'Style', 'text', ...
    'String', 'Soft ..... Load', ...
    'Position', [160 80 120 20]);
% Slider that controls the ticker volume
% The initial value is the midpoint of the range
slivol = uicontrol(gcf, 'Style', 'slider', ...
    'Position', [160 55 120 25], ...
    'Value', 0.5);
```

在上述程序中,前两条语句创建滑标条的说明文字,在第三条语句中,滑标条自身没有定义 callback 属性,设置时钟的函数会取得滑标条的 Value 属性值,并根据滑标条的当前值来设置“滴哒”声的强弱。

5.3.4 弹出式菜单的设计

弹出式菜单可以让用户在多个时间标准中选择一种时间标准,可以选择的时间标准有格林威治时间(GMT)和三个城市的当地时间,每种选择都关联着一个数值,代替与格林威治时间的时钟偏差。在应用程序中,要利用 UserData 来存放和传递这些数值。以下语句用于创建弹出式菜单对象及其相关的说明文字:

```
% Text that labels the pop-up menu
txtpop = uicontrol(gcf, 'Style', 'text', ...
    'Position', [10 180 120 20], ...
    'String', 'Show Local Time');
% Pop-up menu that sets the local time
popcity = uicontrol(gcf, 'Style', 'popupmenu', ...
    'Position', [10 155 120 25], ...
    'String', 'GMT |Boston |Paris |Beijing', ...
    'UserData', [0; -4; 1; 8], ...
    'CallBack', ['ud = get(popcity, "UserData");', ...]
```

```
'setclk(ud(get(popcity, "Value")))];
```

弹出式菜单的 Callback 属性定义了一个 callback 调用, 其操作如下: 首先取得该弹出式菜单对象的 UserData 属性值, 并赋给变量 ud; 然后, 根据用户选择的菜单项的序号, 将该矩阵的相应时间偏差传递给时钟设置函数 setclk。

5.3.5 编辑框的设计

在时钟应用程序中, 用户可以利用与收音机按钮组一起的编辑框输入用户要设定的初始时间:

```
% Editable text for manual entry of the time
entime = uicontrol(gcf, 'Style', 'edit', ...
    'Position', [240 180 40 25]);
```

通常, 编辑框的 callback 包含检查用户输入数据的合法性的步骤。例如, 当用户输入的数据小于 0 时, 下面的语句回应一条错误信息, 提示用户输入值必须大于 0。

注意: 由于编辑框中输入的是数字字符串, 因此, 还必须将其转化成数值:

```
'CallBack', ['if str2num(get(entime, "String"))<=0;', ...
    'disp("Error—Value must be positive"),', ...
    'end']]
```

5.3.6 菜单的设计

前面设计的各项控制元对象, 实现了应用程序所要求的各种操作功能。同样地, 可以设计出一套菜单对象, 用菜单操作的方式实现以上所要求的各项操作功能。在这个应用程序中, 设计了两个主菜单 Options 和 Run。菜单 Options 由子菜单 Clock Type、Settings 和菜单项 Set Start Time 构成; 而子菜单 Clock Type 的菜单项分别是 Digital 和 Three-D, 子菜单 Settings 的菜单项分别是 Audio Ticker 和 Show Seconds; 主菜单 Run 由菜单项 Start Clock、Close Clock 和子菜单 Local Time 组成; 而子菜单 Local Time 的菜单项分别是 GMT、Boston、Paris 和 Beijing。

下面的语句生成 Options 菜单和它的各子项, Set Start Time 菜单项的CallBack 的功能是启动函数 setstart 初始化时钟, 这里没有提供该函数的有关代码:

```
% Options menu and its submenu
Opts = uimenu(gcf, 'Label', 'Options');
% Clock Types submenu and its menu items
CTypes = uimenu(Opts, 'Label', 'Clock Types');
uimenu(CTypes, 'Label', 'Digital', 'CallBack', 'ctype = 1;');
uimenu(CTypes, 'Label', 'Three-D', 'CallBack', 'ctype = 2;');
% Settings submenu (items not defined in this example)
settings = uimenu(Opts, 'Label', 'Settings', 'Separator', 'on');
```

```
% Set Start Time menu item (no handle needed)
uimenu(Opts, 'Label', 'Set Start Time', 'Separator', 'on',...
    'CallBack', 'setstart');
```

当用户选择某种时钟类型时,CallBack 将赋一个值给 ctype 变量。若用户启动时钟(通过选择 Run 菜单中的菜单项),这个值就被用来决定时钟的显示类型。

Run 菜单有三个菜单项,其中一个是子菜单对象。下面的语句用来生成 Run 菜单:

```
% Run menu and its submenus
Run = uimenu(gcf, 'Label', 'Run');
% Start Clock submenu (no handle needed)
uimenu(Run, 'Label', 'Start Clock', ...
    'CallBack', ['inittime', ...
        'if ctype==1,' ...
            'clkdig,' ...
        'elseif ctype==2,' ...
            'clk3d,' ...
        'end']);
% Local Time submenu and its menu items
LocTime = uimenu(Run, 'Label', 'Local Time', 'Separator', 'on');
uimenu(LocTime, 'Label', 'GMT', 'CallBack', 'adjtime(0)');
uimenu(LocTime, 'Label', 'Boston', 'CallBack', 'adjtime(-4)');
uimenu(LocTime, 'Label', 'Paris', 'CallBack', 'adjtime(1)');
uimenu(LocTime, 'Label', 'Beijing', 'CallBack', 'adjtime(8)');
% Close Clock menu item
uimenu(Run, 'Label', 'Close Clock', 'Separator', 'on',...
    'CallBack', 'close(gcf)');
```

Local Time 子菜单让用户选择格林威治标准时间(GMT),或波士顿(Boston)、巴黎(Paris)、北京(Beijing)等当地时间,Local Time 的菜单项中的 callback 就是将时间调整到正确的时间,Close Clock 菜单项用来关闭时钟窗口。

5.4 MATLAB GUI 高级特性

这一节主要介绍 MATLAB 用户界面设计的高级技巧,包括以下内容:

- 创建择一选择的收音机按钮组
- 在函数内部进行用户界面编程
- 鼠标的操作
- CallBack 中断技术

5.4.1 择一选择的收音机按钮组的设计

通常,收音机按钮组的功能是使用户从一组可选状态中选出单一的状态(on),而其他的状态被自动设置为 off。然而,MATLAB 收音机按钮组自身并不具有这种功能,尽管每个收音机按钮可以是 on 与 off 两种状态,但不同的收音机按钮的状态之间无任何必然的联系。要想让一组收音机按钮的状态具有择一性,必须使用专门的程序技巧来实现。在上一节的例子中,提供了一种较为简单的方法。这里将介绍一种较为系统的设计方法。

如果应用程序要求通过多个收音机按钮提供不同的选择状态,首先应该将有关的收音机按钮组成一个逻辑组,然后设计一段程序,保证该组中的收音机按钮只有一个按钮的状态是 on,而其他的按钮的状态是 off。

例如,假设应用程序用一组收音机按钮来控制在线段图形的三种顶点标记中择一地选择一种顶点标记,第一个收音机按钮对应缺省的顶点标记,它的 Value 属性值可以设置为 1。于是,有

```
sym(1) = uicontrol(fig, 'Style', 'radio', ...
    'Position', [100 300 75 25], ...
    'String', 'Circle', 'Value', 1);
sym(2) = uicontrol(fig, 'Style', 'radio', ...
    'Position', [100 275 75 25], ...
    'String', 'Plus', 'Value', 0);
sym(3) = uicontrol(fig, 'Style', 'radio', ...
    'Position', [100 250 75 25], ...
    'String', 'Star', 'Value', 0);
```

为了使得这组收音机按钮中只有一个按钮的状态是 on,可采用下面的技巧,即将每个收音机按钮的 UserData 属性值定义为组中其他收音机按钮的句柄值构成的矩阵,然后,当用户选择一个收音机按钮时,它的 CallBack 就检查组中其他的收音机按钮,将原来状态为 on 的收音机按钮的状态改变为 off。下面的语句用来设置句柄值:

```
for i = 1:3
    set(sym(i), 'UserData', sym(:, [1:(i-1), (i+1):3]))
end
```

而以下的语句定义一个 CallBack:

```
call = ['me = get(gcf, "CurrentObject");',...
    'if (get(me, "Value") == 1),',...
    'set(get(me, "UserData"), "Value", 0),',...
    'else,',...
    'set(me, "Value", 1),...
    'set(get(me, "UserData"), "Value", 0),',...
    'end'];
```

```
set(sym, 'CallBack', call);
```

如果 Min 和 Max 属性值被修改过,那么定义 call 变量的语句中的数值 0 和 1 最好换成 Min 和 Max 的属性值。

5.4.2 GUI 设计方法

到目前为止,大部分例子中的 CallBack 属性值是由多条语句组成的字符串。当 CallBack 要完成的工作很复杂时,这种方法是不适应的。一种有效的方法是根据某些规则,单独编写图形界面的应用程序,将用户界面的设计局限在一个函数内,在这个函数中按不同的选择来创建用户界面对象和定义有关的 CallBack。这种方法至少有以下几个好处:

- (a) 由于 MATLAB 的 clear 命令会清除工作空间中的有关变量,使用函数技巧可以保护图形界面中的有关变量不会被清除;
- (b) 由于 CallBack 与对象创建函数分开,更易于编写和修改,故对 callback 函数进行 debug 时并不要求同时生成使用它的有关图形对象元;
- (c) 执行速度更快,因为 MATLAB 只需要编译函数,而无需将 callback 的值传送给 eval 函数解释。

在编写用户界面程序时,建议使用下面的规则:

- (a) 用下列的语句定义一个函数,它的输入参数决定选用的函数和操作:

```
function uifun(action)
```
- (b) 利用条件语句将生成界面控制元对象的操作和 callback 的操作分开。例如,在创建用户界面对象的代码之前使用下面的语句:

```
if strcmp(action, 'initialize')
```
- (c) 又如,对使用 ButtonDown 属性的 CallBack,可以用下列语句引导出有关的代码:

```
elseif strcmp(action, 'linedown')
```
- (d) 通过发布类似如下的命令来启动用户界面:

```
uifun('initialize')
```
- (e) 定义特定的 CallBack 属性值,用适当的功能说明字符启动 uifun 函数。例如,下面的语句定义 Lineobj 的 ButtonDownFcn 操作:

```
set(lineobj, 'ButtonDownFcn', 'uifun("linedown")')
```
- (f) 保护好用户界面的各层对象的句柄值和应用的有关数据,防止应用程序清除某些变量。为了使得每个函数启动时,这些句柄值和数据可供调用,或者将这些变量宣称为全程变量,或者将它们保存在 UserData 属性中。

1. 使用全程变量

使用全程变量是一种最为直接和有效的保护句柄值和应用数据的方法,但是它仍有以下两个缺点:

- (a) 用户可能会键入命令 clear global,意外地清除某些图形界面中有用的全程变量;
- (b) 如果应用程序打开多个图形窗口,则在没有其他的判别方法时,MATLAB 难以区分各个图形窗口的用户界面对象元。

下面来看一个使用全程变量的例子,这是上述例子的进一步扩展:

```
% uifun function to many Close user interface
function uifun(action)
global UISTART UIRB
% specifying no argument == specifying 'initialize'
if nargin < 1,
    action = 'initialize';
end;
% Create controls
if strcmp(action, 'initialize'),
    UISTART = uicontrol(gcf, 'Style', 'push', ...
        'String', 'Start', ...
        'Position', [10 10 120 25], ...
        'CallBack', 'uifun("startcb")');
    UIRB(1) = uicontrol(gcf, 'Style', 'radio', ...
        'String', 'Digital Clock', ...
        'Position', [10 175 120 25], ...
        'Value', 1);
    UIRB(2) = uicontrol(gcf, 'Style', 'radio', ...
        'String', '3-D Clock', ...
        'Position', [10 150 120 25], ...
        'Value', 0);
% Define Start push button callback
elseif strcmp(action, 'startcb'),
    inittime
    if get(UIRB(1), 'Value') == 1, clkdig,
    elseif get(UIRB(2), 'Value') == 1, clk3d,
    end
% Define other callback
end
```

2. 使用 UserData 矩阵

将有关图形界面的句柄值存在 UserData 矩阵中是一种较为复杂的方法,且要求一些额外的执行时间。但是与全程变量的方法相比,UserData 矩阵的方法至少有两条优点:第一,只要图形窗口是打开的,则所有的句柄值和有关的变量值都存在;第二,当用户打开另一个图形窗口时,它的控制元对象的句柄值和有关的变量的值不会与前一个图形窗口的有关值相混淆。

用 UserData 方法,上一小节的程序可以改写为

```
% uifun function to many Close user interface
function uifun(action)
```

```
% specifying no argument == specifying 'initialize'
if nargin < 1,
    action = 'initialize';
end;
% Create controls
if strcmp(action, 'initialize'),
    start = uicontrol(gcf, 'Style', 'push', ...
        'String', 'Start', ...
        'Position', [10 10 120 25], ...
        'CallBack', 'uifun("startcb")');
    rb(1)= uicontrol(gcf, 'Style', 'radio', ...
        'String', 'Digital Clock', ...
        'Position', [10 175 120 25], ...
        'Value', 1);
    rb(2)= uicontrol(gcf, 'Style', 'radio', ...
        'String', '3-D Clock', ...
        'Position', [10 150 120 25], ...
        'Value', 0);
    set(gcf, 'UserData', [start, rb]);
% Define Start push button callback
elseif strcmp(action, 'startcb'),
    uihandles = get(gcf, 'UserData');
    rb(1) = uihandles(2);
    rb(2) = uihandles(3);
    inittime
    if get(rb(1), 'Value') == 1, clkdig,
    elseif get(rb(2), 'Value') == 1, clk3d,
    end
% Define other callback
end
```

5.4.3 鼠标操作处理技术

根据 Windows 系统的体系结构,下面的这些鼠标动作被称为事件:

- (a) 当鼠标箭头落在图形窗口中,按下鼠标键时;
- (b) 释放鼠标键时;
- (c) 在图形窗口中移动鼠标键时。

MATLAB 系统保持了对这种概念的支持。对图形对象, MATLAB 定义了许多 Callback 属性,以便程序对这些事件作出反应。

本节将介绍如何实现对上述情况的处理,也可以参看 MATLAB 的演示程序 sigdemol.m 和 sigdemo2.m(在 demos 目录下)。

1. 鼠标箭头的位置

鼠标箭头的位置和用户的操作方式决定了 MATLAB 如何执行它的某些特定的 callback。例如,当用户点按鼠标键时,鼠标箭头的位置将确定出图形窗口中的一个当前的图形对象,用户对鼠标键的操作方式和应用程序将决定出某个 callback,并启动执行。为了准确地对鼠标所在位置进行描述,MATLAB 将其位置状态分为四种情况,并根据此时用户的动作来决定应该执行的 Callback。这4种情况是:

- (a) 当箭头落在一个控制元对象上或菜单对象上时;
- (b) 当箭头邻近一个控制元对象时;
- (c) 当箭头邻近或在某些图形对象上时;
- (d) 当箭头在图形窗口中,但既不邻近也不在控制元对象或某些图形对象上时,这种情况称为鼠标箭头落在图形窗口内。

这里,我们对“邻近”的概念作如下的解释:图形对象在图形窗口中占据一定的位域,如果鼠标箭头是落在某个图形对象区域的5个像素宽的外边界(不是对象区域本身)内,就称为鼠标是“邻近”该图形对象。

MATLAB 提供了专门的属性来定义图形窗口对象、控制元对象、菜单对象和某些图形对象的 callback。MATLAB 根据鼠标箭头的不同位置,来决定启动不同层次的 callback 程序。表5-2列出了鼠标箭头位置、鼠标的操作方式和 callback 启动属性之间的关系。

表5-2 Callback 调用方式

鼠标箭头位置	用户处理方式	Callback 启动属性
落在控制元对象或菜单对象上	按下鼠标键	启动控制元或菜单的 Callback
邻近控制元对象	按下鼠标键	启动图形窗口的 WindowButtonDownFcn 定义的程序,再启动控制元对象的 ButtonDownFcn 定义的程序
邻近控制元对象	释放鼠标键	启动图形窗口的 WindowButtonUpFcn 定义的程序
邻近控制元对象	移动鼠标箭头	启动图形窗口的 WindowButtonMotionFcn 定义的程序
邻近或在图形对象(除控制元对象和图形窗口对象外)上	按下鼠标键	启动图形窗口的 WindowButtonDownFcn 定义的程序
邻近或在图形对象(除控制元对象和图形窗口对象外)上	释放鼠标键	启动图形窗口的 WindowButtonUpFcn 定义的程序
邻近或在图形对象(除控制元对象和图形窗口对象外)上	移动鼠标箭头	启动图形窗口的 WindowButtonMotionFcn 定义的程序
图形窗口内	按下鼠标键	启动图形窗口的 WindowButtonDownFcn 定义的程序
图形窗口内	释放鼠标键	启动图形窗口的 WindowButtonUpFcn 定义的程序
图形窗口内	移动鼠标箭头	启动图形窗口的 WindowButtonMotionFcn 定义的程序

2. 按下鼠标键的处理

当用户在图形窗口中按下鼠标键时, MATLAB 按顺序作下列处理:

- (a) MATLAB 检测鼠标箭头的位置, 决定是否选择某个图形对象元。如果一个图形对象被选择, 则 MATLAB 将图形窗口的属性 CurrentObject 设置为该图形对象的句柄值; 如果无图形对象被选择, 属性 CurrentObject 就被置为图形窗口本身;
- (b) MATLAB 设置图形窗口的 CurrentPoint 属性和 SelectionType 属性, 以及根对象的 CurrentFigure 属性。这些属性的具体含义在后面讨论;
- (c) MATLAB 将当前的图形对象置于选择栈的栈首, 栈内顺序在“选取对象”一段中讨论;
- (d) 如果用户选择了控制元对象, 则 MATLAB 仅仅是启动该控制元对象的 callback, 不再执行以下的处理工作;
- (e) MATLAB 执行图形窗口的 WindowButtonDownFcn 定义的程序;
- (f) MATLAB 执行所选对象的 ButtonDownFcn 定义的程序。

3. 释放鼠标键的处理

当用户释放鼠标键, 且鼠标箭头所在的图形窗口的 WindowButtonUpFcn 的程序已定义时, MATLAB 按下列顺序进行处理: 首先, MATLAB 更新图形窗口的 CurrentPoint 属性值; 然后, MATLAB 执行图形窗口 WindowButtonUpFcn 定义的程序。如果该属性未被定义, 即使是鼠箭头移到另一个图形窗口中, MATLAB 也不作任何处理, 即要执行的是按下鼠标键时, 鼠标所在的图形窗口的 WindowButtonUpFcn 定义的程序。

4. 移动鼠标箭头的处理

当用户在图形窗口内移动鼠标箭头, 且图形窗口 WindowButtonMotionFcn 已定义时, 根据已打开的图形窗口的数目和用户移动鼠标箭头的方式, MATLAB 分别执行不同的处理过程。当应用程序使用一个图形窗口, 或打开了多个图形窗口, 但仅仅在一个图形窗口内移动鼠标箭头时, 执行下列处理工作:

- (a) MATLAB 更新图形窗口的 CurrentPoint 属性值;
- (b) MATLAB 执行图形窗口的 WindowButtonMotionFcn 定义的程序。如果使用多个图形窗口, 且用户将鼠标箭头从一个图形窗口移到另一个图形窗口, 则执行下列的处理工作:
 - (a) 当用户移动鼠标箭头, 但不按下鼠标键时, 称为纯移动。对包含移动的箭头的图形窗口, MATLAB 将更新它的 CurrentPoint 属性值, 并执行该窗口的 WindowButtonMotionFcn 定义的程序;
 - (b) 当用户按下鼠标键时, 称为拖动。对于拖动时鼠标箭头所在的图形窗口, MATLAB 会更新它的 CurrentPoint 属性值, 并执行该窗口 WindowButtonMotionFcn 定义的程序; 当箭头移动到另一个图形窗口中时, 该图形窗口的属性值不受任何影响。用户按下鼠标键时所在的图形窗口称为是这个过程的主控图形窗口。

注意: 如果包含鼠标箭头的窗口未定义 WindowButtonMotionFcn, MATLAB 就不会更新它的 CurrentPoint 属性。

5. 相关属性总结

本节再介绍几个受鼠标操作影响的属性。

(1) 根对象的属性

根对象的 CurrentFigure 属性值为用户按下鼠标键时, 鼠标箭头所在的图形窗口的句柄值, 这个属性不受释放鼠标键和鼠标移动操作的影响, 而 PointerWindow 属性值为包含鼠标箭头的图形窗口的句柄值。

(2) 图形窗口的属性

- (a) WindowButtonDownFcn 属性定义当用户在图形窗口内(不在控制元对象上)按下鼠标键时 MATLAB 要执行的 callback;
- (b) WindowButtonUpFcn 属性定义当用户释放鼠标键时, MATLAB 要执行的 callback。该图形窗口应该是用户按下鼠标键时, 鼠标箭头所在的图形窗口;
- (c) WindowButtonMotionFcn 属性定义当用户在图形窗口内移动鼠标箭头时, MATLAB 要执行的 callback;
- (d) CurrentObject 属性指出鼠标箭头选择的图形对象, 如果无图形对象被选择, 它的值是当用户按下鼠标键时鼠标箭头所在的图形窗口的句柄值;
- (e) CurrentMenu 属性指出当前所选择的菜单或子菜单;
- (f) CurrentPoint 属性值是由鼠标箭头的位置定义的 x,y 坐标值, 其单位由图形窗口的 Units 属性决定。在以下3种情况下, MATLAB 将更新 CurrentPoint 的属性值: 第一, 当用户按下鼠标键时; 第二, 当用户释放鼠标键, 且 WindowButtonUpFcn 有定义时; 第三, 当用户移动鼠标箭头, 且 WindowButtonMotionFcn 有定义时;
- (g) SelectionType 属性指明按下鼠标键的方式, 即是否有附加的组合键 Shift 和 Ctrl 键, 以及是否单按或双按鼠标键。

(3) 坐标系的属性

坐标系的 CurrentPoint 属性包含图形窗口 CurrentPoint 属性定义的两个点的坐标值。坐标系 CurrentPoint 属性是将图形窗口的 CurrentPoint 属性值变换到坐标系的坐标值时导出的, 详见 5.4.3 节。

(4) 图形对象的属性

- (a) CallBack 属性定义用户在一定的条件下, 应用程序应该执行的预定义的操作或功能;
- (b) ButtonDownFcn 属性定义用户在图形对象附近(称为热区)按下鼠标键时, 应用程序应该执行的操作或功能。

6. 对象选择规则

如果某个对象(如线段对象)与其他的对象重叠, 因而难以选择该对象, 则 MATLAB 使用两条规则决定出用户按下鼠标键时所选择的对象: 第一, 由对象的栈序决定; 第二, 由对象的热区决定。在多个对象重叠的情况下, 仍按上述的规则决定每次选择的图形对象。

(1) 对象栈序

当鼠标箭头落在多个对象的重叠区中, 用户按下鼠标键时, 对象的栈序被用来确定应该选择的对象。开始时, 栈序反映了对象创建的顺序, 最后生成的对象在栈首。当鼠标箭头落在某个对象上, 并按下鼠标键时, 就将该对象移到了栈序的首位。在3维空间中, 并不是离观察

者近的对象在栈序的高位。为了确定出图形窗口中图形对象的栈序，可以查看图形窗口的 Children 属性值。子对象的句柄值向量的顺序即是当前的图形对象的栈序，栈首对象对应着该向量的第一个元素。

(2) 热区

一个对象的热区由该对象本身以及该对象图形的一定宽度的边界区域组成，不同类型的对象，其热区的形状不同。热区是计算机屏幕上的一个矩形区域，用户可以将鼠标箭头置于对象的热区内，按下鼠标键就选中热区中的对象。具体定义如下：

- (a) 对线段对象，热区是由线段本身以及围绕线段周围5个像素宽的周边区域构成的。所以线段对象的热区是线段本身加上10个像素宽度的区域；
- (b) 坐标系对象的热区是坐标系框加上坐标系外的坐标轴标题区(label 区)和坐标系标题区(title 区)，但不包括坐标系内的子对象或控制元对象等；
- (c) 曲面、平面片区域和文字说明对象的热区是包括这些对象的最小矩形区域。

在3维的坐标系中，这个最小矩形区域是屏幕上的2维矩形区域，它恰好包含了曲面对象、平面片区域对象或文字说明对象。例如，在图5-6中，矩形框围成的区域就是其中的曲面对象的热区。

- (d) 控制元对象的热区定义为控制元对象本身占据的矩形区域再加上四周5个像素宽的区域。

(3) 热区操作的响应

当用户在图形对象的热区中按下鼠标键时，MATLAB 将针对不同的热区类型作出不同的响应。下面对此略作分析：

对于前面定义的(a)~(c)类型的热区，当用户在热区按下鼠标键时，就表明用户选中了相应的图形对象，MATLAB 会将该对象移动至对象栈序的栈首，也就是把这个图形对象移到了屏幕的最前端，如果它与其他的图形对象是重叠的，那么此时，这个图形对象将挡住其他的图形对象。另一方面，如果被选中的图形对象的 ButtonDownFcn 属性被定义了一个 callback 子程序，MATLAB 还会启动该程序的执行。例如，下面的语句

```
>> line ([0,1],[1,0], 'ButtonDownFcn', 'disp("line")');
```

创建了一个线段对象，并定义了一个 callback，callback 的操作是在命令窗口中显示字符串“line”。然后，用户将鼠标箭头移到图形窗口中该线段附近（即线段的热区中）按下鼠标键时，MATLAB 命令窗口中就会显示一行字符串“line”，每按一次，字符串就被显示一次。

但是，在有些情况下，并不能很顺利地选中某个希望被选中的图形对象。在这样的情形中，改变坐标系的视点可以使选中图形对象的可能性增大。下面来分析一个较复杂的例子。

以下两条语句：

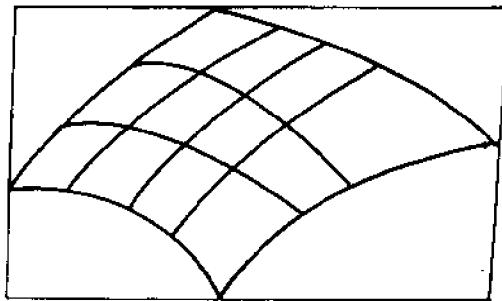


图5-6 曲面对象的热区

```
>> patch('ButtonDownFcn','disp("patch")');
>> line([1,0],[0,1],[0,2],'ButtonDownFcn','disp("line")');
```

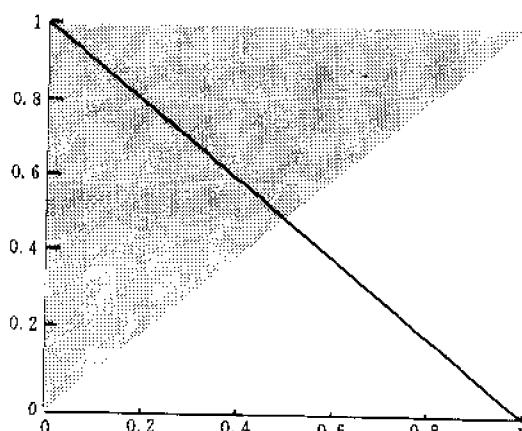


图5-7 平面片对象和线段对象

创建两个图形对象。第一条语句以缺省的方式定义一个平面片对象。执行第一条语句后, MATLAB 在打开的图形窗口中创建一个2维的坐标系,并在其中创建一个三角形区域的平面片对象。在执行第二条语句时,就在当前的坐标系中绘出一条直线段。由于使用的是低层线段对象创建函数 line,故它不改变当前的坐标系。尽管线段对象是3维空间中的线段,但仍在2维坐标系中输出线段对象的俯视图,因为 line 函数不改变坐标系的视点。最后输出的图形如图5-7所示。

平面片对象的热区包含它的最小矩形区域,即图5-7中整个矩形部分。很明显,线段对象的热区包含在平面片对象的热区中。由于线段对象是后生成的,因而当前线段对象处在栈首的位置。在线段对象的附近(即热区)点按鼠标键时,MATLAB 命令窗口中显示字符串“line”,表明了线段对象被选中,运行了 Button Down Fcn 属性定义的 callback。但是,一旦偏离了线段对象(因为它的热区范围很小),并点按了鼠标键,那么平面片对象则被选中,平面片对象将被移到对象栈首。此后,就不能再在平面片对象的热区中选线段对象了。

为了能够再次选中上述的线段对象,需要改变坐标系的视点。由于平面片对象是落在 x-y 平面上的,而线段对象是连接点(1,0,0)和点(0,1,2)的空间线段,所以可以改变坐标系的视点(目前是[0,90])将坐标系变成3维的坐标系,使得在新的坐标系中,线段对象的热区不完全包含在平面对象的热区中。

对于(d)类型热区,即控制元对象的热区,MATLAB 对于点按鼠标键的响应与前述有所不同。其不同之处主要在于对 callback 的处理。由于控制元对象有两个属性,即 Callback 属性和 Button Down Fcn 属性,可定义点按鼠标键的 callback 调用,所以,MATLAB 规定:在不同的位置上按下鼠标键,执行不同的 callback。具体规定如下:

如果用户是在控制元对象上点按鼠标键,就执行 Callback 属性所定义的 callback 调用;如果用户是在控制元对象的热区,但又不是在控制元对象上,即在控制元对象周边5个像素宽的区域内点按鼠标键,MATLAB 就执行由 ButtonDownFcn 属性所定义的 callback 调用。在这种情况下还必须注意的是:虽然是在控制元对象的热区中点按了鼠标键,但是,由于不是在控制元对象上点按鼠标键,因此,还相当于是在图形窗口中点按鼠标键。当图形窗口的 Window Button Down Fcn 属性有定义时,必须首先执行这个属性定义的 callback,再执行控制元对象的 Button Down Fun 属性定义的 callback。

7. 确定当前点

在用户进行下列3种操作,即第一,不论图形窗口的 WindowButtonFcn 定义与否,

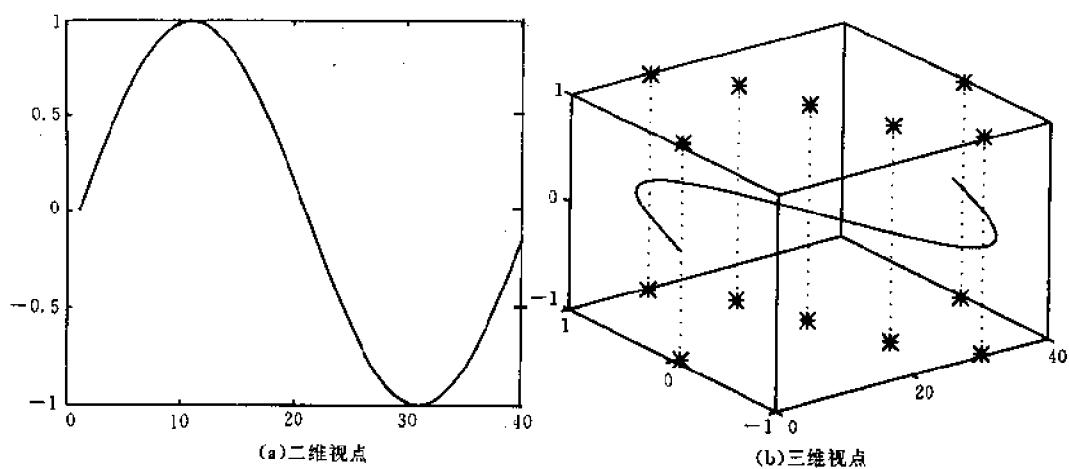


图 5-8 正弦曲线及当前点坐标

再换成另一个视点,就可以看到两个端点决定的线段既不平行于 x 轴,也不平行于 y 轴和 z 轴。换句话说,CurrentPoint 记录的是垂直于屏幕平面的直线被 3 维坐标系长方体截下的线段端点的 3 维坐标值。

8. 应用例子

下面的例子说明如何实现在图形窗口中移动一个按钮对象的功能。

```

>> fig = figure('Color', 'w');
>> posvec = [10 10 75 25];
>> pb = uicontrol(fig, 'Style', 'push', 'Position', posvec);
>> wbdeb = ['set(fig, "WindowButtonMotionFcn", [ ...
    ' "pv = get(fig, "CurrentPoint");", ...
    ' "posvec(1) = pv(1);", ...
    ' "posvec(2) = pv(2);"])];
>> curpos = 'set(pb, "Position", posvec)';
>> set(fig, 'WindowButtonDownFcn', wbdeb);
>> set(fig, 'WindowButtonUpFcn', [ ...
    ' set(fig, "WindowButtonMotionFcn", ""), curpos])

```

下面对上述的语句组作解释。第一条语句打开一个白色背景的图形窗口;第三条语句创建一个控制按钮对象,它在图形窗口中的位置由向量 posvec 定义;第四、五条语句各定义了一组语言构成的字符串向量;第六条语句将 wbdeb 的内容作为图形窗口 WindowButtonDownFcn 属性值。这样,如果用户在图形窗口按下鼠标键,MATLAB 就执行 wbdeb 中的 MATLAB 语句。在 wbdeb 中定义的语句是设置图形窗口对象的 Window Button Motion Fcn 属性的属性值,即定义在图形窗口移动鼠标时应该运行的 callback。这样,一旦用户在控制按钮上按下鼠标键,MATLAB 就为图形窗口定义 Window Button Motion Fcn 属性值。当用户继续移动(此时还未放开鼠标键)鼠标时,MATLAB 就会执行刚定义的 Window Button Motion Fcn 属性的 callback。也就是获取图形窗口 Current Point

属性值,即鼠标箭头的当前位置的坐标值,随后将坐标值赋给向量 posvec 的前两个分量;最后一条语句定义图形窗口的 Window Button Up Fcn 属性。这样,当用户释放鼠标键时,就执行该属性的 callback,也就是重新将 WindowButton Motion Fcn 属性定义为空矩阵,即无任何操作;再运行 curpos 中的语句,即把控制按钮 pb 的 Position 属性置为更新后的 posvec,相当于将控制按钮移到了 posvec 决定的位置上。

这个例子定义了 3 个 callback,分别关联于按钮对象、移动鼠标箭头操作和释放鼠标键操作。还有几点值得说明:

- (a) 在上例中,WindowButtonDownFcn 的 callback 操作定义了与 WindowButtonMotionFcn 相关的 callback 操作。如果 WindowButtonMotionFcn 是在创建图形窗口的语句中定义的,那么鼠标箭头的任何移动都会移动按钮对象。这里是按下鼠标键时,才让 MATLAB 定义图形窗口的鼠标移动操作的 callback;
- (b) 毫无疑问,这种方法可以用来移动任何图形对象,如坐标系,等等。这是进行交互式程序设计的技巧。

5.5 中断 callback 操作

通常情况下,一旦 MATLAB 启动某个 callback,则 callback 不会中途被打断地执行完毕。中途打断 callback 的过程称为中断,引起中断的事件称为中断事件。程序设计者可以在设计某个对象的 callback 时,指明它的 callback 执行是否可以被其他的 callback 或其他的某些操作所中断。当然,某些情况下,不允许中断(打断)正在执行的 callback 过程是十分必要的。例如,当定义一个按钮启动某个较长的初始化过程时,就不希望被用户的某些操作,如鼠标的操作所打断。

对大部分的图形对象来说,它的 Interruptible 属性可以被用来设置是否可以对它的 callback 过程进行中断操作。属性 Interruptible 的取值可以是 yes 或 no(在 MATLAB 5.0 中,取值分别是 on 或 off)。

当 Interruptible 属性值是 no(或 off)时,图形对象的 callback 操作不能被其他的 callback 操作所中断;在 MATLAB 5.0 中,图形对象的 callback 操作是否能被其他的中断事件所中断,要由图形对象的 BusyAction 属性决定。

当 Interruptible 属性值是 yes(或 on)时,意味着该图形对象的 callback 操作可以被其他的 callback 操作中断。

例如,下面的语句定义了一个按钮对象,它的 callback 操作不允许被其他的 callback 操作中断:

```
>> pbstart = uicontrol(fig, 'Style', 'push', ...
    'Position', [10 10 75 25], ...
    'Interruptible', 'no', ...
    'CallBack', 'myprog1', 'ButtonDownFcn', 'myprog2');
```

一般说来,callback 的执行过程不会被中断,除非它遇到某种特别的 MATLAB 命令。在这种时候,MATLAB 系统开始处理存储在事件队列中的某些等待处理的事件。下一节将

就 MATLAB 的事件及事件队列的概念作些讨论。

5.5.1 事件及事件队列

MATLAB 有关数值计算和图形句柄设置操作的命令,一旦被系统获得后,即刻会被解释与执行。有些命令或动作,例如鼠标操作和重新刷新图形窗口的命令,在 MATLAB 系统中被分割成一个一个的事件(events)。

MATLAB 将这些事件临时存储在一个事件队列中,在图形对象的 callback 程序执行过程中,MATLAB 遇到下列命令时开始检测事件队列:drawnow、figure(包括 gef 和 gca 命令)、getframe、pause。遇到上述命令之一时,MATLAB 将暂时挂起 callback 程序的执行,转而处理事件队列中的事件。MATLAB 如何处理每件事件,依赖于各事件的类型和图形对象的 Interruptible 属性值。如果图形对象的 Interruptible 属性值是 yes(或 on),则对输入型事件,如按下鼠标键、释放鼠标键、移动鼠标箭头等,仅执行与这些事件相关的 callback 程序,如 WindowButtonDownFcn、WindowButtonUpFcn 和 WindowButtonMotionFcn 定义的 callback;如果图形对象的 Interruptible 属性值是 no(或 off),则视 BusyAction 属性值进行事件处理;不论对象的 Interruptible 属性值如何,刷新图形的事件总是立即执行。

但是,如果引起 MATLAB 挂起 callback 执行过程的是带 discard 参数的 drawnow 命令,那么上述三种类型的事件都不会被处理。有关更详细的说明,参看下一节。

5.5.2 MATLAB 处理 callback 的过程

MATLAB 按照下列步骤启动和处理 callback:

- (a) 在一个 callback 的执行过程中,当 MATLAB 遇到 drawnow、figure、getframe 和 pause 等命令时,MATLAB 会挂起当前的 callback 的执行,转向处理事件队列;
- (b) 如果导致挂起 callback 执行的是带 discard 参数的 drawnow 命令,或图形对象的 BusyAction 属性值为 cancel,MATLAB 将删除事件队列中的所有事件,然后转向步骤(d);
- (c) 如果事件队列中的第一个事件是刷新图形的事件,MATLAB 将执行有关的刷新工作,并开始处理下一个事件;如果事件队列中的第一个事件是启动另一个 callback 的执行,那么 MATLAB 将检测正在被挂起的 callback 对应的图形对象的 Interruptible 属性;如果 Interruptible 属性值为 yes(或 on),那么 MATLAB 就执行该中断事件的 callback,如果该 callback 中还包含有 drawnow、figure、getframe 和 pause 等命令,MATLAB 将按步骤(a)到(d)循环执行;如果 Interruptible 属性值是 no(或 off),MATLAB 将从事件队列中删除该事件或根据 BusyAction 属性值对事件队列进行处理,再处理事件队列中的下一个事件,直至将事件处理完为止;
- (d) 当事件队列空时,MATLAB 重新启动被挂起的 callback,继续执行该程序。MATLAB 持续地处理 callback,如果达到另一个 drawnow、figure、getframe 和 pause

等,重复上述步骤。下面看两个例子:

1. 不可中断的 callback

假设定义了一个按钮对象,它的 Interruptible 属性值为 no,该按钮对象的 callback 程序中包含一条 drawnow 命令。假设 MATLAB 正在执行按钮对象的 callback,此时,如果用户在某个已经定义了 WindowButtonDownFcn 属性的图形窗口中按下鼠标键,将按下列步骤处理有关的事件:

- (a) 当 MATLAB 遇到 drawnow 命令时,MATLAB 检查事件队列,就会发现用户在某个图形窗口中按下鼠标键的事件存在,并且该图形窗口的 WindowButtonDownFcn 属性定义了 callback;
- (b) MATLAB 检查按钮对象的 Interruptible 属性,看看是否可以执行那个图形窗口的 WindowButtonDownFcn 定义的 callback;
- (c) 根据假设,由于现在 Interruptible 属性值为 no,于是 MATLAB 从事件队列中去掉按下鼠标键这个事件;如果是在 MATLAB 5.0 系统中,还必须检查 BusyAction 属性值,再作处理;
- (d) MATLAB 继续处理事件队列,并且执行刷新图形的事件,但不执行其他的 callback。

注意:当 MATLAB 遇到的是带 discard 参数的 drawnow 命令时,甚至连刷新图形的操作也不执行;

- (e) 原来的 callback 继续执行。

如果原来的 callback 中不包含 drawnow(或 figure、getframe、pause)命令,那么不会处理按下鼠标键这一事件,直到 MATLAB 执行完 callback,回到它的起始状态。

2. 可中断的 callback

假设定义了一个按钮对象,它的 Interruptible 属性值为 yes,该按钮对象的 callback 程序中包含一条 drawnow 命令。又设某个图形窗口的 WindowButtonDownFcn 属性已被定义。一旦用户在这个图形窗口中按下鼠标键时,MATLAB 便按下列步骤处理有关的事件:

- (a) 当 MATLAB 执行到 drawnow 命令时,MATLAB 检查事件队列,就会发现在某个图形窗口中“按下鼠标键”的事件存在,并且该图形窗口的 WindowButtonDownFcn 有定义;
- (b) MATLAB 检查按钮对象的 Interruptible 属性,看看是否可以执行那个图形窗口的 WindowButtonDownFcn 定义的 callback;
- (c) 由于 Interruptible 属性值为 yes,于是 MATLAB 开始执行 WindowButtonDownFcn 定义的 callback;
- (d) 在 WindowButtonDownFcn 定义的 callback 执行完毕时,MATLAB 继续处理事件队列中的事件,处理完毕后,返回到原来的 callback。

然而,在中断过程中,MATLAB 中断前的状态未被保留下。如果原来的 callback 中不包含 drawnow(或 figure、getframe、pause)命令,那么就不会处理按下鼠标键的事件,直到 MATLAB 执行完 callback,回到它的起始状态。

5.5.3 事件的处理

1. 中断事件对 MATLAB 运行状态的影响

当 MATLAB 中断某个 callback, 转而处理其他某项事件时, MATLAB 不会保护它的当前状态及其有关的状态变量, 如当前图形窗口、当前坐标系、对象属性、工作空间的变量等。因此, 被中断的 callback 再次启动时, 它的某些执行条件可能发生了变化。所以在必要的时候, callback 程序应该自己设法保护它被中断前的某些状态和重要的变量值。

2. 移动鼠标箭头与事件队列

移动一次鼠标箭头的操作过程可能在事件队列中被分割成多个“移动鼠标”的事件, 这取决于计算机的系统设置, 以及鼠标移动的单步距离大小和移动速度。在处理事件队列中的事件时, 如果 MATLAB 发现移动鼠标箭头的事件, 它将检查事件队列中余下的事件, 并会删除相邻近的移动鼠标事件, 即把连续的几个移动鼠标事件当作一个移动事件来处理。

3. 保护 callback 不被中断

当执行某个 callback, 遇到 drawnow 命令时, MATLAB 开始处理事件队列。MATLAB 执行有关的刷新图形的命令时, 不论该对象的 Interruptible 的属性值是什么, 但是可以用带 discard 参数的 drawnow 命令来防止刷新图形的事件发生。在这种情况下, MATLAB 将删除事件队列中所有的事件。

5.6 GUI 工具集 Guide

为了更方便地设计 GUI 程序, MATLAB 5.0 提供了 GUI 程序设计工具集 Guide。工具集使得 GUI 程序设计变得方便简捷, 工具集 Guide 本身就是一个很好的 GUI 应用程序, 利用它可以进行“所见即所得”的窗口程序设计。Guide 工具集由 5 部分构成, 它们分别是: Guide 控制板, 属性编辑器, Callback 编辑器, 菜单编辑器和位置调整工具, 每种工具完成相应的任务。在 MATLAB 命令窗口中, 可以用行命令直接打开任何一种工具: guide 命令打开 Guide 控制板, propedit 命令打开属性编辑器, cbedit 命令打开 Callback 编辑器, menuedit 命令打开菜单编辑器, align 命令打开位置调整工具。另外, 后 4 种工具还可以在 Guide 控制板中启动。下面对各种工具分别作简单介绍。

5.6.1 Guide 控制板

Guide 控制板是 Guide 工具集的集成环境, 它使得生成图形对象和管理图形对象的过程变得简单和直接。例如, 可以很方便地访问图形对象的各种属性, 可以方便地在图形窗口中将任何图形对象拖动到任意的位置上, 使得整体布局更加美观。几乎所有的图形对象创建工具都集成在 Guide 控制板中, 这样, 程序设计者可以在 Guide 控制板中调用任何所需要的工具。

整个 Guide 控制板分为以下 3 部分:

- Guide 工具表, 一组用于启动四种工具的按钮;
- Guide 控制的图形窗口列表;
- 图形对象创建工具栏, 用于在各图形窗口中添加坐标系对象和控制元对象。

Guide 控制板图形窗口如图5-9所示。在第一排 Guide 工具表上, 点按任何一个按钮, 即

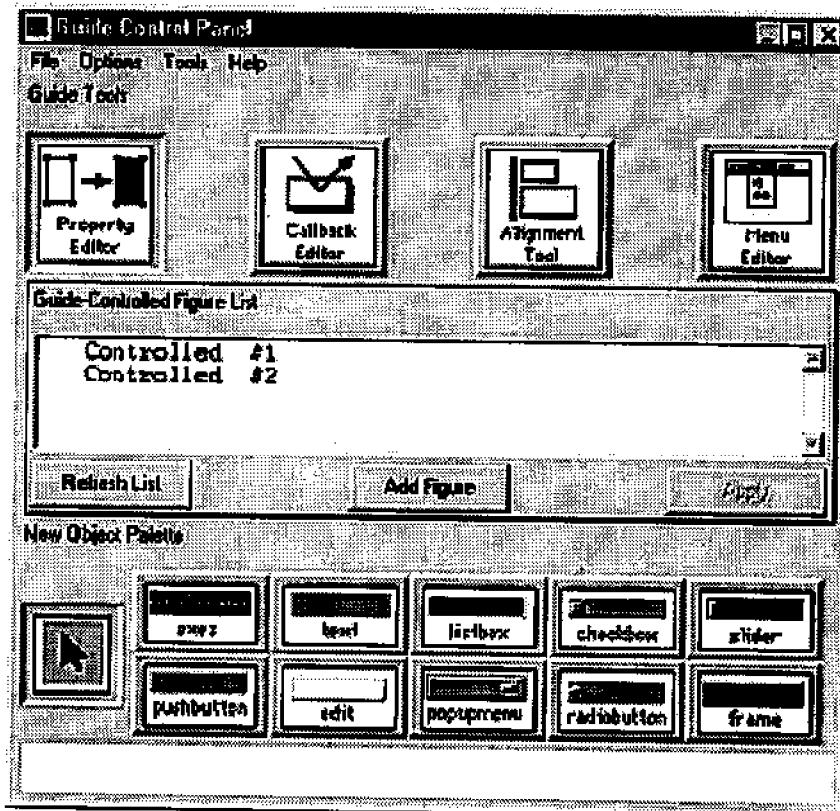


图5-9 Guide 控制板图形窗口

可打开相应的工具。第二排是图形窗口列表框, 与图形窗口列表框相关的概念是所谓的(控制窗口与活动窗口)。在 Guide 意义下, 任何一个 MATLAB 图形窗口或者是控制窗口, 或者是活动窗口, 这两种概念是相对的。控制窗口是指那些被 Guide 控制板管理的图形窗口, 可以用 Guide 控制板提供的任何工具对其进行管理和操作, 且该图形窗口在受控期间不能够完成正常的操作功能; 相反地, 活动窗口是通常意义上的图形窗口, 用户能够对其操作完成正常功能和任务, 例如, 活动图形窗口中的按钮可以按动, 弹出式菜单可以打开, 等等。

Guide 控制板可以让图形窗口在两种状态之间转换, 如果需要对某个图形窗口进行编辑, 例如, 要改变其中某个按钮的位置, 那么可以先将该图形窗口转变为 Guide 的控制窗口。

当从命令行用命令 guide 启动 Guide 控制板时, 总有一个图形窗口成为控制窗口。为了将某个图形窗口转变为控制窗口, 只要在图形窗口列表框中点按相应的图形窗口名字, 再点按 Apply 按钮即可。在控制窗口中, 可以进行如下的工作:

- 选中控制元对象和坐标系对象；
- 移动控制元对象和坐标系对象；
- 创建或删除控制元对象和坐标系对象。

Guide 控制板可以同时控制多个图形窗口。如果在控制窗口中的修改工作完成后，在 Guide 控制板的图形窗口列表框点按相应的列表行，再点按 Apply 按钮，那么该图形窗口便回复到活动窗口状态。

Guide 控制板只能对控制窗口完成图形窗口的编辑工作，即对控制元对象和坐标系对象进行操作。Guide 控制板的其他工具可以对任何图形窗口进行操作，例如，可以重新设置图形窗口的背景颜色，不管它是否是控制窗口。

5.6.2 属性编辑器

属性编辑器是 GUI 工具之一，用于对 MATLAB 图形对象属性进行操作和管理，代替命令行形式的命令 set 和 get。属性编辑器与命令 set 和 get 在功能上是一样的，但属性编辑器更容易操作和使用。

可以在 MATLAB 命令窗口中用 propedit 命令启动属性编辑器，也可以在 Guide 控制板中启动属性编辑器。属性编辑器启动后，MATLAB 打开一个属性编辑图形窗口进行交互式的编辑工作。下面通过一个简单例子说明属性编辑器的使用方法。

首先，用下列命令创建图形窗口，并打开属性编辑器：

```
>> surf(peaks(25))
>> propedit(gcf)
```

打开的属性编辑器窗口如图 5-10 所示。为了将图形窗口的背景颜色改换为红色，可以使

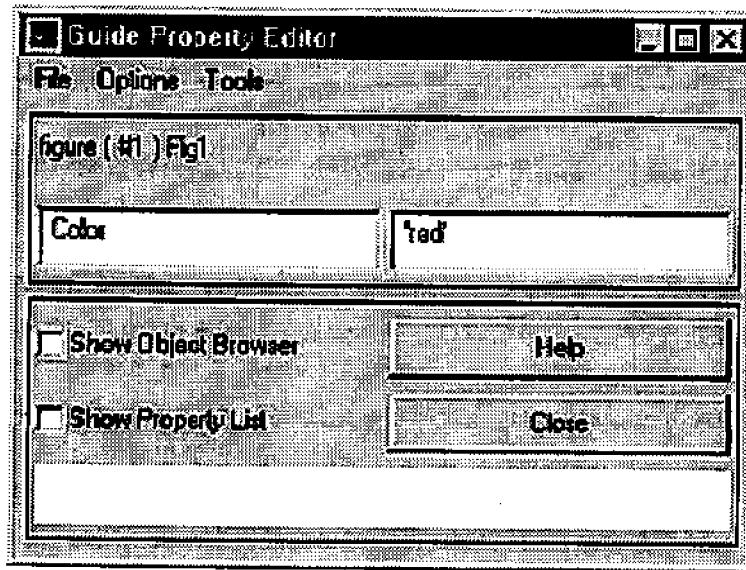


图 5-10 属性编辑器窗口

用如下命令：

```
>> set(gcf,'Color',[1 0 0])
```

用属性编辑器可以做与上述语句同样的工作。在属性编辑器的 figure (#1) 标题下，有一个编辑框，称为(属性域)。在其中输入属性名 Color，不要引号。回车后，Color 属性的缺省值就会出现在另一个编辑框中，称为值域。将该缺省值改为 red(带单引号)或[1 0 0]，回车后，相应的图形窗口背景颜色就变为红色。如同使用 set 命令一样，也必须提供“属性名/属性值”对。

在上述方法中，由于必须向属性编辑器提供相应的属性名，故使用起来不够方便。为了避免这一点不足，属性编辑器提供了属性列表框。在属性编辑器中，用 Show Property List 检测框打开属性列表框，如图5-11(a)所示。在属性列表框中列出被编辑的图形对象的全部属性和当前的属性值。为了编辑某个属性，只要点按相应的属性列项，相应的属性名和属性值就会出现在属性域和值域编辑框中，然后，改变值域中的属性值即可。

在前面的例子中，如果想要修改图形窗口中坐标系对象的某个属性，那么可以用 propedit 命令：

```
>> propedit(gca)
```

将属性编辑器对准图形窗口中的当前坐标系对象，如前所述，对坐标系对象的属性进行编辑。另一方面，属性编辑器以及 Callback 编辑器和位置调整工具还提供了图形对象浏览功能。这项功能可以在图形对象树型结构层次上管理图形对象。

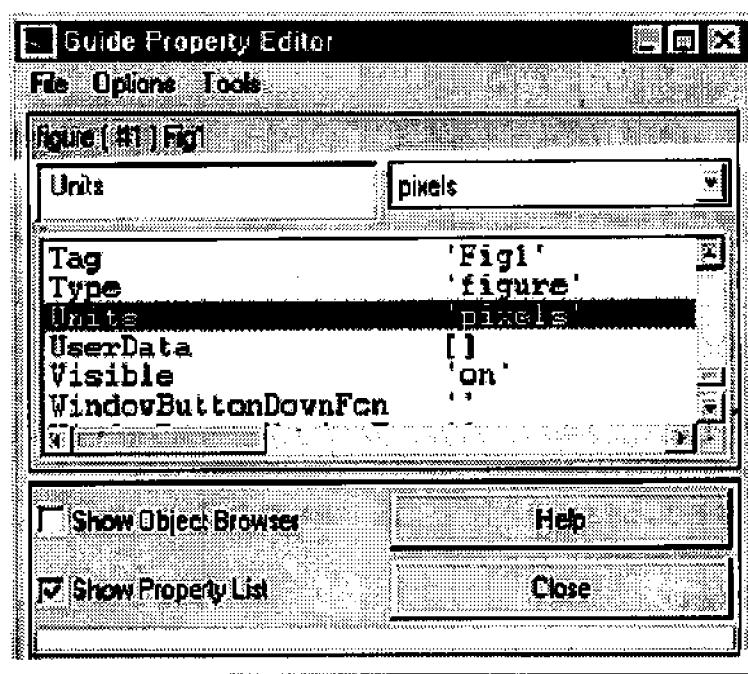
可以用 Show Object Browser 检测框打开图形对象列表框。在图形对象列表框中会列出当前图形窗口中的图形对象及其层次结构，如图5-11(b)所示。对于前面的例子，只要在图形对象列表框中选择坐标系对象，即可将属性编辑器对准到坐标系，再对其属性进行必要的编辑。如果此时属性列表框是打开的，那么该图形对象的所有属性就在属性列表框中列出，如图5-12所示。

此外，属性编辑器还可以提供如下智能性的操作：

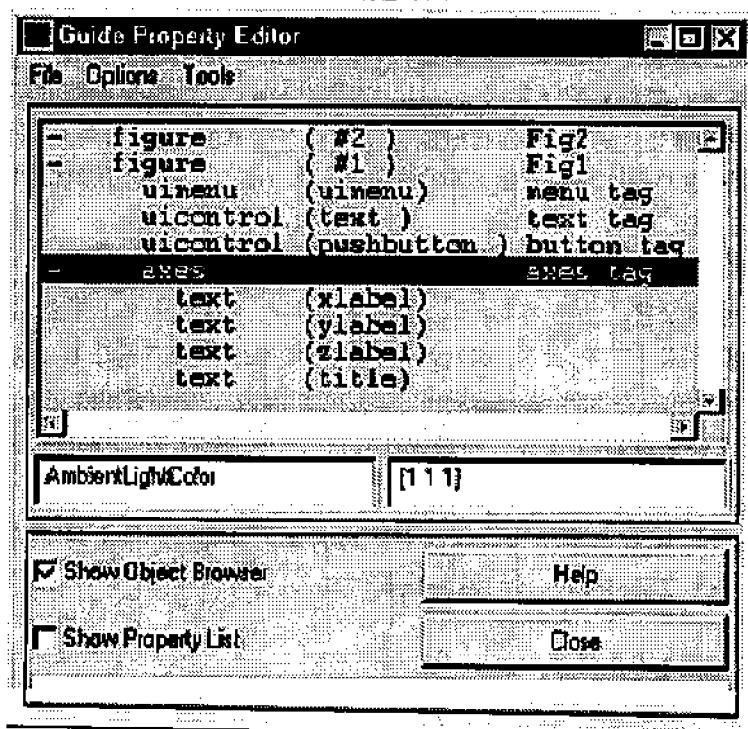
- 在属性编辑器的图形对象列表框中，可以同时选择多个图形对象，甚至可以是不同类型的图形对象。此时，这些被选择的图形对象的公共属性列在属性列表框中；
- 在属性编辑器的属性域编辑框中，只要键入可以区分的部分属性名即可。例如，如果正在对图形窗口对象的属性进行编辑，那么在属性域中键入“Col”即可以代替整个“Color”属性名；
- 在属性域中，属性编辑器忽略含有空格的输入。所以，如果出现了输入错误，只要键入一个空格再回车，就可清除先前的输入。

5.6.3 Callback 编辑器

用 Callback 编辑器可以修改某个图形对象的 Callback 属性值。在 Callback 编辑器的编辑框中可以编辑多行的 callback 代码。最为方便的是，在 Callback 编辑器的编辑框中可以省略 callback 代码中所需的单引号。



(a) 属性列表框



(b) 图形对象列表框

图5-11 属性列表框和图形对象列表框窗口

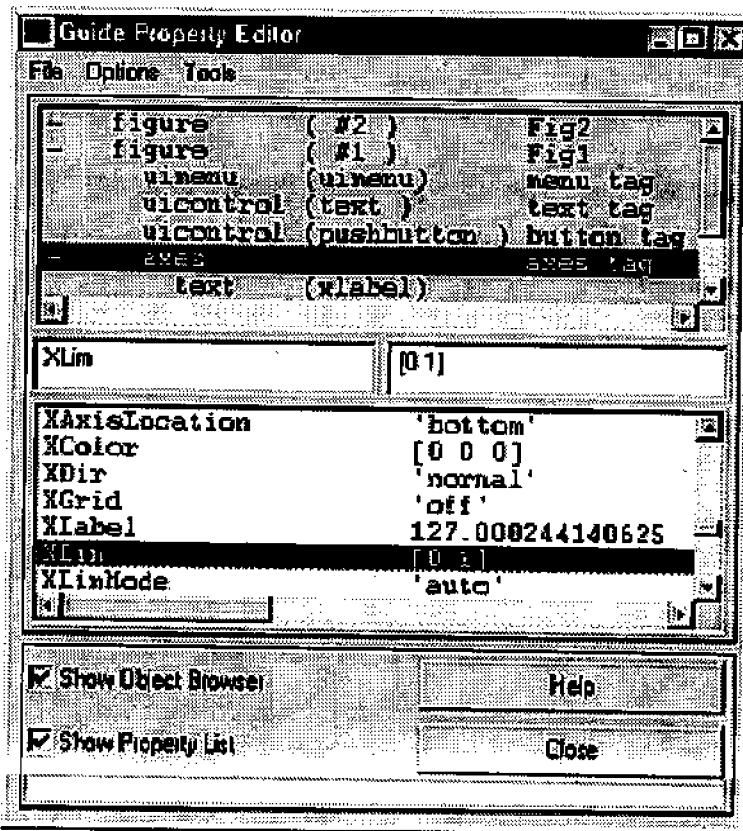


图5-12 属性编辑器和列表框窗口

同样,还可以用 cedit 命令启动 Callback 编辑器,也可以从 Guide 控制板中启动 Callback 编辑器。Callback 编辑器的图形窗口如图5-13所示。

可以直接用 Callback 编辑器对图形窗口对象的 callback 进行修改。例如,为了修改ButtonDownFcn 属性,首先用 Show Object Browser 检测框列出图形对象,选择ButtonDownFcn,在 Callback 编辑器的编辑框中编辑希望执行的代码,然后用 Apply 按钮完成操作。

5.6.4 菜单编辑器

菜单编辑器既可以在某个图形窗口的菜单条上添加用户菜单,也可以编辑已定义的某个用户菜单。有一点要注意,在图形窗口成为活动窗口之前,用菜单编辑器加入或修改的用户菜单不会出现在图形窗口的菜单条上。菜单编辑器的图形窗口如图5-14所示,其使用方法与其他的工具相同。

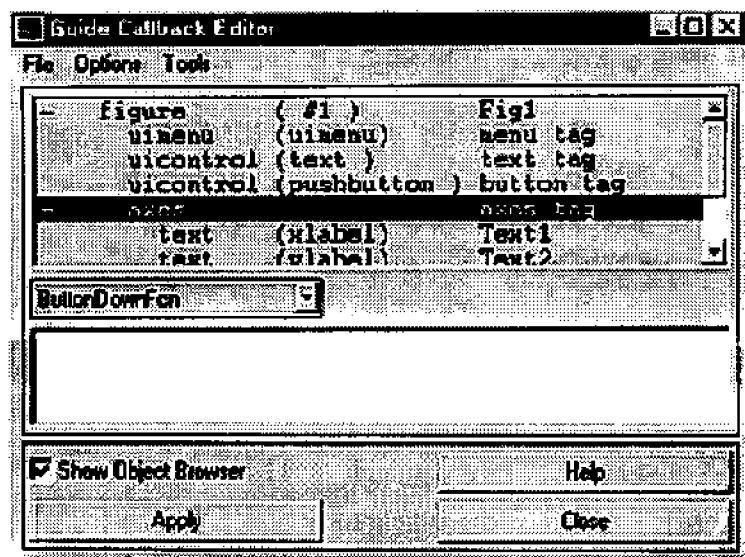


图5-13 Callback 编辑器窗口

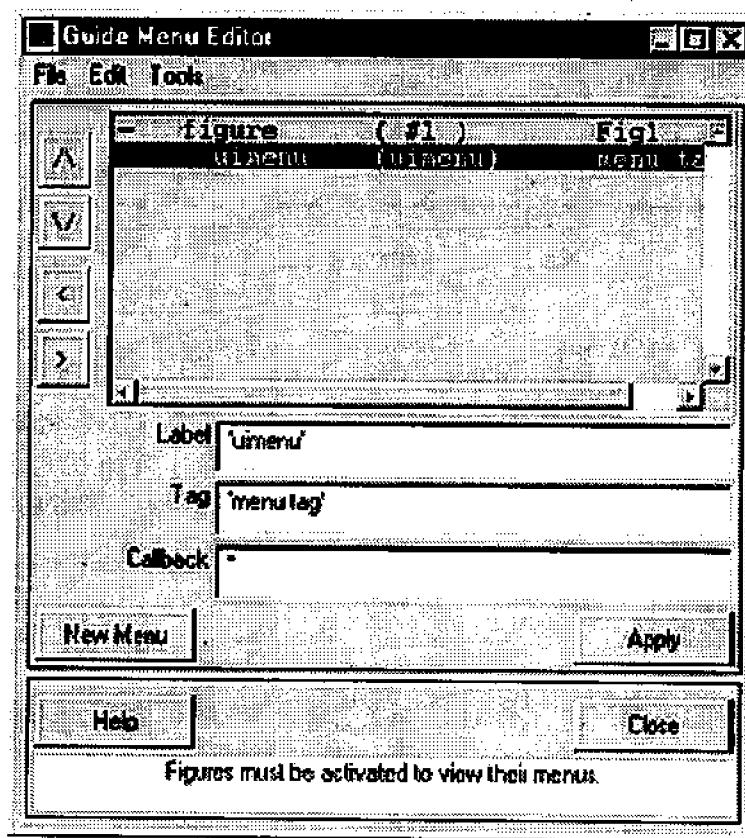


图5-14 菜单编辑器窗口

5.6.5 位置调整器

位置调整器主要用于对图形对象进行几何位置的调整,使得 GUI 程序的图形界面更加美观。在命令行式的 GUI 程序设计中,需要仔细地计算每个图形对象在图形窗口中的位置坐标,且不能直接面对所设计的界面。无疑,这给程序设计带来许多麻烦。利用位置调整器,这个难题就迎刃而解。

位置调整器的图形窗口如图 5-15 所示,它由对象列表框和调整工具按钮组成。在选择了图形对象后,就可以用各种按钮来移动被选的图形对象,或调整它与其他图形对象的相对位置。

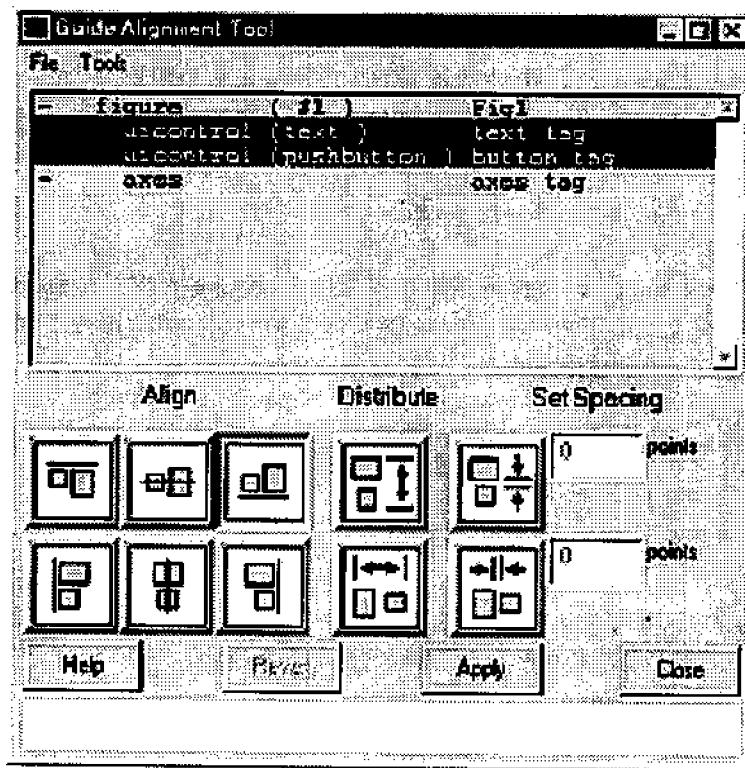


图 5-15 位置调整工具窗口

第六章 小波 (Wavelet) 分析工具包

本章将给出一个综合例子,即小波分析工具包,这是作者为分析化学光谱信号而开发的。可以把它看作是一个 GUI 程序设计的例子。限于篇幅,这里只给出了部分函数,其主要功能有:计算 Daubechies 小波系数;由小波系数生成小波函数及图形;计算一维小波变换及其逆变换;用 threshold 方法压缩一维信号,并绘制信号及其恢复信号的图形等等。

6.1 主 程 序

主程序名为 hplccomp.m,其作用是组合各子程序,构造各种图形对象,传递计算信息等等。并创建一个交互式的用户界面。读者可先从主结构分析这个应用程序,即按照第二个 if 语句的结构来分析,将主程序变为

```
if strcmp(lower (Action), 'new)
    语句组
elseif strcmp (Lower (Action), 'initial')
elseif strcmp (Lower (Action), 'Computing')
end
```

再逐步向每个 elseif 部分添加相应的语句组。这样,就可以弄清应用程序设计的一般技巧。每个 elseif 部分都是各子任务的细节。主程序源代码如下:

```
function hplecomp(Action)
% Filename:      HPLCCOMP.M
% Author:        Dr. Junbin Gao
% Date:          08--10--1996
% Version:       1.0
% Usages:        To compress a HPLC-DAD data by wavelet transform.

%Variables List
%=====
```

%Handles List

```
%=====
%Possible Action List
%=====

if nargin<1
    Action='new';
end

global Handlist WAVELETH WAVELETG WAVELETRH WAVELETRG
OKWAVELET
global FILENAME PARAMETER LEVELS THRESHOLD AVERAGBIT
ORDERM ORDERN
global RECONDATA WAVELETIDX METHODIDX INPUTFIGURE RATIO
RMSD
global PARAIDX HPLCDATA ORIGFILE

if strcmp(lower(Action),'new')

    %Program initialization
    clc;
    disp(' =====')
    disp(' | HPLC-DAD Data Compression |')
    disp(' | Version 1.0 (1996) |')
    disp(' =====')
    disp(' | Prof. F. T. Chau, Dr. J. B. Gao and Mr. K. M. Leung |')
    disp(' | The Hong Kong Polytechnic University |')
    disp(' | Department of Applied Biology and Chemical Technology |')
    disp(' =====')
    disp(' ')
    pause(2)

    CurFigNum=figure;
    set(gcf,'units','normalized','NumberTitle','off','Name',...
        'HPLC-DAD DATA Compression',...
        'Position',[0.0375 0.0833 0.9 0.7333],'Resize','on');
```

```

%%Welcome Session
set(gcf,'visible','off')
hh0 = text(0.5, 0.85,' Welcome to HPLC-DAD DATA Compression
System');
hh1 = text(0.5, 0.68,' Version 1.0 (1996)');
hh2= text(0.5, 0.45,[' Prof. F. T. Chau, Dr. J. B. Gao and ',...
'Mr. K. M. Leung']);
hh3= text(0.5, 0.4, [' Department of Applied Biology and ',...
'Chemical Technology']);
hh4 = text(0.5, 0.35, 'The Hong Kong Polytechnic University');
set(hh0,'fontsize',16,'color','y','HorizontalAlignment','center');
set(hh1,'fontsize',14,'color','y','HorizontalAlignment','center');
set([hh2, hh3, hh4],'fontsize',12,'color','c',...
'HorizontalAlignment','center');
uicontrol('position',[0.2 0.05 0.2 0.1],'style','pushbutton',...
'enable','on','string','Continue','units','normalized',...
'callback','hplccomp("initial")');
uicontrol('position',[0.6 0.05 0.2 0.1],'style','pushbutton',...
'string','Exit','units','normalized','callback','close(gcf)');
elseif strcmp(lower(Action),'initial')
    delete(get(gcf,'children'))
    drawnow

%Parameters initialization
OKWAVELET = 0;
ORDERM = 8;
ORDERN = 12;
AVERAGBIT = 6;
THRESHOLD = 0.03;
[WAVELETH, WAVELETG, WAVELETRH, WAVELETRG] = daub
(ORDERM);
PARAMETER=str2mat("'",Wavelet: Daubechies 8', 'Levels: 4',...
'Method: Thresholding', 'Threshold Value: 0.03');
LEVELS = 4;
HPLCDATA = [];
WAVELETIDX = 1;
METHODIDX = 1;

```

```
PARAIDX = 0;

%%%%Frame
uicontrol('position',[0.005 0.005 0.210 0.995],...
    'backgroundcolor',[0.1 0.1 0.1],'enable','on',...
    'style','frame','units','normalized');

%%%%Text Handles
uicontrol('position',[0.010 0.7875 0.200 0.05],...
    'backgroundcolor',[0.1 0.1 0.1],...
    'Foregroundcolor',[1 1 1],'string','Wavelet Types',...
    'style','text','units','normalized');
uicontrol('position',[0.010 0.6375 0.200 0.05],...
    'backgroundcolor',[0.1 0.1 0.1],...
    'Foregroundcolor',[1 1 1],'string','Compress Method',...
    'style','text','units','normalized');
uicontrol('position',[0.010 0.4875 0.200 0.05],...
    'backgroundcolor',[0.1 0.1 0.1],...
    'Foregroundcolor',[1 1 1],'string','Maximal Levels',...
    'style','text','units','normalized');

ParaUsed=uicontrol('position',[0.29 0.89 0.30 0.05],...
    'backgroundcolor',[0.1 0.1 0.1],...
    'foregroundcolor',[1 1 1],'string','Default Parameters',...
    'style','text','visible','off','units','normalized',...
    'horizontalalignment','left');

%%%%%%Display the parameter used
EditPara=uicontrol('position',[0.29 0.71 0.30 0.18],...
    'backgroundcolor',[0 0 0],...
    'foregroundcolor',[1 1 0],'style','edit',...
    'min',0,'max',3,'string','PARAMETER','visible','off',...
    'units','normalized');

drawnow

%%%%PopupMenu
%%%%%Select wavelet function to be used;
PopWavelet=uicontrol('position',[0.01 0.7375 0.2 0.05],...
    'backgroundcolor',[1 1 0],'string',['Daubechies Wavelet',...
    'Haar Wavelet','Mexican Hat Wavelet','Gabor Wavelet',...
    'Morlet Wavelet','Mexican Hat Wavelet','Gabor Wavelet',...
    'Morlet Wavelet']);
```

```

' |Spline Wavelet|Maximal Flat Wavelet|Least Asymmetric ',...
'Daub|Battle Lemarie Wavelet' ],...
'style','popupmenu','value',1,'units','normalized',...
'callback','hplecomp("Wavelet")');

%%%%%Select the compression method;
PopMethod=uicontrol('position',[0.01 0.5875 0.2 0.05],...
    'backgroundcolor',[1 1 0],'string','Thresholding|Coding',...
    'style','popupmenu','value',1,'units','normalized',...
    'callback','hplecomp("Method")');

%%%%%Transform Levels;
PopLevels=uicontrol('position',[0.01 0.4375 0.2 0.05],...
    'backgroundcolor',[1 1 0],'string','Level 1|Level 2',...
    'Level 3|Level 4|Level 5|Level 6|Level 7|Level 8',...
    'Level 9|Level 10|Level 11|Level 12'],...
    'style','popupmenu','value',4,'units','normalized',...
    'callback','hplecomp("Levels")');

%%%%Button;
%%%%%Load data files
ButLoad=uicontrol('position',[0.01 0.8875 0.2 0.05],...
    'string','Loading Data',...
    'style','pushbutton','units','normalized',...
    'callback','hplecomp("load")');

%%%%%Set default parameter needed
ButDefault=uicontrol('position',[0.01 0.3375 0.2 0.05],...
    'string','Default Parameter',...
    'style','pushbutton','units','normalized',...
    'callback','hplecomp("setDefault")');

%%%%%Start computing
ButComput=uicontrol('position',[0.01 0.2375 0.2 0.05],...
    'string','Computing',...
    'style','pushbutton','units','normalized',...
    'callback','hplecomp("computing")');

%%%%%Save computation results
ButSave=uicontrol('position',[0.01 0.1375 0.2 0.05],...
    'string','Save Results',...
    'style','pushbutton','units','normalized',...
    'callback','hplecomp("save")');

```

```
%%%%Exit program
uicontrol('position',[0.01 0.0375 0.2 0.05],...
    'string','Exit',...
    'style','pushbutton','units','normalized',...
    'callback','close(gcf);

%Axis
%%%Display Spline Wavelet
AxisSpline=axes('position',[0.29 0.6 0.30 0.32],...
    'box','on','units','normalized',...
    'visible','off');
set(AxisSpline,'fontsize',9,'fontname','times')

%%%Display the original HPLC-DAD data
AxisOrig=axes('position',[0.69 0.6 0.30 0.32],...
    'box','on','units','normalized',...
    'visible','off');
set(AxisOrig,'fontsize',9,'fontname','times')

%%%Display the transformed results
AxisTrans=axes('position',[0.29 0.1 0.30 0.32],...
    'box','on','units','normalized',...
    'visible','off');
set(AxisTrans,'fontsize',9,'fontname','times')

%%%Display the compressed HPLC-DAD data
AxisComp=axes('position',[0.69 0.1 0.30 0.32],...
    'box','on','units','normalized',...
    'visible','off');
set(AxisComp,'fontsize',9,'fontname','times')

%%%Display the message
AxisMessage = axes('position',[0.29 0.1 0.7 0.8],...
    'box','off','visible','off',...
    'units','normalized');
MESSAGE = text(0.5,0.5,'Starting Program ...');
set(MESSAGE,'fontsize',16,'color','y',...
    'HorizontalAlignment','center');
```

```

Handlist=[EditPara, ParaUsed, AxisOrig, AxisTrans, AxisComp, ...
          AxisSpline, AxisMessage, MESSAGE, 0, 0, PopWavelet, ...
          PopMethod, PopLevels, 0, 0, 0, 0, 0, 0, 0, 0, 0];

pause(1)
set(Handlist(8),'string','Setting Default Parameters ...');
pause(1)

set(Handlist(8),'visible','off');
set(Handlist(1),'visible','on');
set(Handlist(2:5),'visible','on');

elseif strcmp(lower(Action),'computing')
    if length(get(Handlist(7),'children'))>1
        delete(Handlist(16:22));
    end
    set(Handlist(1:6),'visible','off')
    set(get(Handlist(3),'children'),'visible','off')
    set(get(Handlist(4),'children'),'visible','off')
    set(get(Handlist(5),'children'),'visible','off')
    set(get(Handlist(6),'children'),'visible','off')
    if isempty(HPLCDATA)
        set(Handlist(8),'string',[ 'No HPLC-DAD Data Loaded,'...
            'Please Load Data First ... !' ],...
            'visible','on')
    else
        if METHODIDX == 1
            meth_thr
        elseif METHODIDX == 2
            disp('Method Coding')
        end
    end
    PARAIDX = 1; % Identifying New Results to be Saved

elseif strcmp(lower(Action),'setdefault')
    PARAMETER=str2mat(",'Wavelet:Daubechies 8','Levels:4',...
        'Method: Thresholding','Threshold Value: 0.03');

```

```
set(Handlist(1),'string',PARAMETER);
set(Handlist(2),'string','Default Parameters')
LEVELS = 4;
ORDERM = 8;
ORDERN = 12;
AVERAGBIT = 6;
THRESHOLD = 0.03;
[WAVELETH, WAVELETG, WAVELETRH, WAVELETRG] = daub
(ORDERM);
set(Handlist(13),'value',LEVELS);
set(Handlist(11),'value',1);
for m=3:6
    eval(['delete(get(Handlist(' ,num2str(m),'),"children"))']);
    eval(['axes(Handlist(' ,num2str(m),'))']);
    title("");
    xlabel("");
    ylabel("");
end
if length(get(Handlist(7),'children'))>1
    delete(Handlist(16:22));
end
PARAIDX = 0;% Identifying Parameters Changed
set(Handlist(6),'visible','off')
set(Handlist(1:5),'visible','on')

elseif strcmp(lower(Action),'load')
[SpecFile, Path]=uigetfile('*.dat','Get HPLC-DAD Data');
set(Handlist(1:6),'visible','off')
for m=3:6
    eval(['delete(get(Handlist(' ,num2str(m),'),"children"))']);
    eval(['delete(get(Handlist(' ,num2str(m),'),"xlabel"))']);
    eval(['delete(get(Handlist(' ,num2str(m),'),"ylabel"))']);
    eval(['delete(get(Handlist(' ,num2str(m),'),"title"))']);
end
if length(get(Handlist(7),'children'))>1
    delete(Handlist(16:22));
end
set(Handlist(8),'string','Loading Data ...','visible','on');
```

```

watchon;
drawnow
FILENAME = [Path, SpecFile];
ORIGFILE = SpecFile;
HPLCDATA = file2var(FILENAME);
watchoff;
PARAIDX = 0;      % Loaded New Data, to be computed again
set(Handlist(8),'visible','off')
set(Handlist(1:5),'visible','on')

elseif strcmp(lower(Action),'save')
if PARAIDX == 1
    result; % Call M-file ' result'
%     disp(' save all OK')
else
    set(Handlist(1:6),'visible','off')
    set(get(Handlist(3),'children'),'visible','off')
    set(get(Handlist(4),'children'),'visible','off')
    set(get(Handlist(5),'children'),'visible','off')
    set(get(Handlist(6),'children'),'visible','off')
    set(Handlist(8),'string',[ ' Parameters changed,' ...
        ' please compute first ... '],...
        'visible','on');
if length(get(Handlist(7),'children'))>1
    delete(Handlist(16:22));
end
drawnow
end

elseif strcmp(lower(Action),'wavelet')
WAVELETIDX=get(Handlist(11),'value');
PARAIDX = 0;
selwave(WAVELETIDX)

elseif strcmp(lower(Action),'method')
METHODIDX=get(Handlist(12),'value');
if METHODIDX == 1
    INPUTFIGURE = figure;

```

```
set(gcf,'NumberTitle','off','Name','Thresholding Method',...
    'position',[210 190 300 150],'Resize','off',...
    'MenuBar','none',...
    'color',[1 1 1],'units','normalized');
uicontrol('Position',[0.1 0.75 0.8 0.1],'style','text',...
    'backgroundcolor',[1 1 1], 'foregroundcolor',[0 0 0]...
    'string','Threshold Value Input','units','normalized');
Handlist(14)=uicontrol('Position',[0.4 0.55 0.2 0.15]...
    'style','edit','backgroundcolor',0.8*[1 1 1]...
    'foregroundcolor','red',...
    'string',num2str(THRESHOLD),'units','normalized');
uicontrol('position',[0.25 0.1 0.5 0.2]...
    'style','pushbutton','string','OK',...
    'callback','hplccomp("Threshvalue")',...
    'units','normalized');

elseif METHODIDX == 2
    INPUTFIGURE = figure;
    set(gcf,'NumberTitle','off','Name','OBA Coding Method',...
        'position',[210 190 300 150],'Resize','off',...
        'MenuBar','none','color',[1 1 1],...
        'units','normalized');
    uicontrol('Position',[0.1 0.75 0.8 0.1],'style','text',...
        'backgroundcolor',[1 1 1]...
        'foregroundcolor',[0 0 0]...
        'string','Average Bit Input','units','normalized');
    Handlist(15)=uicontrol('Position',[0.4 0.55 0.2 0.15]...
        'style','edit',...
        'backgroundcolor',0.8*[1 1 1]...
        'foregroundcolor','red',...
        'string',num2str(AVERAGBIT),'units','normalized');
    uicontrol('position',[0.25 0.1 0.5 0.2]...
        'style','pushbutton',...
        'string','OK','callback','hplccomp("obapara")',...
        'units','normalized');

end

elseif strcmp(lower(Action),'levels')
    LEVELS = get(Handlist(13),'value');
```

```

level = ['Levels: ', num2str(LEVELS)];
PARAMETER = str2mat(PARAMETER(1,:,:),PARAMETER(2,:,:),...
    level,PARAMETER(4,:,:),PARAMETER(5,:));
set(Handlist(1),'string',PARAMETER);
set(Handlist(2),'string','Parameters to be Used');
if length(get(Handlist(7),'children'))>1
    delete(Handlist(16:22));
end
set(Handlist(1:5),'visible','on')
set(get(Handlist(6),'children'),'visible','off')
set(Handlist(6),'visible','off')
PARAIDX = 0;

elseif strcmp(lower(Action),'threshvalue')
    PARAIDX = 0;
    THRESHOLD = str2num(get(Handlist(14),'string'));
    method = ['Method: Thresholding'];
    para = ['Threshold Value: ', num2str(THRESHOLD)];
    PARAMETER = str2mat(PARAMETER(1,:,:),PARAMETER(2,:,:),...
        PARAMETER(3,:),method,para);
    set(Handlist(1),'string',PARAMETER);
    set(Handlist(2),'string','Parameters to be Used');
    if length(get(Handlist(7),'children'))>1
        delete(Handlist(16:22));
    end
    set(Handlist(1:5),'visible','on')
    set([get(Handlist(6),'children'),Handlist(6),Handlist(8)],...
        'visible','off')
    close(INPUTFIGURE);

elseif strcmp(lower(Action),'obapara')
    PARAIDX = 0;
    AVERAGBIT = str2num(get(Handlist(15),'string'));
    method = ['Method: OBA Coding'];
    para = ['AverageBitValue: ', num2str(AVERAGBIT)];
    PARAMETER = str2mat(PARAMETER(1,:,:),PARAMETER(2,:,:),...
        PARAMETER(3,:),method,para);
    set(Handlist(1),'string',PARAMETER);

```

```

set(Handlist(2), 'string', 'Parameters to be Used');
if length(get(Handlist(7), 'children'))>1
    delete(Handlist(16,22));
end
set(Handlist(1,5), 'visible', 'on')
set(get(Handlist(6), 'children'), 'visible', 'off')
set(Handlist([6,8]), 'visible', 'off')
close(INPUTFIGURE)

elseif strcmp(lower(Action), 'view')
    viewwave
    close(INPUTFIGURE)
end

```

6.2 小波变换计算函数

一维小波变换，函数名为 wt1.m。这个函数有四个输入参数：x 是一维向量，即待进行小波变换的信号数据；h 是小波的低通滤波器系数；g 是相应的高通滤波器系数向量，k 是进行小波变换的层数；函数的输出变量 y 是经过小波变换以后的数据向量，按照低频成分，次高频成分，……高频成分的顺序存放。

```

function y=wt(x,h,g,k)
% x: The original signal matrix; y: The transformed signal matrix
% h: Low filter coefficients; g: High filter coefficients
% k: Levels to be transformed

%
% CHECK PARAMETERS AND OPTIONS
%
h=h(:)'; % Arrange the filters so that they are row vectors.
g=g(:)';
if length(x)<2^k
    disp('The scale is too high. The maximum for the signal is:')
    floor(log2(length(x)))
    return
end
[liy,lix]=size(x);
if liy==1      % And arrange the input vector to a row if
    x=x';      % it's not a matrix.

```

```

trasp=1;      % (and take note of it)
[liy,lix]=size(x);
end
%-----
% DELAY CALCULATION
%-----
% Calculate delays as the C.O.E. of the filters
dlp=wtcenter(h);
dhp=wtcenter(g);
if rem(dhp-dlp,2)^=0
    dhp=dhp+1;
end
%-----
% WRAPPAROUND CALCULATION
%-----
llp=length(h);    % Length of the lowpass filter
lhp=length(g);    % Length of the highpass filter,
L=max([lhp,llp,dlp,dhp]);
%-----
% START THE ALGORITHM
%-----
for it=1:liy,      % For every row of the input matrix...
    tm=[];
    t=x(it,:);
    odd_index = 0;
    for i=1:k
        lx=length(t);
        if rem(lx,2)^=0
            odd_index = 1;
            temp = t(lx);
            t = t(1:lx-1);
            lx=lx-1;
        end
        tp=t;
        pl=length(tp);
        while L>pl
            tp=[tp,t];
            pl=length(tp);
        end
        tm=[tm,tp];
    end
    x(it,:)=tm;
end

```

```

    end
    t=[tp(pl-L+1:pl),t,tp(1:L)];
    yl=conv(t,h); % Do lowpass filtering ...
    yh=conv(t,g); % ... and highpass filtering.
    yl=yl((dlp+1+L):2:(dlp+L+lx)); % Decimate the outputs
    yh=yh((dhp+1+L):2:(dhp+L+lx)); % and leave out wraparound
    if odd_index
        tm=[yh,temp,tm]; % Put the resulting wavelet step
        odd_index = 0;
    else
        tm = [yh, tm];
    end
    t=yl;
    end
    y(it,:)=[t,tm];
end           % End of the "rows" loop.
%-----
% END OF THE ALGORITHM
%-----
if trasp==1
    y=y';
end

```

目的是计算矩阵数据 x (二维)的小波变换,其中 h,g,k 的意义与一维小波变换函数 $wt1.m$ 中 h,g,k 的意义相同。 y 是输出变量,也是一个矩阵,它是 x 经过二维小波变换以后的结果,该矩阵的左上角部分对应变换后的低频成分,右下角部分对应变换后的高频成分。使用的算法是先对输入矩阵的每行作一维小波变换,再对每列作一维小波变换。二维小波变换,函数名 $wt2d1.m$ 。

```

function y=wt2d(x,h,g,k)
%-----
% CHECK PARAMETERS AND OPTIONS
%-----
h=h(:,');          % Arrange the filters so that they
g=g(:,');          % are row vectors.
[ly,lx]=size(x);   % The original image size
%-----
% SETUP WAVELET SIZES
%-----

```

```

lox(1)=lx; % Keep in two vectors the successive
loy(1)=ly; % wavelet widths and heights.
index_x = [];
index_y = [];
for i=1:k
    lox(i+1)=floor(lox(i)/2);
    index_x(i)=lox(i)-2 * lox(i+1);
    loy(i+1)=floor(loy(i)/2);
    index_y(i)=loy(i)-2 * loy(i+1);
end
l1=sum(lox(2:k+1))+lox(k+1)+sum(index_x);
l2=sum(loy(2:k+1))+loy(k+1)+sum(index_y);
%-----
% SETUP ARRAYS & OFFSETS
%-----
y=x; % Set up matrix.
px=l1-lx+1; % The offsets to put the wavelet
py=l2-ly+1; % sub-image into the final matrix
tx=x(1:2 * floor(ly/2),1:2 * floor(lx/2)); % Copy the original.
%-----
% START THE ALGORITHM
%-----
for ind=1:k % For every scale...
    [lxy,lxx]=size(tx);
    lwx=floor(lxx/2) * 2;
    lwy=floor(lxy/2) * 2;
    tw(1:lxy,1:lxx)=wt1(tx(1:lxy,1:lxx),h,g,1);
    %% And now over the columns %%
    tw(1:lxy,1:lxx)=wt1(tw(1:lxy,1:lxx)',h,g,1)';
    y(1:lwy,1:lwx)=tw;
    tx=tw(1:2 * floor(floor(lwy/2)/2),1:2 * floor(floor(lwx/2)/2));
    tw=[];
end % end of all scales.

```

一维逆小波变换,函数名为 iwt1.m。该函数使用 Mallat 算法计算一维逆小波变换,wx 是待变换的小波分量数据;rh 和 rg 是逆小波低通和高通滤波器;k 是变换的层数,应该与正变换的层数一致。tam 是 wx 的长度,可以不输入此参数,kl 是一个向量,指明在进行逆变换恢复信号时,应该使用的小波频段,如果不输入这个参数,表明由所有的频段恢复原信号。在

通常情况下,只输入前4个参数。

```

function y=iwt(wx,rh,rg,k,tam,kl)

%-----
% CHECK PARAMETERS AND OPTIONS
%-----
rh=rh(:)'; % Arrange the filters so that they are row vectors.
rg=rg(:)';
if nargin<6, % KL not given means reconstructing all bands.
    kl=k;
end;
if size(kl)==[1,1],      % If KL specifies the number of scales
    k1=zeros(1,k);        % then build the KL vector with KL ones
    k1(1:kl)=ones(1,kl); % and K-KL zeros.
    kl=k1;
else
    if length(kl)~=k,   % Make sure that KL is K elements long.
        disp(['KL should be a Single number (<=K) or ',...
               ' a vector with K elements']);
        return;
    end;
    for i=1:k           % And make sure that all nonzero elements in KL
        if kl(i)~=0, % are ones.
            kl(i)=1;
        end;
    end;
    [liy,lix]=size(wx);
    if lix==1           % If the input matrix is a column wavelet
        wx=wx';         % vector, we transpose it
        trasp=1;         % and take note of it.
        [liy,lix]=size(wx);
    end
%-----
% CHECK THE ORIGINAL LENGTH
%-----
if (nargin<5)
    tam = length(wx);

```

```

end;
if tam==0, % If the original size is unknown
    tam=maxrsize(lix,k); % the maximum possible will be set.
if tam==0,
    disp('Can''t determine the original length. K might be wrong');
    return;
end
end
lo(1)=tam;
res_index = []; % Keep in a vector the successive sizes of
for i=1:k, % the wavelet bands.
    lo(i+1)=floor(lo(i)/2);
    res_index(i) = lo(i)-2 * lo(i+1);
end
if lix~=sum(lo(2:k+1))+lo(k+1)+sum(res_index(1:k)),
    tam=maxrsize(lix,k);
    sprintf(1,['\nThe given size is not correct. ',...
        ' Trying default size: ']);
    fprintf(1, '%u\n',tam);
if tam==0,
    disp('No default size found. K might be wrong');
    return;
end;
lo(1)=tam;
res_index = [];
for i=1:k,
    lo(i+1)=floor((lo(i)+1)/2);
    res_index(i) = lo(i)-2 * lo(i+1);
end
if lix~=sum(lo(2:k+1))+lo(k+1)+sum(res_index(1:k)),
    disp('Default failed. K might be wrong');
    return
end;
fprintf(1,'(Check the result, may be wrong. If so, check K )\n');
end
%-----
% DELAY CALCULATION
%-----

```

```
llp=length(rh); % Length of the lowpass filter.  
lhp=length(rg); % Length of the highpass filter.  
suml = llp+lhp-2;  
difl = abs(lhp-llp);  
if rem(difl,2)==0  
    suml = suml/2;  
end;  
% Calculate analysis delays as the reciprocal M. C.  
dlpa=wtcenter(rg);  
dhpa=wtcenter(rh);  
if rem(dhpa-dlpa,2)~=0  
    dhpa=dhpa+1;  
end;  
dlp = suml - dlpa;  
dhp = suml - dhpa;  
%-----  
% WRAPPAROUND CALCULATION  
%-----  
L=max([llp,lhp,dhp,dlp]);  
%-----  
% START THE ALGORITHM  
%-----  
for it=1:liy,      % For every row in WX ...  
    ind=1+lo(k+1);  
    w=wx(it,1:lo(k+1)*2);  
    for i=1:k  
        lw=length(w);      % Length of the actual wavelet.  
        yl=w(1:lw/2);      % 1-The lowpass vector...  
        yl=[yl,zeros(1,length(yl))];  
        yl=yl(:)';  
        l=length(yl);  
        pyl=yl;  
        pl=length(pyl);  
        while L>pl  
            pyl=[pyl,yl];  
            pl=length(pyl);  
        end  
        yl=[pyl(pl+1-L:pl),yl,pyl(1:L)];
```

```

if (kl(i) ~= 0)
    yh=w((lw/2+1):lw); % 2-The highpass vector.
    yh=[yh;zeros(1,length(yh))];
    yh=yh(:,');
    pyh=yh;
    pl=length(pyh);
    while L>pl
        pyh=[pyh,yh];
        pl=length(pyh);
    end
    yh=[pyh(pl+1-L:pl),yh,pyh(1:L)];
end;

lx=length(yl);      % The length of each vector
yl=conv(yl,rh);    % Do the lowpass synthesis filtering
yl=yl(dlp+1+L:dlp+1+L);
if (kl(i) ~= 0)
    yh=conv(yh,rg);
    yh=yh(dhp+1+L:dhp+1+L);
    yl=yl+yh;          % Sum the two outputs
else
    yl=yl;            % or get only the lowpass side.
end;
lx=length(yl);
if res_index(k+1-i) == 1,
    lx=lx+1;
end
if i<k
    ind=ind+lo(k+2-i)+res_index(k+1-i);
    w=[y1,wx(it,(ind-1):(lo(k+1-i)+ind-1))];
end
if (i==k)&(res_index(1)==1)
    y1 = [y1, wx(ind+lo(2))];
end
end
y(it,:)=y1;      % One row vector reconstructed.
end           % End if all vectors reconstructed
%-----
% END OF THE ALGORITHM

```

```
%-----
if trasp==1      % If the input wavelet was a column vector
    y=y';          % then transpose the result.
end
```

辅助函数 `wtcenter.m`。这个函数由 `wtl.m` 和 `iwtl.m` 函数调用，它计算滤波器的延时偏差。用户不需要直接调用这个函数。

```
function d=wtcenter(x);
% WTCENTER Calculates the delay of filters for alignment.
% x: Filter coefficients;

global WTCENTERMETHOD
if size(WTCENTERMETHOD)==[0,0]
    WTCENTERMETHOD=0;
end
if WTCENTERMETHOD>3 | WTCENTERMETHOD<0
    WTCENTERMETHOD=0
end
d=floor(center(x,WTCENTERMETHOD));
```

辅助函数 `center.m`。该函数由函数 `wtcenter.m` 函数调用，计算4种模式下滤波器的延时偏差。

```
function d=center(x,op);
% CENTER Delay calculation for Wavelet transform alignment.
% x: Filter coefficient; op: Method index (0, 1, 2, 3)
% 0 : First Absolute Maxima Location
% 1 : Zero delay in analysis (Full for synthesis).
% 2 : Mass center (sum(m*d)/sum(m))
% 3 : Energy center (sum(m^2*d)/sum(m^2))

lx=length(x);
l=1:lx;
if op==1
    d=0;
else
    if op==2
        xx=abs(x(:)');
        L=l;
```

```

end
if op==3
    xx=x(:)' .^ 2;
    L=1;
end
if op==0
    [mx,d]=max(abs(x));
else
    d=sum(xx.*L)/sum(xx);
end
end
if nargout==0,
    cad='rgcbk';
    plot(x)
    l=line([d,d],[min(x),max(x)]);
    set(l,'Color',cad(op+1));
end

```

6.3 Daubechies 小波函数的生成函数

主函数 daub.m: 其目的是计算 Daubechies 型小波的正、逆滤波器系数, 输入参数是 Daubechies 小波函数的阶数, 必须是偶数。关于生成 Daubechies 小波函数的数学算法, 请读者参看有关小波分析的书籍。该函数输出四个向量, h, g 分别是正变换的低通和高通滤波器系数, 而 rh, rg 是逆变换的低通和高通滤波器系数。主程序调用这个函数, 根据用户的要求计算相应阶的 Daubechies 小波系数。

```

function [h,g,rh,rg]=daub(num_coefs)
%DAUB Generates Daubechies filters.

if rem(num_coefs,2)
    error('Error: NUM_COEFS must be even!!!')
end
N=num_coefs/2;
poly=trigpol(N); %Calculate trigonometric polynomial
zeros=roots(poly); %Calculate roots
% To construct rh for the minimum phase choice, we choose all
% the zero inside the unit circle.
modulus=abs(zeros);
j=1;

```

```
for i=1:(2*(N-1))
    if (modulus(i)<1)
        zerosinside(j)=zeros(i);
        j=j+1;
    end;
end;
if j== N
    error('Error!!!');
end
An=poly(1);
realzeros=[];
imagzeros=[];
numrealzeros=0;
numimagzeros=0;
for i=1:(N-1)
    if (imag(zerosinside(i))==0)
        numrealzeros=numrealzeros+1;
        realzeros(numrealzeros)=zerosinside(i);
    else
        numimagzeros=numimagzeros+1;
        imagzeros(numimagzeros)=zerosinside(i);
    end;
end;
cte=1;
for i=1:numrealzeros
    cte=cte * abs(realzeros(i));
end
for i=1:numimagzeros
    cte=cte * abs(imagzeros(i));
end
cte=sqrt(abs(An)/cte);
cte=0.5^N * sqrt(2) * cte;
% Construction of rh from its zeros
rh=[ 1 1];
for i=2:N
    rh=conv(rh,[1 1]);
end
for i=1:numrealzeros
```

```

    rh=conv(rh,[1 -realzeros(i)]);
end
for i=1:2:numimagzeros
    rh=conv(rh,[1 -2 * real(imagzeros(i)) abs(imagzeros(i))^2]);
end
rh=cte * rh;
[rh,rg,h,g]=rh2rg(rh);

```

辅助函数 trigpol.m:这个函数由 daub.m 函数调用,其目的是生成计算 Daubechies 小波函数所需要的三角多项式。用户不必直接使用这个函数。

```

function polinomio=trigpol(N)
% TRIGPOL generate trigonometric polynomial.

coefs=zeros(N,2*N-1);
coefs(1,N)=1;
for i=1:N-1
    fila=[1 -2 1];
    for j=2:i
        fila=conv(fila,[1 -2 1]);
    end;
    fila=numcomb(N-1+i,i)*(-0.25)^i*fila;
    fila=[ zeros(1,(N-i-1)) fila zeros(1,(N-i-1))];
    coefs(i+1,:)=fila;
end
for i=0:(2*(N-1))
    polinomio(i+1)=0;
    for j=1:N
        polinomio(i+1)=polinomio(i+1)+coefs(j,i+1);
    end
end

```

辅助函数 numcomb.m:用于计算二项式系数 C_n^k 。

```

function y=numcomb(n,k)
% NUMCOMB combinatorial number.

if n==k,
    y=1;
elseif k==0,

```

```

y=1;
elseif k==1,
    y=n;
else
    y=fact(n)/(fact(k)*fact(n-k));
end

```

辅助函数 fact.m:计算阶乘 x!

```

function y=fact(x)
% FACT Factorial.

for j=1:length(x)
    if x(j)==0,
        y(j)=1;
    else
        y(j)=x(j)*fact(x(j)-1);
    end
end

```

辅助函数 rh2rg.m:由逆变换低通滤波器系数计算其他三组滤波器系数。对于正交小波函数,具有基本的数学关系 $g_n = (-1)^n h_{1-n}$, 参见有关小波理论的书籍。该函数使用了这条数学关系。

```

function [rh,rg,h,g]=rh2rg(rh)
%RH2RG      Calculates all the filters from the synthesis lowpass
%              in the orthogonal case.

for i=1:length(rh)
    rg(i) = -(-1)^i * rh(length(rh)-i+1);
end
h=rh(length(rh):-1:1);
g=rg(length(rg):-1:1);

```

6.4 辅助函数

函数 meth_thr.m:这个函数没有输入参数,由主程序调用。它实现基于小波变换的信号去噪和压缩算法中的阈值方法。

```

function threshold()
%threshold method

```

```

global Handlist HPLCDATA WAVELETH WAVELETG WAVELETRH
WAVELETRG
global LEVELS THRESHOLD RECONDATA RATIO RMSD MINDATA
MAXDATA
global THRESHCOEFFS

set(Handlist(8),'string','Starting Computation ... ','visible','on');
drawnow
[sizx, sizy] = size(HPLCDATA);
sizmin = min(sizx,sizy);
if sizmin == 1
    data = HPLCDATA(:,1);
    MINDATA = min(data);
    MAXDATA = max(data);
    data = (data-MINDATA)/(MAXDATA-MINDATA);
    set(Handlist(8),'string','Calculating 1D Wavelet Transform ... ');
    drawnow
    wav_data = wt1(data, WAVELETH, WAVELETG, LEVELS);
    size(wav_data);
    index = (abs(wav_data)>=THRESHOLD);
    RATIO = (1-sum(index)/length(data)) * 100;
    THRESHCOEFFS = wav_data.*index;
    set(Handlist(8),'string',[ 'Calculating 1D Inverse ...
        'Wavelet Transform ... ']);
    drawnow
    RECONDATA = iwt1(THRESHCOEFFS, WAVELETRH,
                      WAVELETRG,... LEVELS,length(HPLCDATA));
    RMSD = sqrt(mean((data-RECONDATA).^2));
    RECONDATA=(MAXDATA-MINDATA)*RECONDATA
                +MINDATA;
    if sizx == 1
        RECONDATA = RECONDATA';
    end
    axes(Handlist(3))
    plot(HPLCDATA)
    set(gca,'fontsize',9,'fontname','times')
    set(gca, 'xlim',[0 length(HPLCDATA)])

```

```
title('Original Spectrum')
axes(Handlist(4))
plot(wav_data)
set(gca,'fontsize',9,'fontname','times')
set(gca,'xlim',[0 length(HPLCDATA)])
title('Transformed Spectrum')
axes(Handlist(5))
plot(RECONDATA)
set(gca,'fontsize',9,'fontname','times')
set(gca,'xlim',[0 length(HPLCDATA)])
title('Reconstructed Spectrum')
else
    [tamy, tamx] = size(HPLCDATA);
    data = HPLCDATA;
    MINDATA = min(min(data));
    MAXDATA = max(max(data));
    data = (data-MINDATA)/(MAXDATA-MINDATA);
    set(Handlist(8),'string','Calculating 2D Wavelet Transform ...');
    drawnow
    wav_data = wt2d1(data, WAVELETH, WAVELETG, LEVELS);
    index = (abs(wav_data)>=THRESHOLD);
    total = sum(sum(index));
    RATIO = 100 * (1-total/(tamy*tamx));
    THRESHCOEFS = wav_data.*index;
    set(Handlist(8),'string',['Calculating 2D Inverse',...
        'Wavelet Transform ...']);
    drawnow
    RECONDATA=iwt2d1(THRESHCOEFS,WAVELETRH,
                      WAVELETG,...LEVELS,tamx,tamy);
    RMSD=sqrt(mean(mean((data-RECONDATA).^2)));
    RECONDATA=(MAXDATA-MINDATA)*RECONDATA
              +MINDATA;
    axes(Handlist(3))
    show(HPLCDATA)
    colormap(gray)
    set(gca,'fontsize',9,'fontname','times')
    title('Original Spectrum')
    axes(Handlist(4))
```

```

ad_wave = bandadj1(wav_data,LEVELS);
show(ad_wave)
set(gca,'fontsize',9,'fontname','times')
title('Transformed Spectrum')
axes(Handlist(5))
show(RECONDATA)
set(gca,'fontsize',9,'fontname','times')
title('Reconstructed Spectrum')
end
set(Handlist(8),'visible','off')
set(Handlist(1:5),'visible','on')
%set(Handlist(9:12),'visible','on')

```

函数 selwave.m:这是主程序调用的辅助函数,它根据用户选定的小波函数类型,打开一个对话式的图形窗口,继续由用户提供各小波函数的参数,例如当用户选择了Daubechies型小波后,用户还需要在打开的对话窗口中输入阶数参数。该函数可创建很多的图形对象,读者可以从中学设计图形对象的技巧。

```

function selectwavelet(WAVELETIDX)
global INPUTFIGURE ORDERM ORDERN Handlist
global OKWAVELET

if WAVELETIDX == 1
    INPUTFIGURE = figure;
    set(gcf,'NumberTitle','off','Name',...
        'Daubechies Series Selection','position',...
        [210 190 300 150],'Resize','off','MenuBar','none',...
        'color',[1 1 1],'units','normalized');
    uicontrol('Position',[0.1 0.75 0.8 0.1],'style','text',...
        'backgroundcolor',[1 1 1],'foregroundcolor',0*[1 1 1],...
        'string','Wavelet Order (even in 2—20)', 'units','normalized');
    Handlist(9)=uicontrol('Position',[0.4 0.55 0.2 0.15],...
        'style','edit','backgroundcolor',0.8*[1 1 1],...
        'foregroundcolor','red',...
        'string',num2str(ORDERM), 'units','normalized');
    uicontrol('position',[0.05 0.1 0.4 0.2],'style','pushbutton',...
        'string','OK','callback',hplccomp("view");',...
        'units','normalized');
    uicontrol('position',[0.55 0.1 0.4 0.2],'style','pushbutton',...

```

```
'String','View Wavelet','callback',...
'global OKWAVELET, OKWAVELET=1; hplccomp("view")',...
'units','normalized');
elseif WAVELETIDX == 2
    INPUTFIGURE = figure;
    set(gcf,'NumberTitle','off','Name','Spline Wavelets Selection',...
        'position',[210 190 300 150],'Resize','off',...
        'MenuBar','none','color',[1 1 1]...
        'units','normalized');
    uicontrol('Position',[0.1 0.75 0.8 0.1],'style','text',...
        'backgroundcolor',[1 1 1], 'foregroundcolor',0*[1 1 1]...
        'string','Wavelet Order M <= N','units','normalized');
    uicontrol('Position',[0.3 0.55 0.05 0.15],'style','text',...
        'backgroundcolor',[1 1 1], 'foregroundcolor',0*[1 1 1]...
        'string','M=','units','normalized');
    Handlist(9)=uicontrol('Position',[0.35 0.55 0.1 0.15],...
        'style','edit','backgroundcolor',0.8*[1 1 1],...
        'foregroundcolor','red',...
        'string',num2str(ORDERM),'units','normalized');
    uicontrol('Position',[0.55 0.55 0.05 0.15],'style','Text',...
        'backgroundcolor',[1 1 1], 'foregroundcolor',0*[1 1 1]...
        'string','N=','units','normalized');
    Handlist(10)=uicontrol('Position',[0.6 0.55 0.1 0.15],...
        'style','edit',...
        'backgroundcolor',0.8*[1 1 1], 'foregroundcolor','red',...
        'string',num2str(ORDERN),'units','normalized');
    uicontrol('position',[0.05 0.1 0.4 0.2],'style','pushbutton',...
        'string','OK','callback',hplccomp("view");',...
        'units','normalized');
    uicontrol('position',[0.55 0.1 0.4 0.2],'style','pushbutton',...
        'String','View Wavelet','callback',...
        'global OKWAVELET, OKWAVELET=1;hplccomp("view")',...
        'units','normalized');
elseif WAVELETIDX == 3
    INPUTFIGURE = figure;
    set(gcf,'NumberTitle','off','Name',...
        'Maximally Flat Wavelets Selection','position',...
        [210 190 300 150],'Resize','off','MenuBar','none',...
        'units','normalized');
```

```

' color',[1 1 1],'units','normalized');
uicontrol('Position',[0.1 0.75 0.8 0.1],'style','text',...
    'backgroundcolor',[1 1 1], 'foregroundcolor',0*[1 1 1],...
    'string',' Wavelet Order NO = NPI','units','normalized');
uicontrol('Position',[0.25 0.55 0.1 0.15],'style','text',...
    'backgroundcolor',[1 1 1], 'foregroundcolor',0*[1 1 1],...
    'string',' NO=','units','normalized');
Handlist(9)=uicontrol('Position',[0.35 0.55 0.1 0.15],...
    'style','edit','backgroundcolor',0.8*[1 1 1],...
    'foregroundcolor','red',...
    'string',num2str(ORDERM),'units','normalized');
uicontrol('Position',[0.55 0.55 0.1 0.15],'style','Text',...
    'backgroundcolor',[1 1 1], 'foregroundcolor',0*[1 1 1],...
    'string','NPI=','units','normalized');
Handlist(10)=uicontrol('Position',[0.65 0.55 0.1 0.15],...
    'style','edit',...
    'backgroundcolor',0.8*[1 1 1], 'foregroundcolor','red',...
    'string',num2str(ORDERN),'units','normalized');
uicontrol('position',[0.05 0.1 0.4 0.2],'style','pushbutton',...
    'string','OK','callback','hplecomp("view");',...
    'units','normalized');
uicontrol('position',[0.55 0.1 0.4 0.2],'style','pushbutton',...
    'String',' View Wavelet','callback',...
    'global OKWAVELET, OKWAVELET=1;hplecomp("view")',...
    'units','normalized');
elseif WAVELETIDX == 4
INPUTFIGURE = figure;
set(gcf,'NumberTitle','off','Name',...
    'Least-asymmetric Daubechies Wavelets Selection',...
    'position',[210 190 300 150],'Resize','off',...
    'MenuBar','none','color',[1 1 1],...
    'units','normalized');
uicontrol('Position',[0.1 0.75 0.8 0.1],'style','text',...
    'backgroundcolor',[1 1 1], 'foregroundcolor',0*[1 1 1],...
    'string',' Wavelet Order (even in 2--20)',...
    'units','normalized');
Handlist(9)=uicontrol('Position',[0.4 0.55 0.2 0.15],...
    'style','edit','backgroundcolor',0.8*[1 1 1],...

```

```

'foregroundcolor','red',...
'string',num2str(ORDERM),'units','normalized');
uicontrol('position',[0.05 0.1 0.4 0.2],'style','pushbutton',...
    'string','OK','callback',hplccomp("view");',...
    'units','normalized');
uicontrol('position',[0.55 0.1 0.4 0.2],'style','pushbutton',...
    'String','View Wavelet','callback',...
    'global OKWAVELET, OKWAVELET=1;hplccomp("view")',...
    'units','normalized');
elseif WAVELETIDX == 5
    INPUTFIGURE = figure;
    set(gcf,'NumberTitle','off','Name',...
        'Battle-Lemarie Wavelets Selection','position',...
        [210 190 300 150],'Resize','off','MenuBar','none',...
        'color',[1 1 1],'units','normalized');
    uicontrol('Position',[0.1 0.75 0.8 0.1],'style','text',...
        'backgroundcolor',[1 1 1], 'foregroundcolor',0*[1 1 1],...
        'string','Wavelet Order (even in 16-40)',...
        'units','normalized');
    Handlist(9)=uicontrol('Position',[0.4 0.55 0.2 0.15],...
        'style','edit',...
        'backgroundcolor',0.8*[1 1 1], 'foregroundcolor','red',...
        'string',num2str(ORDERM),'units','normalized');
    uicontrol('position',[0.05 0.1 0.4 0.2],'style','pushbutton',...
        'string','OK','callback',hplccomp("view");',...
        'units','normalized');
    uicontrol('position',[0.55 0.1 0.4 0.2],'style','pushbutton',...
        'String','View Wavelet','callback',...
        'global OKWAVELET, OKWAVELET=1;hplccomp("view")',...
        'units','normalized');
end

```

函数 viewwave.m:本函数由主程序调用。当用户选定了某种小波函数后,如果希望看一看小波函数的图形,就可以在主程序中启动这个函数。这个函数调用 wavelet.m 函数计算小波,并绘出小波函数的图形。

```

function view()
global WAVELETIDX ORDERM ORDERN PARAMETER
global Handlist OKWAVELET

```

```

global WAVELETH WAVELETG WAVELETRH WAVELETRG

for m=3:6
    eval([' delete(get(Handlist(' ,num2str(m),'),"children"))' ]);
    eval([' delete(get(Handlist(' ,num2str(m),')," xlabel"))' ]);
    eval([' delete(get(Handlist(' ,num2str(m),')," ylabel"))' ]);
    eval([' delete(get(Handlist(' ,num2str(m),')," title"))' ]);
end
if length(get(Handlist(7),' children'))>1
    delete(Handlist(16:22));
end
if WAVELETIDX == 2,
    ORDERM = str2num(get(Handlist(9),' string'));
    ORDERN = str2num(get(Handlist(10),' string'));
    wavel = [' Wavelet: Spline M = ',num2str(ORDERM),...
              ' ; N = ',num2str(ORDERN)];
    PARAMETER = str2mat(PARAMETER(1,:),wavel,...%
                         PARAMETER(3,:),PARAMETER(4,:),PARAMETER
                         (5,:));
    [WAVELETH,WAVELETG, WAVELETRH, WAVELETRG] = ...
        wspline(ORDERM,ORDERN);
if OKWAVELET == 1
    lenh = length(WAVELETH);
    leng = length(WAVELETG);
    lenrh = length(WAVELETRH);
    lenrg = length(WAVELETRG);
    maxhrh = max(lenh, lenrh);
    maxgrg = max(leng, lenrg);
    [s,w]=wavelet(WAVELETRH,WAVELETG,6);
    hrev=WAVELETH(lenh:-1:1); grev=WAVELETG(leng:-1:1);
    [sd,wd]=wavelet(hrev,grev,6);
    s = s(:); sd = sd(:); w = w(:); wd = wd(:);
    lens = length(s);
    lenw = length(w);
    lensd = length(sd);
    lenwd = length(wd);
    maxs = max(lens, lensd);
    maxw = max(lenw, lenwd);

```

```

if maxs == lensd
    s = [zeros(fix((lensd-lens)/2),1); s; ...
          zeros(lensd-lens-fix((lensd-lens)/2), 1)];
else
    sd = [zeros(fix((lens-lensd)/2),1); sd; ...
          zeros(lens-lensd-fix((lens-lensd)/2), 1)];
end
if maxw == lenwd
    w = [zeros(fix((lenwd-lenw)/2),1); w; ...
          zeros(lenwd-lenw-fix((lenwd-lenw)/2), 1)];
else
    wd = [zeros(fix((lenw-lenwd)/2),1); wd; ...
          zeros(lenw-lenwd-fix((lenw-lenwd)/2), 1)];
end
axes(Handlist(6))
plot((linspace(0, maxhrh, maxs))',s);
set(gca,'fontsize',9,'fontname','times','xlim',[0 maxhrh])
title('Synthesis Scale Function')
axes(Handlist(3))
plot((linspace(0, maxhrh, maxs))',sd);
set(gca,'fontsize',9,'fontname','times','xlim',[0 maxhrh])
title('Analysis Scale Function');
axes(Handlist(4))
plot((linspace(0, maxgrg, maxw))',w);
set(gca,'fontsize',9,'fontname','times','xlim',[0 maxgrg])
title('Synthesis Wavelet Function')
axes(Handlist(5))
plot((linspace(0, maxgrg, maxw))', wd);
set(gca,'fontsize',9,'fontname','times','xlim',[0 maxgrg])
title('Analysis Wavelet Function');
set(Handlist(1:2),'visible','off')
OKWAVELET = 0;
else
    set(Handlist(1:5),'visible','on');
    set(Handlist(6),'visible','off');
end
else
    if WAVELETIDX == 1

```

```

ORDERM = str2num(get(Handlist(9),'string'));
wavel=['Wavelet:Daubechies',num2str(ORDERM)];
PARAMETER=str2mat(PARAMETER(1,:),wavel,PARAMETER
(3,:),...,PARAMETER(4,:),PARAMETER(5,:));
[WAVELETH,WAVELETG, WAVELETRH, WAVELETRG]=
daub(ORDERM);
elseif WAVELETIDX == 3
    ORDERM = str2num(get(Handlist(9),'string'));
    ORDERN = str2num(get(Handlist(10),'string'));
    wavel = ['Wavelet: Maximally Flat NO = ',num2str(ORDERM),...
        ', NPI = ',num2str(ORDERN)];
    PARAMETER = str2mat(PARAMETER(1,:),wavel,PARAMETER
(3,:),...,PARAMETER(4,:),PARAMETER(5,:));
    [WAVELETH,WAVELETG, WAVELETRH, WAVELETRG] = ...
        maxflat(ORDERM, ORDERN);
elseif WAVELETIDX == 4
    ORDERM = str2num(get(Handlist(9),'string'));
    wavel = ['Wavelet: Asymmetric Daub ', num2str(ORDERM)];
    PARAMETER = str2mat(PARAMETER(1,:),wavel,PARAMETER
(3,:),...,PARAMETER(4,:),PARAMETER(5,:));
    [ WAVELETH, WAVELETG, WAVELETRH, WAVELETRG ] =
        symlets(ORDERM);
elseif WAVELETIDX == 5
    ORDERM = str2num(get(Handlist(9),'string'));
    wavel = ['Wavelet: Battle-Lemarie ', num2str(ORDERM)];
    PARAMETER = str2mat(PARAMETER(1,:),wavel,PARAMETER
(3,:),...,PARAMETER(4,:),PARAMETER(5,:));
    [ WAVELETH, WAVELETG, WAVELETRH, WAVELETRG ] =
        lemarie(ORDERM);
end
if OKWAVELET == 1
    [s, w] = wavelet(WAVELETH, WAVELETG, 6);
    lenh = length(WAVELETH);
    leng = length(WAVELETG);
    lens = length(s);
    lenw = length(w);
    s = flipud(s(:));
    w = flipud(w(:));

```

```

axes(Handlist(4))
plot((linspace(0,lenh,lens))', s);
set(gca,'fontsize',9,'fontname','times','xlim',[0 lenh])
title(' Scale Function');
axes(Handlist(5))
plot((linspace(0,leng,lenw))', w);
set(gca,'fontsize',9,'fontname','times','xlim',[0 leng])
title(' Wavelet Function')
OKWAVELET = 0;
end
set(Handlist(1:5),'visible','on');
set(Handlist(6),'visible','off');
end
set(Handlist(1),'string',PARAMETER);
set(Handlist(2),'string','Parameters to be Used');

```

函数 wavelet.m: 这个函数由 view wave.m 函数调用。它的目的是根据细分(subdivision)迭代算法(参见有关小波理论的书籍),由小波滤波器的系数,计算并生成小波函数。j 是迭代次数,j 越大,计算的小波函数就越精确。

```

function [f,w]=wavelet(h,g,j)
%WAVELET    Construct the wavelet and scale functions
%                      associated to a pair of wavelet filters

h=h(:)';
g=g(:)';
hh=h;
gg=g;
if j==0,
    hh=1; gg=1;
end;
if j>1
    for k=2:j-1
        h1=[h;zeros(2^(k-1)-1,length(h))];
        h1=h1(:)';
        h1=h1(1:length(h1)-2^(k-1)+1);
        hh=conv(h1,hh)*sqrt(2);
    end
    g1=[g;zeros(2^(j-1)-1,length(g))];

```

```

g1=g1(:)';
g1=g1(1:length(g1)-2^(j-1)+1);
gg=conv(g1,hh)*sqrt(2);
h1=[h;zeros(2^(j-1)-1,length(h))];
h1=h1(:)';
h1=h1(1:length(h1)-2^(j-1)+1);
hh=conv(h1,hh)*sqrt(2);
end;
f = hh;
w = gg;

```

函数 result.m: 该函数由主程序调用。其目的是存储计算的结果, 它会打开一个对话窗口, 要求用户提供保存计算结果的文件名字。

```

function save_result()
global Handlist PARAMETER ORIGFILE RATIO RMSD
global MINDATA MAXDATA THRESHCOEFFS
global WAVELETRH WAVELETG LEVELS THRESHOLD

[SpecFile, Path]=uiputfile('*.rst','Save The Results');
set(Handlist(1:6),'visible','off')
set(get(Handlist(3),'children'),'visible','off')
set(get(Handlist(4),'children'),'visible','off')
set(get(Handlist(5),'children'),'visible','off')
set(get(Handlist(6),'children'),'visible','off')
set(Handlist(8),'string','Saving Result Data ...','visible','on');
drawnow
lengthWAVELETRH = length(WAVELETRH);
lengthWAVELETG = length(WAVELETG);
[ROWS, COLUMNS] = size(THRESHCOEFFS);
if (ROWS~=1)&(COLUMNS~=1)
    tempROWS = ROWS;
    tempCOLUMNS = COLUMNS;
    for ii=1:LEVELS
        tempROWS = fix(tempROWS/2);
        tempCOLUMNS = fix(tempCOLUMNS/2);
    end
    Coefficients = THRESHCOEFFS(1:tempROWS, 1:tempCOLUMNS);
    THRESHCOEFFS(1:tempROWS,1:tempCOLUMNS) = ...

```

```
zeros(tempROWS, tempCOLUMNS);
[tempindex1, tempindex2, tempvalues] = find(THRESHCOFFS);
Index_and_Values = [tempindex1'; tempindex2'; tempvalues'];
SpecFile = [Path, SpecFile];
temp=[lengWAVELETREH; WAVELETREH(:); lengWAVELETRG; ...
       WAVELETRG(:); LEVELS; THRESHOLD; ROWS;
       COLUMNS;... MINDATA; MAXDATA;...
       Coefficients(:); Index_and_Values(:)];
eval(['save ', SpecFile, ' temp -ascii']);
else
    ROWS = max(ROWS, COLUMNS);
    COLUMNS = min(ROWS, COLUMNS);
    tempROWS = ROWS;
    for ii=1:LEVELS
        tempROWS = fix(tempROWS/2);
    end
    Coefficients = THRESHCOFFS(1:tempROWS);
    THRESHCOFFS(1:tempROWS) = zeros(tempROWS, 1);
    [tempindex1, tempindex2, tempvalues] = find(THRESHCOFFS);
    Index_and_Values = [tempindex1'; tempvalues'];
    Index_and_Values = Index_and_Values(:);
    SpecFile = [Path, SpecFile];
    temp=[lengWAVELETREH; WAVELETREH(:); lengWAVELETRG; ...
           WAVELETRG(:); LEVELS; THRESHOLD; ROWS;
           COLUMNS;... MINDATA; MAXDATA;...
           Coefficients(:); Index_and_Values(:)];
    eval(['save ', SpecFile, ' temp-ascii']);
end
[m,n] = size(PARAMETER);
set(Handlist(8), 'visible', 'off')
if length(get(Handlist(7), 'children')) > 1
    delete(Handlist(16:22));
end
mes = str2mat(['Original Data:      ', ORIGFILE], ...
              [PARAMETER(2,1:8), '      ', PARAMETER(2,9:n)], ...
              [PARAMETER(3,1:7), '      ', PARAMETER(3,8:n)], ...
              [PARAMETER(4,1:7), '      ', PARAMETER(4,8:n)], ...
              [PARAMETER(5,1:16), '      ', PARAMETER(5,17:n)], ...
```

```
sprintf(' Compression Ratio:      %6.2f',RATIO),...
sprintf(' RMSD:                  %12.10f',RMSD));
txthndlsls = message(Handlist(7), mes);
Handlist(16:(16+length(txthndlsls)-1)) = txthndlsls(:)';
```

函数 message.m: 这是一个辅助函数, 用于向指定的坐标系中写说明文字。输入参数 handle 是某个坐标系的句柄值, messageStr 是字符串, alignment 参数提供说明文字的对齐方式。

```
function [txtHndlList] = message(handle, messageStr, alignment)
% Display message on the axis with handle

if nargin<3
    alignment = 'left';
end;
[rows, cols] = size(messageStr);
top=1;
if strcmp(lower(alignment), 'center')
    left = 0.5;
else
    left=0.05;
end
labelHt=0.08;
spacing=0.005;
shiftspace = (1-rows * labelHt - (rows-1) * spacing)/2;
set(get(handle, 'Children'), 'visible', 'off');
axes(handle);
for count=1:rows
    txtHndlList(count) = text(left, ...
        top-shiftspace-labelHt-(count-1)*(labelHt+spacing), ...
        messageStr(count,:));
    set(txtHndlList(count), 'horizontalAlignment', alignment, ...
        'visible', 'off', 'fontsize', 16, 'fontname', 'courier', ...
        'color', 'y');
end
set(txtHndlList(1:rows), 'visible', 'on')
drawnow
```

函数 file2var.m: 这是一个辅助函数, 其目的是装入一个数据文件。

```
function Data=file2var(Filename)
% Filename: FILE2VAR. M [Matlab Function]
% Author: K. M. Leung
% Date: 14—05—1996
% Version: 1. 0 (Updated at 14—05—1996)
% Usages: To load data from a particular and assign the
%          contents of a data file to a variable.
% Format: Data=file2var(Filename)
%          Data—Content in the file
%          Filename—File name of a particular file

i=find(Filename=='.');
if isunix==1
    j=find(Filename=='/');
else
    j=find(Filename=='\');
end
if length(Filename)<=2
    TempStr=['Error in loading file. '];
    disp(TempStr)
    Data=NaN;
    break;
end
if exist(Filename)==2
    load(Filename,'-ascii');
    if isempty(j)
        Data=eval(Filename(1:(i-1)));
    else
        [m n]=size(j);
        Data=eval(Filename(j(m,n)+1:(i-1)));
    end
else
    TempStr=sprintf('The file %s is not exist.',Filename);
    disp(TempStr)
    Data=NaN;
end
```