MATLAB 应用与提高系列

MATLAB 程序设计

阮沈勇　王永利　桑群芳　编

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书从计算、绘图和编程 3 个方面介绍 MATLAB 的基础知识。全书共分 10 章。第 1~第 4 章为计算部分，主要介绍 MATLAB 的数值计算和符号计算功能。第 5 和第 6 章为绘图部分，主要介绍 MATLAB 的一般图形功能和科学计算可视化能力。第 7~第 10 章为编程部分，从介绍 M 文件开始，讲解 MATLAB 编程的一般特点和语法，然后介绍 MATLAB 中图形用户界面（GUI）设计的方法和技巧。附录部分列出了 MATLAB 的各种功能函数及其含义说明。

本书可供对 MATLAB 感兴趣的大学生、研究生和科研人员阅读和使用。

图书在版编目(CIP)数据

MATLAB 程序设计 / 阮沈勇等编．—北京：电子工业出版社，2004.1
（MATLAB 应用与提高系列）
ISBN 7-5053-9229-8

Ⅰ. M… Ⅱ. 阮… Ⅲ. 计算机辅助计算—软件包，MATLAB Ⅳ. TP391.75

中国版本图书馆 CIP 数据核字（2003）第 091320 号

责任编辑：詹善琼
印　　刷：北京四季青印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
经　　销：各地新华书店
开　　本：787×1 092　1/16　印张：21.50　字数：550.4 千字
印　　次：2004 年 1 月第 1 次印刷
印　　数：5 000 册　　定价：30.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

前　言

许多人很喜欢 MATLAB，觉得它是一个很不错的软件，能够给从事科学计算的人员带来更多的便利和可能性。

MATLAB 好，首先表现在它的不断创新。MATLAB 的每次更新都能给人以惊喜，要么是原有的功能得到扩充或提高，要么是出现新的工具箱或实用工具，要么是整体性能得到改进。DDE、OLE、ActiveX、COM 这样一些流行或曾经流行的标准和技术，在 MATLAB 中都能得到及时的响应。其次，它能满足个性化的需求。MATLAB 提供了几十个工具箱，利用这些工具箱，可以解决不同领域的数学问题。而且，由于 MATLAB 的可扩展性，用户还可以编写自己领域的工具箱，提高工作效率。除了工具箱以外，MATLB 还提供了琳琅满目的实用工具。利用它们，可以实现不同的功能。比如，你用 MATLAB 开发了一套新算法，是 M 文件，但不想让别人看到源代码，想保密，于是考虑做成独立应用程序，用 mcc 来做。如果 mcc 解决不了，就用运行时服务器。如果想把该算法集成到 VB、VC 中去，但不想重写代码，可以用 COM 生成器把对应 M 文件做成 COM 组件，然后集成。所以，只要你需要，总有一款适合你。

MATLAB 是解释型语言，运行速度比较慢。但从 MATLAB 6.5 开始，已比较全面地提速了，提速后的运行速度与向量化后的效果相当。虽然在某些情况下，仍然需要通过循环向量化或预分配数组内存空间等技巧来加速运行，但我们仍然能看到 MATLAB 所做的努力。MATLAB 提供了多种方法来加速运行，通过 Profiler 工具或 profile 函数，可以获取每行代码的运行情况，包括运行时间和调用次数等，因而能知道哪些语句行花费的时间最多，可以集中精力进行改进。

作为一个专业的科学计算软件，MATLAB 的功能首先在于应用，即应用现有函数和工具（箱）解决具体问题。在用的过程中，用户会发现问题，并逐渐有更高的要求。比如想开发自己的算法，想开发速度更快的应用，或者想用 VC、VB 等开发更美观的界面等。所以，用而优则开发，这是很自然的学习追求，也是大多数 MATLAB 学习者要走的路。

整套书共分 3 册，分别偏重于入门、工具箱应用和接口。第一册分计算、绘图和编程 3 部分，介绍 MATLAB 的入门知识和技巧。第二册主要介绍我们所熟悉的统计、优化、偏微分方程数值解、样条、信号处理和曲线拟合等 6 个工具箱的最新版本。第三册介绍 MATLAB 与外部程序的接口，包括 MATLAB 与 FORFILER、C、Visual Basic、Visual C++、Excel，Spss、硬件等接口的技术，其中还介绍 MATLAB 编译器、COM 生成器、Excel 生成器、运行时服务器、报表生成器、Excel Link、ImportWizard、Profiler 等工具的用法。

应该说，除了受专业限制，有一些工具箱没有介绍以外，MATLAB 所提供的大部分功能在这 3 本书中都有不同程度的阐述，只要认真阅读，终会有所收获。当你在学习的过程中，感觉自己一天天变得更加充实，因而内心充满喜悦的时候，我们为你高兴！

关于这本书

本书从计算、绘图和编程 3 个方面介绍 MATLAB 的基础知识，适合于 MATLAB 的初、中级读者。

计算部分主要介绍 MATLAB 的数值计算和符号计算功能。

绘图部分主要介绍 MATLAB 的一般图形功能和科学计算可视化能力。一般图形包括线形图、条形图、散点图、直方图等基本图形，玫瑰花图、阶梯图、火柴杆图、羽列图等功能图形以及多轴图、对数坐标图、半对数坐标图、极坐标图等特殊图形。科学计算可视化已经是一门新的学科，MATLAB 提供了丰富的科学计算可视化能力。使用 MATLAB，可以绘制矢量图、等值线图、网格图、曲面图、云图、剖面图和流域图等多种图形。采用先进的区域填充算法，还能绘制各种具有高度真实感的图形。

编程部分从介绍 M 文件开始，讲解了 MATLAB 编程的一般特点和语法，然后介绍 MATLAB 中图形用户界面（GUI）设计的方法和技巧。值得关注的是，在 MATLAB 中也能实现面向对象编程了，而且能实现构造函数、继承和运算符重载，比 VB 还牛！最后，给出了 MATLAB 操作和编程中的一些小技巧。

附录部分列出了 MATLAB 各种功能函数及其含义说明。

阮沈勇和桑群芳负责编写第 3 章、第 4 章、第 7 章、第 8 章和附录的内容；王永利负责编写第 5 章，其余内容由苏金明编写。王能峰、钟国华等也参与了部分内容的编写。刘玉珊和苏华惠做了大量的录入工作，在此表示衷心感谢！

由于能力有限，书中错误和不足之处在所难免，谨请读者批评指正！有任何问题，请通过电子邮件与我们联系：

阮沈勇 r_shenyong@yahoo.com.cn
王永利 wy_11971@tom.com
桑群芳 sqf_rsy@163.com
苏金明 s_jm@263.net.cn

编　者

2003．6

目　　录

第 1 章　MATLAB 简介

1.1　概述

MATLAB 是由 MathWorks 公司于 1984 年推出的一套数值计算软件，分为总包和若干个工具箱，可以实现数值分析、优化、统计、偏微分方程数值解、自动控制、信号处理、图像处理等若干个领域的计算和图形显示功能。它将不同数学分支的算法以函数的形式分类成库，使用时直接调用这些函数并赋予实际参数就可以解决问题，快速而且准确。

近年来，MATLAB 在国内的知名度越来越大，并已广泛地应用于教学和科研领域。该软件的特点可以归纳为以下几点。

（1）简单易学　MATLAB 是一门编程语言，其语法规则与一般的结构化高级编程语言如 C 语言等大同小异，而且使用更方便，具有一般编程语言基础的用户很快就可以掌握。

（2）代码短小高效　由于 MATLAB 已经将数学问题的具体算法编成了现成的函数，用户只要熟悉算法的特点、使用场合、函数的调用格式和参数意义等，通过调用函数就可以很快解决问题，而不必花大量的时间纠缠于具体的算法。

（3）计算功能非常强大　该软件具有强大的矩阵计算功能，利用一般的符号和函数就可以对矩阵进行加、减、乘、除运算以及转置和求逆运算，而且可以处理稀疏矩阵等特殊的矩阵，非常适合于有限元等大型数值算法的编程。此外，该软件现有的数十个工具箱，可以解决应用中的大多数数学问题。

（4）强大的图形表达功能　该软件不仅可以绘制一般的二维/三维图形，如线图、条形图、饼图、散点图、直方图、误差条图等，还可以绘制工程特性较强的特殊图形，如玫瑰花图、极坐标图等。科学计算要涉及到大量的数据处理，利用图形展示数据场的特征，能显著提高数据处理的效率，提高对数据反馈信息的处理速度和能力。MATLAB 提供了丰富的科学计算可视化功能，利用它，可以绘制二维/三维矢量图、等值线图、三维表面图、假彩色图、曲面图、云图、二维/三维流线图、三维流锥图、流沙图、流带图、流管图、卷曲图、切片图等，此外还可以生成快照图和进行动画制作。

（5）可扩展性能　可扩展性能是该软件的一大优点，用户可以自己编写 M 文件，组成自己的工具箱，方便地解决本领域内常见的计算问题。此外，利用 MATLAB 编译器和运行时服务器，可以生成独立的可执行程序，从而可以隐藏算法并避免依赖 MATLAB。MATLAB 支持 DDE 和 ActiveX 自动化等机制，可以与同样支持该技术的应用程序接口。利用最近推出的 COM 生成器和 Excel 生成器，可以利用给定的 M 文件和/或 MEX 文件创建 COM 组件和 Excel 插件，从而能够实现与 VB、VC 等程序的进程内无缝集成。利用 Web 服务器，可以实现 MATLAB 与网络程序的接口。在目前情况下，采用互操作技术，可以实现 MATLAB 与.NET 程序的接口。利用端口 API 函数，可以实现 MATLAB 与硬件的接口。

1.2 运行环境介绍

1.2.1 MATLAB 的运行方式

MATLAB 提供了两种运行方式，即命令行运行方式和 M 文件运行方式。两种运行方式各有特点，下面分别予以介绍。

1．命令行运行方式

可以通过直接在命令窗口输入命令行来实现计算或作图功能。例如，要计算矩阵 A 和 B 的和，其中

$$A=\begin{bmatrix}2 & 5\\ 6 & 3\end{bmatrix} \qquad B=\begin{bmatrix}-7 & 9\\ -2 & 0\end{bmatrix}$$

首先打开 MATLAB 界面，如图 1-1 所示。

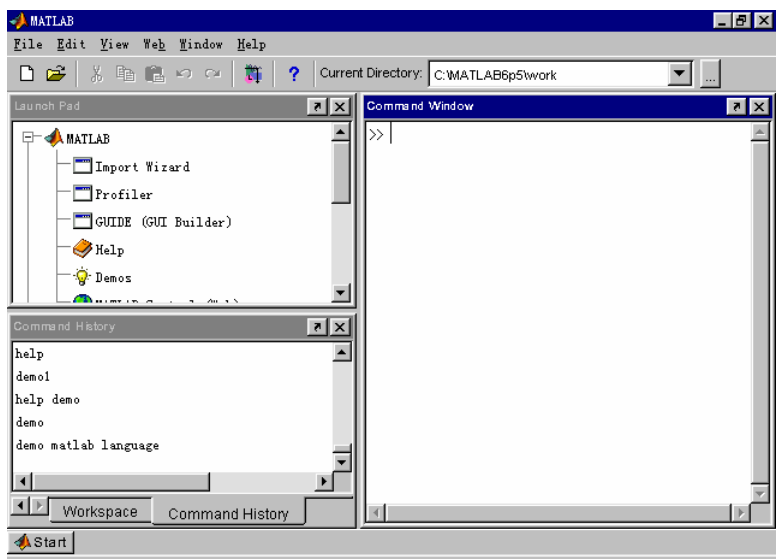


图 1-1 MATLAB 界面

在命令窗口输入下面的命令行：

```
A=[2 5;6 3];
B=[-7 9;-2 0];
C=A+B
C =
    -5    14
     4     3
```

2．M 文件运行方式

在 MATLAB 窗口中单击“File”菜单，然后依次选择 New→M-file 选项，打开 M 文件窗口（输入运行界面），如图 1-2 所示。在该窗口中输入程序文件，可以进行调试和运行。与命令行运行方式相比，M 文件运行方式的优点是可调试，可重复应用。

对于前面的矩阵求和问题，可在 M 文件窗口中输入程序，如图 1-2 所示。然后在“Debug”

菜单中选择“Run”选项，将在命令窗口输出矩阵 C=A+B 的值。

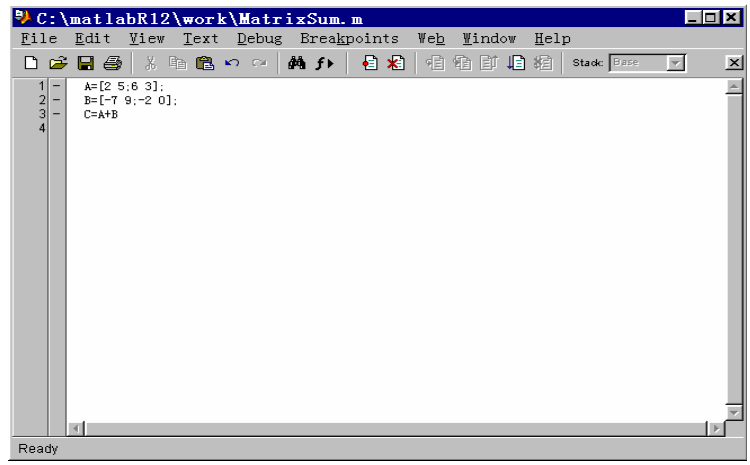


图 1-2　M 文件输入运行界面

1.2.2　MATLAB 中的窗口

在 MATLAB 中，常见的窗口有命令窗口、M 文件窗口、起始面板窗口、工作空间窗口、命令历史窗口、当前目录窗口和图形窗口等。

1．命令窗口

命令窗口如图 1-1 中右侧所示窗口。在该窗口中可以输入命令行，实现计算或绘图功能。命令窗口中有一些常用的功能键，利用它们可以使操作更简便快捷。常见的功能键如表 1-1 所示。

表 1-1　命令窗口中常用的功能键

功　能　键	功　　能	功　能　键	功　　能
↑，Ctrl-P	重新调入上一命令行	Home,Ctrl-A	光标移到行首
↓，Ctrl-N	重新调入下一命令行	End,Ctrl-E	光标移到行尾
←，Ctrl-B	光标左移一个字符	Esc	清除命令行
→，Ctrl-F	光标右移一个字符	Del,Ctrl-D	删除光标处字符
Ctrl-←	光标左移一个字	Backspace	删除光标左边字符
Ctrl-→	光标右移一个字	Ctrl-K	删除至行尾

2．M 文件窗口

M 文件窗口如图 1-2 所示。在该窗口中，可以输入、编辑、调试和运行 M 文件（有关 M 文件的编制参见第 7 章的内容）。利用“Edit”菜单中的选项，可以对 M 文件进行编辑。利用“Debug”和“Breakpoints”菜单中的选项，可以进行调试。利用“Breakpoints”菜单中的选项，可以设置和取消断点。利用“Debug”菜单中的选项，可以确定运行方式，如逐行运行、运行至光标处等等。单击“Run”选项，运行 M 文件。

3．起始面板窗口

起始面板窗口如图 1-3 所示。该窗口中显示 MATLAB 总包和已安装的工具箱的帮助、演示、GUI 工具、实用工具和产品主页等几个方面的内容。如果用户希望查看总包或某个工具

箱的上述几个方面的内容，则在起始面板窗口中双击对应的图标就可以很快得到相关信息。

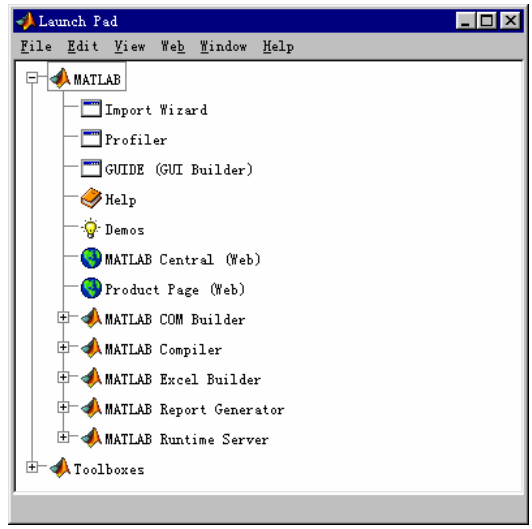


图 1-3　起始面板窗口

4．工作空间窗口

工作空间窗口中列出数据的变量信息，包括变量名、变量数组大小、变量字节大小和变量类型。

在命令窗口中输入命令行：

load cities

load wind

则工作空间窗口中将显示这两个数据系统的变量信息，如图 1-4 所示。

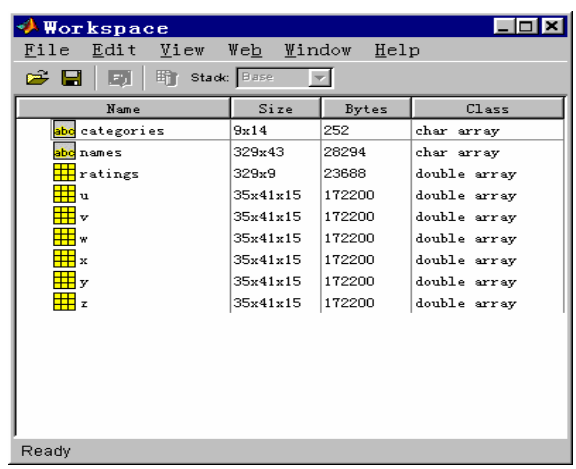
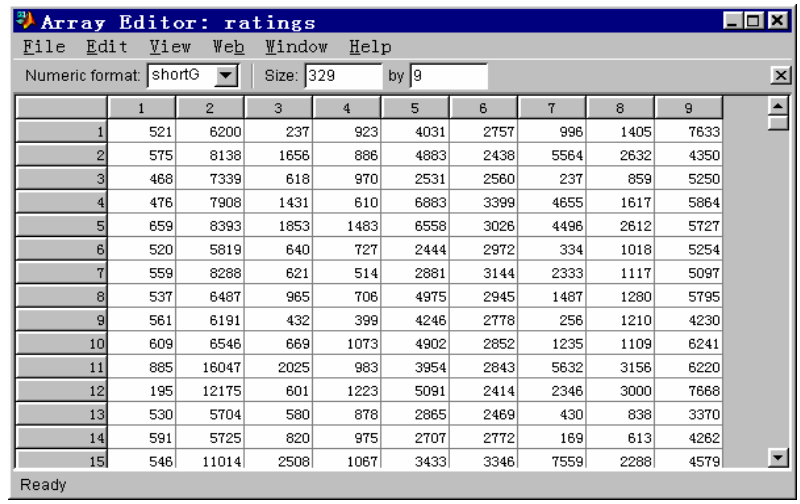


图 1-4　工作空间窗口

图 1-4 中，“ Name ”列、“ Size ”列、“ Bytes ”列和“ Class ”列分别对应变量名、变量数组大小、变量字节大小和变量类型。变量名前面的图标表示对应的变量类型。如第一个变量的变量名为 categories，为 9×14 的矩阵，字节大小为 252，变量类型为字符型。

在工作空间窗口中选择某个变量以后，图标和变为可用。单击图标，将打开数

组编辑器，显示该变量的具体内容。该显示主要应用于数值型变量。例如，选中 ratings 变量以后，单击图标，将显示图 1-5 所示的数组编辑器。图中，ratings 变量的行和列上的数据均显示在电子表格中了，用户可以进行编辑。选中变量以后，单击图标，将删除该变量的信息。



	1	2	3	4	5	6	7	8	9
1	521	6200	237	923	4031	2757	996	1405	7633
2	575	8138	1656	886	4883	2438	5564	2632	4350
3	468	7339	618	970	2531	2560	237	859	5250
4	476	7908	1431	610	6883	3399	4655	1617	5864
5	659	8393	1853	1483	6558	3026	4496	2612	5727
6	520	5819	640	727	2444	2972	334	1018	5254
7	559	8288	621	514	2881	3144	2333	1117	5097
8	537	6487	965	706	4975	2945	1487	1280	5795
9	561	6191	432	399	4246	2778	256	1210	4230
10	609	6546	669	1073	4902	2852	1235	1109	6241
11	885	16047	2025	983	3954	2843	5632	3156	6220
12	195	12175	601	1223	5091	2414	2346	3000	7668
13	530	5704	580	878	2865	2469	430	838	3370
14	591	5725	820	975	2707	2772	169	613	4262
15	546	11014	2508	1067	3433	3346	7559	2288	4579

图 1-5　数组编辑器

5．命令历史窗口

命令历史窗口如图 1-6 所示，它显示命令窗口中所有执行过的命令。利用该窗口，一方面可以查看曾经执行过的命令，另一方面，可以重复利用原来输入的命令行。可以从命令历史窗口中直接通过双击某个命令行来执行该命令行，也可以通过拖曳或复制操作将命令行复制到命令行窗口后再回车执行。

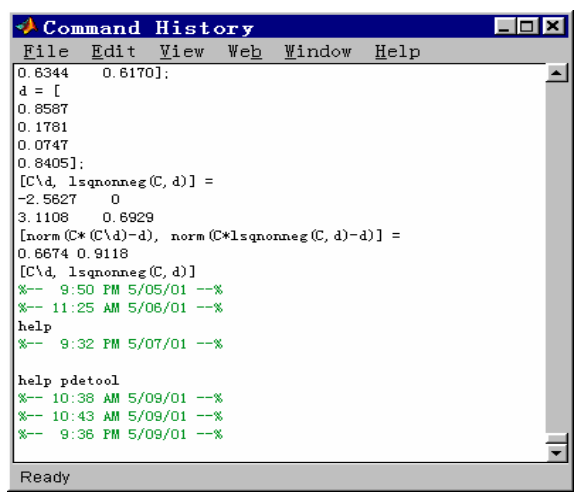


图 1-6　命令历史窗口

6．当前目录窗口

当前目录窗口如图 1-7 所示，该窗口中显示当前目录下所有文件的文件名、文件类型和最后修改时间。

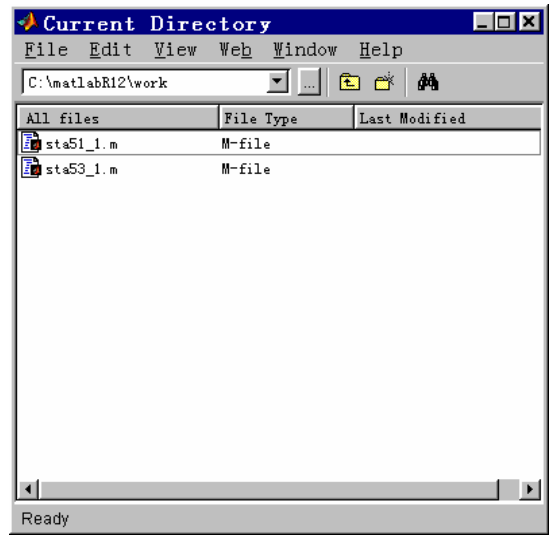


图 1-7　当前目录窗口

7．**图形窗口**

在“ File **”菜单下的“** New **”次级菜单中选择“** Figure **”选项或在命令窗口中输入** figure **或执行其他绘图命令，将打开图形窗口，如图** 1-8 **所示。**

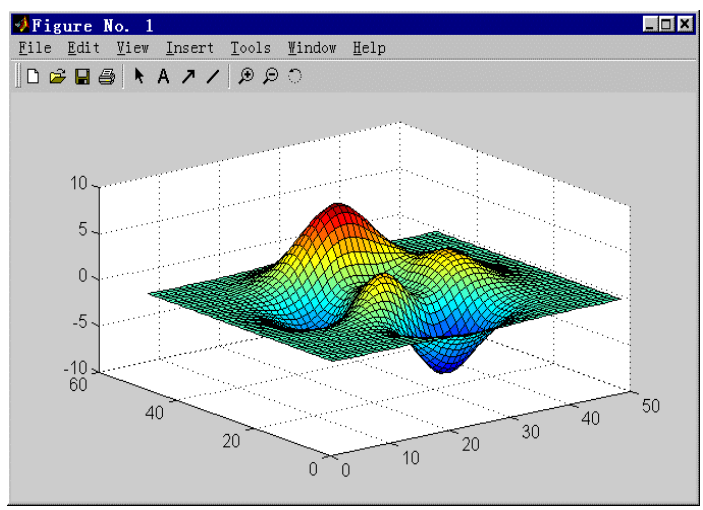


图 1-8　图形窗口

图 1-8 **是在命令窗口中输入下面的命令行后生成的。**

```
surf(peaks)
```

利用图形窗口菜单和工具栏中的选项，可以对图形进行线型、颜色、标记、三维视图、光照、坐标轴等内容的设置。

8．GUI **制作窗口**

在“ File **”菜单下选择“** New **”次级菜单中的“** GUI **”选项，可打开图形用户界面制作窗口，如图** 1-9 **所示。**MATLAB **给出了** 4 **种制作图形用户界面的模板。利用它们，用户可以快速创建自己的图形界面或对已有界面进行编辑（关于图形用户界面制作的详细内容，请参见**

第 8 章）。

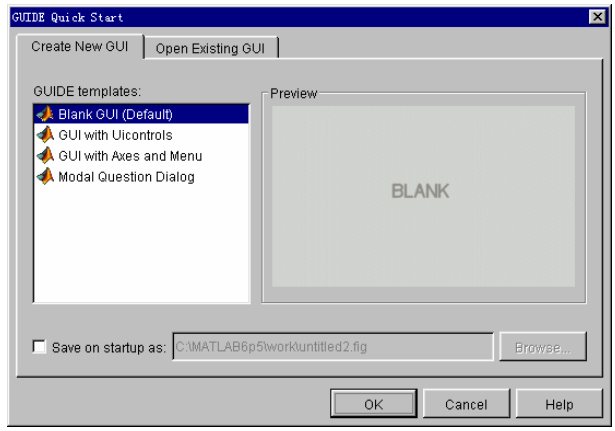


图 1-9　图形用户界面（GUI）制作窗口

1.3　MATLAB 的帮助系统

MATLAB 具有完善的帮助系统，包括命令行帮助、联机帮助和演示帮助等。充分利用 MATLAB 的帮助系统，用户可以更快更准确地掌握其使用方法。

1.3.1　命令行帮助

利用 Help 命令可以获得命令行帮助。例如，在命令窗口输入命令

help

将显示如图 1-10 所示的帮助信息。图中列出了所有函数类别和工具箱的名称和功能。

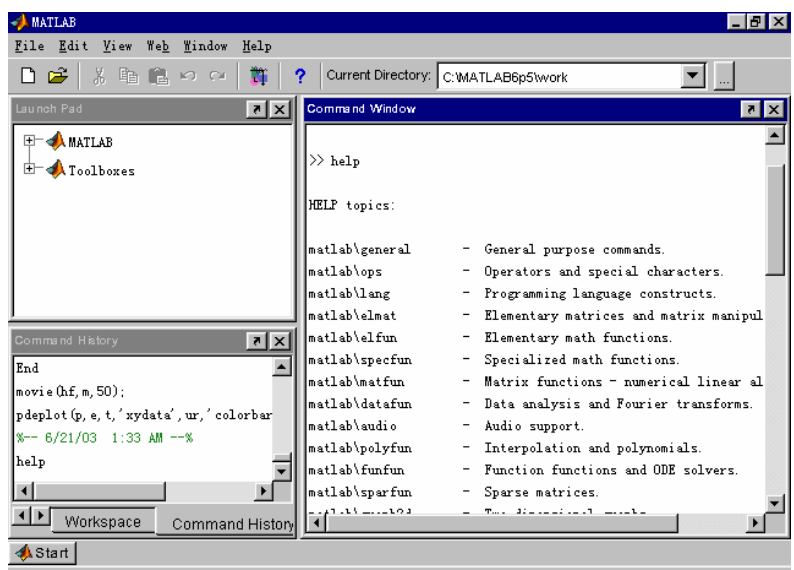


图 1-10　命令行帮助信息

在 help 命令后面添加工具箱名或命令名，可以显示对应的功能信息。例如，在命令行中

键入

help stats

将获得统计工具箱中各种类别函数的名称和功能说明。

1.3.2 联机帮助

在 MATLAB 界面中单击工具条上的问号按钮，或单击“ Help ”菜单中的“ MATLAB Help ”选项，可以打开联机帮助界面，如图 1-11 所示。在界面左边的目录栏中单击项目名称或图标，将在右侧的窗口中显示对应的帮助信息。

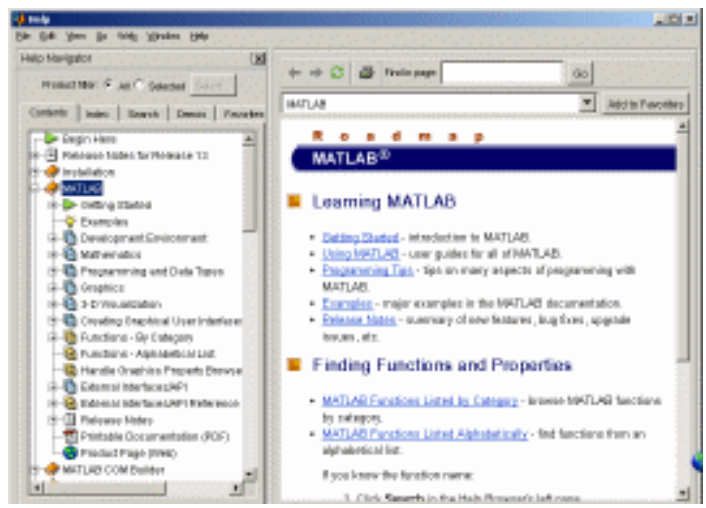


图 1-11　联机帮助界面

1.3.3 演示帮助

单击“ Help ”菜单中的“ Demos ”选项，可以打开演示帮助说明窗口，如图 1-12 所示。初学者可以从 Demo 演示开始学习。窗口中给出了 MATLAB 各项功能的演示方法。

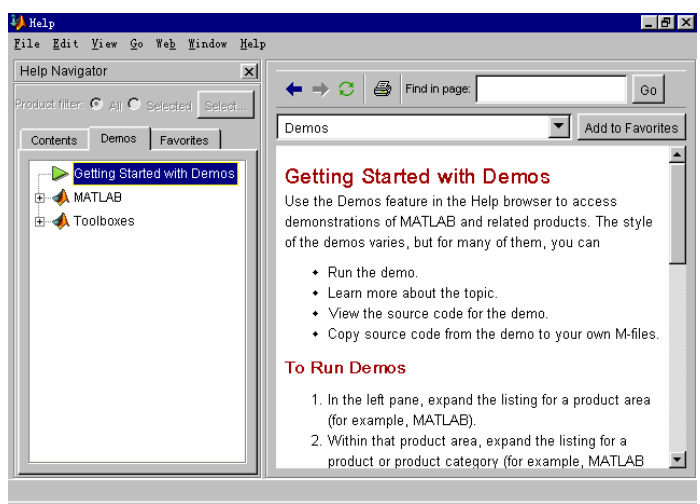


图 1-12　演示帮助说明窗口

第 2 章　数 据 类 型

2.1　概述

2.1.1　MATLAB 数组

MATLAB 语言中只有一种对象类型：MATLAB 数组。所有的 MATLAB 变量，包括标量、向量、矩阵、字符串、单元数组、结构和对象等都以数组形式保存。

所有 MATALB 数据都按列存放，这与 FORTRAN 中相似。因为 MATLAB 最初是用 FORTRAN 编写的。例如，对于下面的字符串数组 a：

```
a=['house'; 'china';'tiger']
a=
   house
   china
   tiger
```

用 size 函数查看数组 a 的大小：

```
size(a)
ans=
     3     5
```

即，a 是一个 3 行 5 列的数组。

a 数组的保存顺序为：

```
h c t o h i u i g s n e e a r
```

2.1.2　MATLAB 中的数据类型

MATLAB 提供的数据类型有复数双精度矩阵、数值矩阵、MATLAB 字符串、稀疏矩阵、单元数组、结构、对象等。下面先做一简略介绍。

1．复数双精度矩阵

MATLAB 中最常见的数据类型是复数双精度的非稀疏矩阵。这些矩阵的元素是 Double 型的，大小为 $m \times n$，其中，m 为行数，n 为列数。此类数据保存为两个值为双精度类型的向量：一个保存实部数据，另一个保存虚部数据。指向数据的指针分别引用为 pr（指向实部数据）和 pi（指向虚部数据）。

2．数值矩阵

MATLAB 还提供了其他的数值矩阵类型，包括单精度浮点型、8 位、16 位和 32 位整型等。它们还是像复数双精度矩阵那样保存为两个向量。

3．MATLAB 字符串

MATLAB 字符串为 char 类型，保存为正 16 位整型，没有虚部。

4．稀疏矩阵

MATLAB 中，稀疏矩阵的存储方式与满秩矩阵的存储方式不同。除了有 pr 和 pi 两个参

数以外，它还有另外 3 个参数：nzmax，ir 和 jc。nzmax 是一个整型值，包含 ir，pr 和 pi 的长度，是稀疏矩阵中非零元素的最大可能个数。ir 指向一个长度为 nzmax 的整型数组，包含 pr 和 pi 中对应元素的行号。jc 指向一个长度为 N+1 的整型数组，包含列号信息。

5．单元数组

单元数组是一个 MATLAB 数组的集合，其中每个 mxArray 作为单元引用。这样，允许不同类型的 MATLAB 数组保存在一起。

6．结构

一个 1×1 的结构的保存方式与一个 1×n 的单元数组的保存方式相同。其中，n 为结构中的字段个数。数据向量的成员称为字段，每个字段与 mxArray 中的名称相关联。

7．对象

对象的保存方式和获取数据的方式与结构的相同。MATLAB 中，对象是带有方法的命名结构（在第 9 章中将详细介绍 MATLAB 中的类和对象）。

8．多维数组

任何类型的 MATLAB 数组都可以是多维的。这里保存了一个整型向量，它的每个元素的值对应维的大小。数据的保存与矩阵相同。

9．空数组

任何类型的 MATLAB 数组都可以是空的。一个空的 mxArray 是一个至少有一维等于 0 的数组。

2.2 字符数组

2.2.1 创建字符数值

通过把字符放到单引号中来指定字符数据。例如，下面创建一个 1×5 的字符数组 country。

```
country = 'China';
```

在命令窗口输入 whos 命令，输出显示：

Name	Size	Bytes	Class
country	1×5	10	char array

从上面显示中可以看出，在内存中，每个字符占用 2 个字节。下面使用 class 函数和 ischar 函数测试 country 的数据类型，结果显示它是一个字符数组。

```
class(country)
ans =
    char
ischar(country)
ans =
     1
```

用 strcat 函数可把两个或更多个字符数组组合在一起。例如：

```
country = 'China';
province= 'SiChuan';
strcat(country,',',province)
ans =
```

China,SiChuan

2.2.2 创建二维字符数组

创建一个二维字符数组时，确定每行具有相同的长度。例如，下面两个字符串都有 5 个字符，用它们组合起来的二维字符数组是合法的。

```
name = ['Li    Yi'; 'Hu Xu']
name =
    Li    Yi
    Hu Xu
```

根据不同长度的字符串创建字符数组时，在短的那些字符串后面用空格补齐，使所有字符串的长度相同。例如：

```
name = ['Liu Ying'; 'Hu Xu      '];
```

用 char 函数创建字符串数组更简单，char 函数自动以输入的最长的字符串的长度为标准，进行空格补齐工作。例如：

```
name = char('Liu Ying','Hu Xu')
name =
    Liu Ying
    Hu Xu
```

从数组中提取字符串时，用 deblank 函数删除后面的空格。

```
trimname = deblank(name(2,:))
trimname =
    Hu Xu
size(trimname)
ans =
        1     5
```

字符数组把每个字符作为 16 位数值值保存，使用 double 函数把字符串转换为数值值，使用 char 函数把数值值转换为字符串。

```
name = double(name)
name =
        76     105     117      32      89     105     110     103
        72     117      32      88     117      32      32      32
name = char(name)
name =
        Liu Ying
        Hu Xu
```

使用 str2num 函数把字符数组转换为该字符串表示的数值值。

```
str = '102.724e-1';
val = str2num(str)
val =
        10.2724
```

2.2.3 字符串单元数组

把多组字符串保存为单元数组的形式，可以避免在短字符串后面频繁添加空格。利用 MATLAB 提供的一系列函数，可以针对字符串单元数组进行两种操作：一种是在标准字符数组与字符串单元数组之间进行转换，另一种是对字符串单元数组进行字符串比较操作。

cellstr 函数把字符数组转换到一个字符串单元数组中，对于字符数组 data = ['Sun junfang';'NewStudent';'Beijing ']; 该矩阵通过空格补齐使各行的长度相同。

现在使用 cellstr 函数创建单元的列向量，每个单元包含 data 数组中的一个字符串。

```
celldata = cellstr(data)
celldata =
    'Sun junfang'
    'NewStudent'
    'Beijing'
```

注意：cellstr 函数把各行中为了补齐所添加的空格去掉了。例如：

```
length(celldata{3})
ans =
     7
```

iscellstr 函数确定输入变量是否为字符串单元数组。它返回一个逻辑值 true(1)或 false(0)。

```
iscellstr(celldata)
ans =
     1
```

2.2.4 类型转换

用 char 函数把字符串单元数组转换为标准字符串数组。

```
strings = char(celldata)
strings =
    Sun junfang
    NewStudent
    Beijing
length(strings(3,:))
ans =
    10
```

str2double 函数把一个字符串单元数组转换为字符串表示的双精度值。

```
c = {'37.294e-1'; '-58.375'; '13.796'};
d = str2double(c)
d =
     3.7294
   -58.3750
    13.7960
whos
  Name          Size                Bytes                Class
```

```
c              3×1              224              cell array
d              3×1               24              double array
Grand total is 28 elements using 248 bytes
```

int2str **函数将整型数据转换为字符串型数据。例如：**

```
x=2003;
y = int2str(x);
size(y)
ans =
     1     4
```

可见，int2str **函数将标量** x **转换成了一个包含字符串“** 2003 **”的** 1×4 **的向量。**

num2str **函数对输出字符串的格式提供了更多控制，该函数的第** 2 **个变量是可选的，它设置输出字符串的位数，或指定一个实际的格式。例如：**

```
p = num2str(pi,6)
p =
    3.14159
```

int2str **和** num2str **都可以用于在图中进行标注。例如，下面的命令行使用** num2str **函数自动标注** *x* **轴。**

```
function plotlabel(x,y)
        plot(x,y)
        str1 = num2str(min(x));
        str2 = num2str(max(x));
        out = ['Value of f from ' str1 ' to ' str2];
        xlabel(out);
```

mat2str **函数将数组改变为一个可以评价的字符串。例如，创建一个** 2×3 **的数组** A。

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

mat2str **函数将返回一个包含文本的字符串，该文本在命令行中输入，包含矩阵** A **的元素。**

```
B = mat2str(A)
B =
    [1 2 3; 4 5 6]
```

2.2.5 字符串比较

用 strcmp **函数可以比较字符串。**

```
str1 = 'hello';
str2 = 'help';
strcmp(str1,str2)
C =
    0
```

因为 str1 **和** str2 **不相等，所以调用** strcmp **函数时返回** 0(false).

使用 strncmp 函数，可以比较字符串中的前 *n*+1 个字符。例如：

```
C = strncmp(str1,str2,2)
C =
     1
```

因为“ hello ”和“ help ”的前 3 个字符相同，所以调用 strncmp 函数时返回 1。

对于字符串结构数组，这些函数将它们逐单元地进行比较。

```
A = {'pen';'pen';'rule'};
B = {'pencil';'pen';'pencilbox'};
```

下面用 strcmp 函数和 strncmp 函数对两个字符数组进行比较：

```
strcmp(A,B)
ans =
     0
     1
     0
strncmp(A,B,1)
ans =
     1
     1
     0
```

可见，使用 strcmp 函数进行比较时，由于 A 和 B 中只有第 2 个元素完全相同，所以返回值为 1，其他两个均为 0。使用 strncmp 函数进行比较时，由于 A 和 B 中的第 1 个元素的前两个字符相同，所以返回值为 1。

对于字符数组，可以使用 MATLAB 的关系操作符。例如，可以用（＝＝）确定两个字符串中的相同字符。

```
A = 'face';
B = 'cake';
A == B
ans =
     0     1     0     1
```

所有关系操作符(>, >=, <, <=, ==, !=)都可用来比较对应字符的值。

2.2.6 字符分类

有两个函数可以对字符串中的字符进行分类。

- isletter 函数确定字符是否是字母。
- isspace 函数确定字符是否为空区（空格、空表间隔或空行）。

例如，下面创建一个名为 mystring 的字符串，使用 isletter 函数检查字符串中哪些字符为字母。

```
mystring = 'Room 401';
A = isletter(mystring)
A =
     1     1     1     1     0     0     0     0
```

前 4 个字符是字母，所以返回值为 1。

2.2.7 搜索和替换

MATLAB 提供了几个函数用于字符串的搜索和替换。例如，对于字符串 label：

```
label = 'Sample 1, 04/28/03';
```

strrep 函数进行搜索和替代操作，用 strrep 函数把“04/28”改为“04/30”。

```
newlabel = strrep(label,'28','30')
newlabel =
    Sample 1, 04/30/03
```

findstr 函数把子字符串的起始位置返回到一个更长的字符串中。

```
position = findstr('amp',label)
position =
        2
```

字符串'amp'在 label 字符串中出现的起始位置为 label 字符串中第 2 个字符的位置。

strtok 函数在输入字符串中第一次发现间隔符时返回间隔符前面的字符。可用该函数把句子分离成单词。例如：

```
function all_words = words(input_string)
        remainder = input_string;
        all_words = '';
        while (any(remainder))
          [chopped,remainder] = strtok(remainder);
          all_words = strvcat(all_words,chopped);
        end
```

strmatch 函数在字符数组或字符串单元数组的整个行中进行查找，看有没有以给定字符序列打头的字符串，它返回以该字符串打头的行的行号。例如：

```
maxstrings = strvcat('max','minimax','maximum')
maxstrings =
    max
    minimax
    maximum
strmatch('max',maxstrings)
ans =
      1
      3
```

因为 maxstrings 字符串单元数组中第 1 行和第 3 行的字符串以 max 打头，所以用 strmatch 函数得到的返回值为 1 和 3。

2.3 多维数组

2.3.1 创建多维数组

可以使用与创建二维矩阵相同的技巧创建多维数组。另外，MATLAB 还提供了一个特别

的聚合函数来生成多维数组。

- **用索引生成数组**　　创建多维数组的方法之一是先创建一个二维数组，然后扩展它。例如：

```
A = [5 7 8; 0 1 9; 4 3 6];
```

A 是一个 3×3 的数组，即它的行维和列维都是 3。给 A 添加第三维。

```
A(:,:,2) = [1 0 4; 3 5 6; 9 8 7]
```

可通过增大多维数组的索引值来扩展多维数组。例如：

```
A(:,:,3) = 5;
A(:,:,3)
ans =
     5     5     5
     5     5     5
     5     5     5
```

下面把 A 扩展成一个 3×3×3×2 的四维数组。输入：

```
A(:,:,1,2) = [1 2 3; 4 5 6; 7 8 9];
A(:,:,2,2) = [9 8 7; 6 5 4; 3 2 1];
A(:,:,3,2) = [1 0 1; 1 1 0; 0 1 1];
```

- **使用 MATLAB 函数生成数组**　　可以使用与生成二维数组相同的方式，用 randn，ones 和 zeros 等函数生成多维数组。提供的每个变量表示生成的数组中对应维的大小。例如：要创建一个正态分布随机数的 4×3×2 的数组，可以输入：

```
B = randn(4,3,2)
```

要生成每一个元素的值均为同一常数的数组，使用 repmat 函数。例如：

```
B = repmat(5,[3 4 2])
B(:,:,1) =
     5     5     5     5
     5     5     5     5
     5     5     5     5
B(:,:,2) =
     5     5     5     5
     5     5     5     5
     5     5     5     5
```

注意：通过将数组的任何维设置为空数组的形式，使该维的大小为 0。例如，10×0×20 是一个合法的大小。

- **用 cat 函数生成多维数组**　　cat 函数是创建多维数组的一种简单方式，它按指定的维数将多个数组聚合到一起，其调用格式为：

```
B = cat(dim,A1,A2…)
```

其中，A1，A2 等是进行聚合的数组，dim 是维数。例如，要用 cat 函数创建一个新的数组 B：

```
B = cat(3,[2 8; 0 5],[1 3; 7 9])
B(:,:,1) =
     2     8
```

```
     0     5
B(:,:,2) =
     1     3
     7     9
```

cat 函数接收任何已经存在的数据和新数据。另外，可以嵌套调用 cat 函数。下面的命令行创建一个四维数组。

```
A = cat(3,[9 2; 6 5],[7 1; 8 4])
B = cat(3,[3 5; 0 1],[5 6; 2 1])
D = cat(4,A,B,cat(3,[1 2; 3 4],[4 3;2 1]))
```

2.3.2 多维单元数组

与数值数组相似，多维单元数组是二维单元数组模型的扩展。可以用 cat 函数生成多维单元数组。例如，创建一个简单的三维单元数组 C：

```
A{1,1} = [1 2;4 5];
A{1,2} = 'Name';
A{2,1} = 2-4i;
A{2,2} = 7;
B{1,1} = 'Name2';
B{1,2} = 3;
B{2,1} = 0:1:3;
B{2,2} = [4 5]';
C = cat(3,A,B);
```

2.4 结构

结构是调用字段的 MATLAB 数组。结构的字段能包含任何类型的数据。例如，一个字段可能包含一个表示名称的文本字符串，另一个可能包含一个表示类别的标量，第三个则为测量结果矩阵，等等。

2.4.1 创建结构数组

有两种方式可生成结构数组：一种是使用赋值语句，另一种是使用 struct 函数，下面分别予以介绍。

1．使用赋值语句

可以通过将数据赋给单独的字段来生成一个简单的 1×1 结构数组。

创建一个 student 结构数组。

```
student.name = 'Lu dan';
student.ID = 02;
student.test = [79 75 73; 80 78 77.5; 80 85 85];
```

在命令行中输入

```
student
```

结果显示：

```
student =
    name: 'Lu dan'
    ID: 2
    test: [3x3 double]
```

student 是一个包含有 3 个字段的结构数组。要扩展该结构数组，在结构名后面添加索引号。例如：

```
student(2).name = 'Han xu';
student(2).ID= 10;
student(2).test = [68 70 68; 78 88 81; 92 90 93];
```

现在，student 结构数组的大小为[1,2]。

注意：一旦结构数组包含一个以上的元素，键入数组名时 MATLAB 不显示单独的字段内容，而是显示结构包含的信息类别的一个综述列表。

1×2 结构数组的字段为：

```
student =
1×2 struct array with fields:
    name
    ID
    test
```

还可以使用 fieldnames 函数获取这个信息。该函数返回一个包含字段名称的单元数组，扩展结构时，MATLAB 用空矩阵填充，没有指定的字段，这样，数组中的所有结构具有相同的字段名。例如，输入

```
student(3).name = 'Zhang ling'
```

将 student 数组大小扩展为[1,3]。现在，student(3).ID 和 student(3).test 都包含空矩阵了。

注意：不要求数组中每个元素的字段大小相同。在 student 结构中，name 字段可以有不同的长度，test 字段可以是大小任意的数组，等等。

2．使用 struct 函数

可以用 struct 函数预分配一个结构数组。其基本形式为：

```
str_array = struct('field1',val1,'field2',val2,…)
```

其中，变量为字段名和它们的对应值。

2.4.2 在结构数组中获取数据

使用结构数组索引，能获取结构数组中的任何字段值或字段元素。相似地，可以给任何字段或字段单元赋值。在结构数组名后面添加索引范围，可以获取子数组。例如，下面的命令行生成一个 1×2 的结构数组。

```
mystudents = student(1:2)
1x2 struct array with fields:
    name
    ID
    test
```

mystudents 数组中的第一个结构与 student 数组中的第一个结构相同。

```
mystudents(1)
```

```
ans =
  name: 'Lu dan'
        ID: 2
      test: [3x3 double]
```

要获取一个特定结构的字段，在结构名后面跟一个小数点，小数点后面再跟字段名。即：

```
str = student(2).name
str =
    Han xu
```

可以赋值，例如：

```
student(3).test(2,2) = 87;
```

可以为结构一次提取一个字段值。例如，下面的命令行创建一个 1×3 的包含所有 ID 字段的向量。

```
IDs = [student.ID]
IDs =
     2    10
```

类似地，可以为前两个结构创建一个包含测试数据的单元数组。

```
tests = {student(1:2).test}
tests =
    [3x3 double]    [3x3 double]
```

2.4.3 结构数组的大小

使用 size 函数可以获取结构数组或任何结构字段的大小。给定一个结构数组名作为变量，size 返回一个数组维向量。给定 array(n).field 形式的变量，size 函数返回一个包含字段内容大小的向量。例如：对于 1×3 的结构数组 student，size(student)返回向量[1,3]。语句 size(student(1,2),name)返回 student 的元素（1，2）的名称字符串的长度。

2.4.4 操作字段

可以通过给单一结构添加字段来给数组中的每个结构添加一个字段。例如，要给 student 数组添加一个 gender 字段。

下面使用赋值语句给结构添加字段。

```
student(2).gender = 'boy';
```

现在 student(2).gender 具有指定的值。数组中的每个其他结构也有 gender 字段，但这些字段在给它们赋值以前包含空矩阵。

可以用 rmfield 函数从结构数组中的每个结构中删除给定的字段。它的基本形式为 struc2 = rmfield(array, 'field')，其中，array 是一个结构数组，'field'是要删除的字段的名称。例如，要从 student 数组中删除 name 字段，输入

```
student = rmfield(student,'name');
```

可以像操作其他 MATLAB 函数一样操作字段和字段元素。使用索引获取要操作的元素。例如，计算 student(2)中 test 数组的行的均值：

```
mean((student(2).test));
```

有时候，有多种方法对结构数组中的所有字段使用函数和操作符。求 student 数组中所有 ID 字段的和的方式是：

```
total = 0;
for k = 1:length(student)
    total = total + student(k).ID;
end
```

可用下面的语句简化上面的操作：

```
total = sum ([student.ID]);
```

这等于逗号间隔的列表：

```
total = sum ([student(1).ID, student(2).ID…]);
```

2.4.5 结构嵌套

一个结构字段能包含另一个结构，甚至一个结构数组。一旦创建了一个结构，就可以使用 struct 函数或赋值语句，在已经存在的结构字段中嵌套结构。

1．用 struct 函数创建嵌套结构

要创建嵌套的结构，可以嵌套调用 struct 函数。例如，创建下面的结构数组：

```
A = struct('data',[3 4 7; 8 0 1],'nest',…
    struct('testnum','Test 1', 'xdata',[4 2 8],…
    'ydata',[7 1 6]));
```

可以通过赋值语句生成嵌套的结构。这些语句给数组中添加第二元素。例如：

```
A(2).data = [9 3 2; 7 6 5];
A(2).nest.testnum = 'Test 2';
A(2).nest.xdata = [3 4 2];
A(2).nest.ydata = [5 0 9];
```

2．索引嵌套的结构

要索引嵌套的结构，用点(.)标记添加嵌套的字段名。索引表达式中的第一个文本字符串确认结构数组，后面的表达式获取包含其他结构的字段名。例如，前面创建的数组 A 就有 3 级嵌套。用 A(1).nest 获取 A(1)中的嵌套结构。用 A(2).nest.xdata 获取 A(2)嵌套结构的 xdata 字段，用 A(1).nest.ydata(2)获取 A(1)中 ydata 字段的第二个元素。

2.5 单元数组

单元数组是一种特殊的数组，它的元素是单元，能包含其他 MATLAB 数组。例如，单元数组的一个单元可能包含一个实型矩阵，另一个单元又包含文本字符串数组，第三个单元又包含值为复数的向量。可以生成任何大小和形状的单元数组。结构数组和单元数组都为不同类型的数据提供了一种系统存储的机制。它们主要在组织数据的方式上有所不同。在结构数组中，从命名字段获取数据，在单元数组中，通过矩阵索引操作获取数据。

2.5.1 创建单元数组

可以通过下面的方式创建单元数组。

1．通过赋值语句创建单元数组

通过给单个单元赋值来生成单元数组。每次给一个单元赋值。MATLAB 根据赋值的情况，自动生成一定大小的数组。给单元赋值有两种方式。

- 单元索引：按标准数组方式把单元索引号放到小括号中间，把单元的内容放在赋值语句的右端，用大括号括起来。例如，下面创建一个 2×2 的单元数组 A：

```
A(1,1) = {[1 4 3; 0 5 8; 7 2 9]};
A(1,2) = {'Liu zhong'};
A(2,1) = {3+7i};
A(2,2) = {-pi:pi/10:pi};
```

注意：“{　}”表示一个空的单元数组，就像“[　]”表示空的数值数组矩阵一样。可以在任何单元数组赋值时使用空的单元数组。

- 内容索引：以标准数组标注方式把单元索引号用大括号括起来放在赋值语句的左侧，在赋值语句的右侧指定单元内容。

```
A{1,1} = [1 4 3; 0 5 8; 7 2 9];
A{1,2} = 'Liu zhong';
A{2,1} = 3+7i;
A{2,2} = -pi:pi/10:pi;
```

注意：如果已经有一个给定名称的数值数组，不要在没有清除数值数组的情况下试图创建一个同名的单元数组。如果没有清除数值数组，MATLAB 认为你混淆了单元和数值的语法，会生成一个错误。MATLAB 以压缩的形式显示单元数组。

```
A =
    [3x3 double]                'Liu zhong'
    [3.0000+ 7.0000i]           [1x21 double]
```

要显示单元的全部内容，使用 celldisp 函数。对于单元结构的高级图形显示，使用 cellplot 函数。如果赋值给一个维数大于当前数组的维数的单元，MATLAB 自动扩展数组，以包含指定的索引号，它用空矩阵填充所有间隔的单元。例如，下面的赋值将一个 2×2 的单元数组 A 放到一个 3×3 的单元数组中：

```
A(3,3) = {5};
```

2．用 cell 函数预分配单元数组

使用 cell 函数，可以预分配指定大小的空单元数组。例如，下面的语句创建一个空的单元数组：B = cell(2,3);

用赋值语句填充 B 单元：B(1,3) = {1:3};

2.5.2　从单元数组中获取数据

可以从单元数组中获取数据，或以标准数组或单元数组的形式保存结果。本节讨论：

- 用内容索引获取单元内容。
- 用单元索引获取单元子集。

1．用内容索引获取单元内容

可以用语句右端的内容索引获取单个单元中间的某些或全部数据。在赋值语句的左侧指定一个变量，获取单元中的内容。用大括号把语句右端的单元索引表达式括起来。对于下面

的单元数组 N：

```
N{1,1} = [1 2; 4 5];
N{1,2} = 'Name';
N{2,1} = 2 - 4i;
N{2,2} = 7;
```

可用 c=N{1,2}获取 N{1,2}中的字符串，即

```
c = N{1,2}
c =
    Name
```

注意：赋值时，可以使用内容索引获取单个单元的内容，而不是单元子集。例如，语句 A{1,:} = value 和 B = A{1,:}都是不合法的。但是，可以在逗号间隔的变量列表中的任何地方使用单元子集。

2．用单元索引获取单元子集

使用单元索引把全部单元子集赋给另一个变量，创建一个新的单元数组。使用 colon 操作符获取单元数组中的单元子集。

2.5.3 删除单元和重塑单元数组

可以用单条语句删除单元的整个维，像标准数组的删除一样，删除单元的行或列时使用向量索引，将空矩阵赋给该维。例如：

```
A(2) = []
```

删除单元时，大括号不出现在赋值语句中。

与其他数组一样，可用 reshape 函数重塑单元数组。单元个数必须与重塑以后的相同，不能使用 reshape 函数添加或删除单元。下面的命令行首先创建一个单元数组 A，然后利用 reshape 函数改变它的大小。

```
A = cell(3,4);
size(A)
ans =
     3     4
B = reshape(A,6,2);
size(B)
ans =
     6     2
```

2.5.4 采用函数和操作符

使用索引，可以将函数和操作符应用于单元内容。例如，下面使用内容索引调用一个以单元内容为变量的函数。

```
A{1,1} = [1 2; 3 4];
A{1,2} = randn(3,3);
A{1,3} = 1:5;
B = sum(A{1,1})
```

```
B =
     4     6
```

要将函数应用到多个非嵌套单元数组的单元，使用循环结构。例如：

```
for k = 1:length(A)
    M{k} = sum(A{1,k});
end
```

2.5.5 在单元数组中组织数据

单元数组对于组织由不同类型数据组成的数据很有用。与结构数组相比，它的好处在于以下几方面。

- 用一个语句获取数据的多个字段。
- 以逗号分隔的变量列表获取数据子集。
- 没有固定的字段名称集合。
- 可以按计划从结构中删除字段。

假设要组织的数据由下面这些部分组成：

- 一个 3×4 的数组。
- 数组元素为试验测量值。
- 一个包含技术名称的 15 字节字符串。
- 从过去 5 次试验得到的测量记录，是一个 3×4×5 的数值。

在大多数情况下，对于本数据，结构数组是组织数据的最佳方式。但是，如果按计划只获取前两个字段的信息，则使用字段数组可能更方便。本例演示如何获取单元数组 TEST 的第 1 个和第 2 个元素。

```
[newdata,name] = deal(TEST{1:2})
```

下面演示如何获取结构数组 TEST 的前两个元素。

```
newdata = TEST.measure
name = TEST.name
```

varargin 和 varargout 变量可以看做替代逗号分隔列表的单元数组工具的示例。创建一个数值数组 A：

```
A = [0 1 2;4 0 7;3 1 2];
```

现在给数组 A 应用 normest 函数，并把输出指定给数组 B 的各个单元。

```
[B{1:2}] = normest(A)
B =
    [8.8826]    [4]
```

所有输出值都保存在数组 B 的各个单元中，B(1)包含范数估计，B(2)包含迭代次数。

2.5.6 单元数组嵌套

单元数组的单元可以包含另一个单元数组，或者以单元数组为元素的数组。在 cell 函数中可以使用嵌套的大括号或者赋值语句创建嵌套的单元数组。然后可以获取和操作单个的单元、单元子数组或单元元素。

1．用嵌套的大括号创建嵌套数组

可以通过成对的大括号来创建一个嵌套的单元数组。例如：

```
clear A
A(1,1) = {magic(5)};

A(1,2) = {{[5 2 8; 7 3 0; 6 7 3] 'Test 1'; [2 - 4i 5+7i] {17 []}}}
A =
    [5x5 double]    {2x2 cell}
```

注意：语句的右端被两套大括号括起来，第一套表示单元数组 A 的单元(1,2)，第二套表示在外部单元中包括一个单元数组。

2．用 cell 函数生成嵌套数组

用 cell 函数嵌套单元数据，将单元的输出值赋给一个已经存在的单元。创建一个空的单元数组：A=cell(1,2); 在 A(1,2)中创建一个单元数组：A(1,2)={cell(2,2)}; 通过赋值来填充 A。

```
A(1,1) = {magic(5)};
A{1,2}(1,1) = {[5 2 8; 7 3 0; 6 7 3]};
A{1,2}(1,2) = {'Test 1'};
A{1,2}(2,1) = {[2 - 4i 5+7i]};
A{1,2}(2,2) = {cell(1,2)}
A{1,2}{2,2}(1) = {17};
```

连接索引表达式，可以索引嵌套的单元数据。第一套索引号获取顶层单元，后面依次获取更深层单元的数据。例如：数组 A 有 3 层嵌套，要在单元(1,1)中获取的数组，使用 A{1,1}；要获取单元(1,2)中位置(1,1)上的数组，使用 A{1,2}.{1,1}；要获取单元(1,2)中的数组，使用 A{1,2}；要获取单元(1,2)中位置(2,2)处的空单元，使用 A{1,2}.{2,2}.{1,2}。

2.5.7 在单元和数值数组之间转换

使用循环，在单元和数值数组之间转换。例如，创建一个单元数组 F：

```
F{1,1} = [1 2; 3 4];
F{1,2} = [-1 0; 0 1];
F{2,1} = [7 8; 4 1];
F{2,2} = [4i 3+2i; 1-8i 5];
```

现在使用 3 套循环，将下面的内容拷贝到数值数组 NUM 中。

```
for k = 1:4
    for m = 1:2
        for n = 1:2
            NUM(m,n,k) = F{k}(m,n);
        end
    end
end
```

类似地，必须使用循环来将数值数组中的每个值赋给单元数组中的单元。

```
G = cell(1,16);
for m = 1:16
```

```
    G{m} = NUM(m);
End
```

2.5.8 结构的单元数组

使用单元数组来保存具有不同字段的多组结构。

```
c_str = cell(1,2);
c_str{1}.label = '12/2/94 - 12/5/94';
c_str{1}.obs = [47 52 55 48; 17 22 35 11];
c_str{2}.xdata = [-0.03 0.41 1.98 2.12 17.11];
c_str{2}.ydata = [-3 5 18 0 9];
c_str{2}.zdata = [0.6 0.8 1 2.2 3.4];
```

c_str 数组的单元 1 包含一个两字段的结构，一个是字符串，另一个是向量。单元 2 包含一个有 3 个向量字段的结构。创建结构的单元数组时，必须使用内容索引。同样，必须使用内容索引来获取单元中结构的内容。内容索引的语法为：

```
cell_array{index}.field
```

例如，要获取单元 1 中结构的 label 字段，使用 c_str{1}.label。

2.6 函数句柄

2.6.1 概述

函数句柄是一种 MATLAB 数据类型，它包含用于引用函数的信息。创建函数句柄时，MATLAB 在句柄中保存所有与要运行的函数有关的信息。函数句柄往往在变量列表中传递给其他函数，然后与 eval 函数联合使用来运行句柄所属的函数。一个 MATLAB 函数句柄不仅是对函数的一个引用，它还经常表示一个函数方法的集合。这些方法重载后控制不同的变量类型。给函数创建一个句柄时，MATLAB 快速扫描与该函数名称相同的内部方法或 M 文件方法。评价函数句柄时，MATLAB 只考虑那些保存在句柄中的函数。

使用函数句柄的好处主要有以下几方面。

（1）把函数获取的信息传递给其他函数。
（2）掌握一个重载函数的所有方法。
（3）允许更广泛地获取子函数和私有函数。
（4）保证函数计算的可靠性。
（5）减少定义函数的字段个数。
（6）改进重复操作。
（7）在数组、结构和单元数组中操作句柄。
（8）把函数获取的信息传递给其他函数。
（9）可以在调用中将函数句柄作为变量传递给另一个函数。

2.6.2 创建一个函数句柄

在 MATLAB 中，在函数名前用@符号创建一个函数句柄。下面的例子为 humps 函数创

建一个函数句柄并将它指定给变量 fhandle。

```
fhandle = @humps;
```

可以像传递其他变量一样传递句柄到另一个函数。本例将创建的函数句柄传递给 fminbnd 函数，然后在区间[0.3　1]上进行最小化。

```
x = fminbnd(fhandle, 0.3, 1)
x =
    0.6370
```

fminbnd 函数用 feval 评价@humps 函数句柄。下面是 fminbnd M 文件的一小部分，第一行中，funfcn 输入参数接收传入的函数句柄@humps，第 113 行中的 feval 语句评价句柄。

```
1   function [xf,fval,exitflag,output] =…
        fminbnd(funfcn,ax,bx,options,varargin)
            …
113   fx = feval(funfcn,x,varargin{:});
```

注意：创建函数句柄时，只在@符号后面使用函数名，不能包括任何路径信息。下面的语法是错误的：

```
fhandle = @\home\user4\humps.
```

用于句柄的函数名最多只能有 N 个字符，其中，N 是函数 namelengthmax 返回的数字。如果函数名的字符数大于该长度，则 MATLAB 截断名称的后半部分。

```
N = namelengthmax
N =
    63
```

2.6.3　使用句柄

用 MATLAB 的 feval 命令运行函数句柄的目标函数。用函数句柄来使用本命令的语法为：

```
feval(fhandle, arg1, arg2,…, argn)
```

这与直接调用句柄所代表的函数效果差不多。主要差别在于：函数句柄能从参数传递的目标函数中进行评价。

下面的例子定义一个名为 plot_fhandle 的函数，它接收一个函数句柄和数据，然后基于 data 数据进行计算和绘图。

```
function x = plot_fhandle(fhandle, data)
plot(data, feval(fhandle, data))
```

做如下调用时，生成图 2-1。

```
plot_fhandle(@sin, -pi:0.01:pi)
```

2.6.4　函数句柄操作

MATLAB 提供了两个函数 func2str 和 str2func。使用它们，可以在函数句柄和函数名字符串之间转换。它还提供一些测试函数，看变量是否包含一个函数句柄。另外还可以比较函数句柄。

1．把函数句柄转换为函数名

如果需要完成字符串操作，如基于函数句柄进行字符串比较和显示，可以使用 func2str

以字符串格式获取函数名。例如，要把一个 sin 函数句柄转换为一个字符串，在命令窗口输入下面的命令行：

```
fhandle = @sin;

func2str(fhandle)
ans =
        sin
```

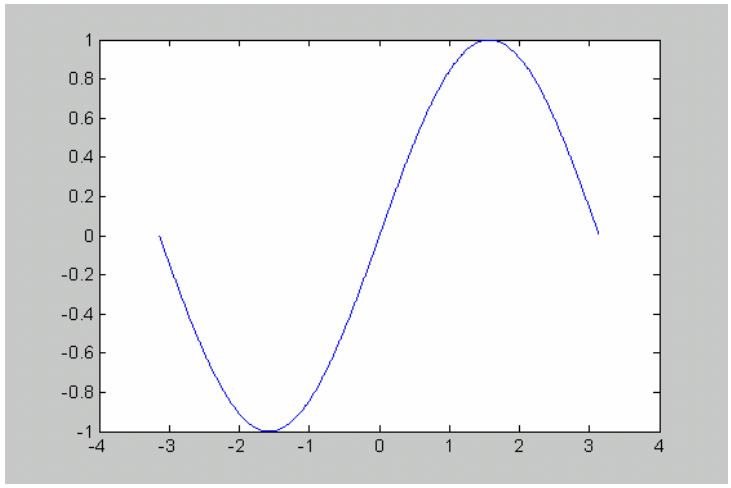


图 2-1　正弦曲线

注意：func2str 命令不操作非标量函数句柄。给 func2str 传递一个非标量函数句柄将生成一个错误。

下面举一个例子来说明如何把函数句柄转换为函数名。

首先创建一个 catcherr 函数，它接收函数句柄和数据变量，并试图用它的句柄来运行。如果运行失败，catcherr 函数使用 sprintf 函数显示出错信息，该出错信息给出运行失败的函数的名称。函数名必须是 sprintf 函数能显示的字符串。

```
function catcherr(func, data)
try
    ans = feval(func, data);
    disp('Answer is:');
    ans
catch
    sprintf('Error executing function "%s"\n', func2str(func))
end
```

下面调用 catcherr 函数，传递参数为一个 round 函数句柄和一个合法的数据变量。命令行能正常运行。第二个调用传递相同的函数句柄和一个数据类型不合法的数据变量，这次 round 函数运行失败，catcherr 函数显示一个出错信息，该出错信息包括运行失败的函数的名称。

```
catcherr(@round, 5.432)
ans =
Answer is 5
```

```
xstruct.value = 5.432;
catcherr(@round, xstruct)
Error executing function "round"
```

2．把函数名转换为函数句柄

使用 str2func **函数，可以创建一个源于字符串的函数句柄，该字符串包含一个** MATLAB **函数名。要把字符串“** sin **”转换为该函数的句柄，使用下面的语句：**

```
fh = str2func('sin')
fh =
    @sin
```

如果传递一个变量中的函数名字符串,接收该变量的函数能将函数名转换为使用 str2func **的函数句柄。下例传递一个变量** funcname **给函数** makeHandle**，然后创建一个函数句柄。**

```
function fh = makeHandle(funcname)
fh = str2func(funcname);
% -- end of makeHandle.m file --

makeHandle('sin')
ans =
    @sin
```

还可以基于函数名字符串单元数组运行 str2func **函数，此时，**str2func **函数返回一个函数句柄数组。**

```
fh_array = str2func({'sin' 'cos' 'tan'})
fh_array =
        @sin    @cos    @tan
```

2.6.5 测试数据类型

使用 functiom_handle **标记符，可以通过** isa **函数察看是否包含函数句柄。下面的函数测试一个输入变量，在使用它以前看它是否是一个函数句柄。**

```
function evaluate_handle(arg1, arg2)
if isa(arg1, 'function_handle')
    feval(arg1, arg2);
else
    disp 'You need to pass a function handle';
end
```

用 isequal **函数比较两个函数句柄，看它们是否相等。**

```
function test_myfun(arg1, arg2)
if isequal(arg1, @myfun)
disp 'Function handle holds unexpected context'
else
    feval(arg1, arg2);
end
```

2.6.6 保存和装载函数句柄

可以使用 MATLAB **的** save **和** load **函数保存** MAT **文件的函数句柄，然后把它们装载到** MATLAB **工作空间。本例演示一个函数句柄数组被保存到** savefile **文件中，然后被重新载入的过程。**

```
fh_array = [@sin @cos @tan];
save savefile fh_array;
clear

load savefile
whos
```

Name	Size	Bytes	Class
fh_array	1x3	48	function_handle array

第3章　数 值 运 算

3.1　MATLAB 中的变量

与其他计算机语言一样，MATLAB 中的变量也有自己的命名规则。MATLAB 中变量的命名规则如下：变量必须以字母打头，之后可以是任意字母、数字或下划线；变量名的字母有大小写之分；变量名不能超过 19 个字符，第 19 个字符以后的字符将被忽略。MATLAB 中默认的变量名为 ans。

【例 3-1】

```
x=[1 3 4;2,6,5;3 2,4];
2*x
ans =
     2     6     8
     4    12    10
     6     4     8
```

除了上述命名规则外，MATLAB 中还包括一些特殊的变量，如表 3-1 所示。

表 3-1　MATLAB 中的特殊变量

变 量 名 称	变 量 含 义
ans	MATLAB 中默认变量
pi	圆周率
eps	计算机中的最小数，PC 机上为 3^{-52}
inf	无穷大，如 1/0
NaN	不定值，如 0/0，∞/∞，0*∞
i(j)	复数中的虚数单位
nargin	所用函数的输入变量数目
nargout	所用函数的输出变量数目
realmin	最小可用正实数
realmax	最大可用正实数

一般地讲，在 MATLAB 中，数据的存储与计算都是以双精度进行的。当然，用户也可以改变其显示格式。控制数据显示格式的命令是 format，其调用格式如下。

- format/ format SHORT　　默认值，5 位定点表示。
- format LONG　　15 位定点表示。
- format SHORT E　　5 位浮点表示。
- format LONG E　　15 位浮点表示。
- format SHORT G　　在 5 位定点与 5 位浮点中选择最好的格式表示，MATLAB 自动选择。

- format LONG G　　在 15 位定点与 15 位浮点中选择最好的格式表示，MATLAB 自动选择。
- format HEX　　十六进制格式表示。
- format +　　在矩阵中，用符号 +、-和空格分别表示正号、负号和零。
- format BANK　　用美元与美分定点表示。
- format RAT　　对整数的近似表示。
- format COMPACT　　变量之间没有空行。
- format LOOSE　　变量之间有空行。

3.2　数组及向量运算

数组运算主要是针对多个数执行同样的计算而运用的。在 MATLAB 中，以一种非常直观的方式来处理数组。

3.2.1　数组构造

在 MATLAB 中，数组是这样表示的：用户只需以左方括号开始，以空格（或逗号）为间隔输入元素值，最后以右方括号结束，如下例所示。

【例 3-2】

```
x=[0 1 3 5 7 9 10]
x =
     0     1     3     5     7     9    10
```

数组构造可以采用如下几种方法。

1．利用 first:increment:last 来创建数组

first:increment:last 表示创建一个从 first 开始，到 last 结束，数据元素的增量为 increment 的数组。冒号表示直接定义数据元素之间的增量，而不是数据元素个数。若增量为 1，上面创建数组的方式可简写为：first:last。

【例 3-3】

```
x=(0:0.5:2)
x =
    0    0.5000    1.0000    1.5000    2.0000
```

上例表示创建一个从 0 开始、0.5 为增量、到 2 结束的数组 x。

2．利用函数 linspace 来创建数组

函数 linspace 通过直接定义数据元素个数，而不是数据元素之间的增量来创建数组。该函数的调用格式如下：

```
linspace(first_value,last_value,number)
```

该调用格式表示创建一个从 first_value 开始，到 last_value 结束，包含有 number 个数据元素的数组。

【例 3-4】

```
x=linspace(0,pi,12)
```

```
x =
Columns 1 through 7
    0         0.2856    0.5712    0.8568    1.1424    1.4280    1.7136
Columns 8 through 12
    1.9992    2.2848    2.5704    2.8560    3.1416
```

上例表示创建一个从 0 **开始、到** pi **结束、包含有** 12 **个数据元素的数组。**

上述两种方法是在 MATLAB **中最常用的创建数组的方法。**

3．利用 logspace **函数来创建一个对数分隔的数组**

与 linspace **函数一样，**logspace **函数也通过直接定义数据元素个数，而不是数据元素之间的增量来创建数组。**logspace **函数的调用格式如下：**

```
logspace(first_value,last_value,number)
```

该调用格式表示创建一个从 10^{first_value} **开始、到** 10^{last_value} **结束、包含有** number **个数据元素的数组。**

【例 3-5】

```
logspace(0,2,10)
ans =
Columns 1 through 7
    1.0000    1.6681    2.7826    4.6416    7.7426   12.9155   21.5443
Columns 8 through 10
   35.9381   59.9484  100.0000
```

上例表示创建一个从 10^0 **开始、到** 10^2 **结束、包含有** 10 **个数据元素的数组。**

以上创建数组的方法主要生成的是行向量的数组，有时还需创建列向量的数组，最简单的方法是以分号分隔，如下例所示。

【例 3-6】

```
x=[1;3;6;8;10]
x =
     1
     3
     6
     8
    10
```

由上述可知，以空格或逗号分隔的元素指定的是不同列的元素，而以分号分隔的元素指定的是不同行的元素，又如下例所示。

【例 3-7】

```
x=[1 3 4;2,6,5;3 2,4]
x =
     1     3     4
     2     6     5
     3     2     4
```

3.2.2 数组运算

1．数组与标量间的四则运算

数组与标量之间的四则运算是指数组中的每个元素与标量进行加、减、乘、除运算，如以下两例所示。

【例 3-8】

```
x=[1 3 4;2,6,5;3 2,4];
a=2*x-2
a =

     0     4     6
     2    10     8
     4     2     6
```

【例 3-9】

```
x=[1 3 4;2,6,5;3 2,4];
c=x/2
c =
    0.5000    1.5000    2.0000
    1.0000    3.0000    2.5000
    1.5000    1.0000    2.0000
```

2．数组间的四则运算

在 MATLAB 中，数组间进行四则运算时，参与运算的数组必须具有相同的维数，加、减、乘、除运算是按元素与元素的方式进行的。其中，数组间的加、减运算与矩阵间的加、减运算相同，运算符号为“+”,“-”；但是，数组间的乘、除运算与矩阵间的乘、除运算完全不同，运算符号为“.*”,“./”或“.\”（关于矩阵间的四则运算将在后面讨论）。注意，运算符号中的小点不能少，否则将不会按数组运算规则进行运算。

【例 3-10】

```
a=[1 3 4;2,6,5;3 2,4];
b=[2 3 1;4 1 2;4 5 3];
c=a+b
c =
     3     6     5
     6     7     7
     7     7     7
```

数组间的乘法与除法运算如下两例所示。

【例 3-11】

```
a=[1 3 4;2,6,5;3 2,4];
b=[2 3 1;4 1 2;4 5 3];
c=a.*b
c =
     2     9     4
```

```
     8     6    10
    12    10    12
```

【例 3-12】

```
a=[1 3 4;2,6,5;3 2,4];
b=[2 3 1;4 1 2;4 5 3];
c=a./b
c =
    0.5000    1.0000    4.0000
    0.5000    6.0000    2.5000
    0.7500    0.4000    1.3333
```

由于数组间的除法运算有点特殊，为了便于读者使用，这里对数组间的除法运算规则总结如下。

（1）数组间的除法运算为参与运算的数组对应元素相除，结果数组与参与运算的数组大小相同。

（2）数组与标量间的除法运算为数组中的每个元素与标量相除，结果数组与参与运算的数组大小相同。

（3）数组间的除法运算符号有两个，即左除号“ ./ ”与右除号“ .\ ”，它们之间的关系如下：

a./b=b.\a。

3. **数组的幂运算**

在 MATLAB **中，数组的幂运算与矩阵的幂运算完全不同。数组的幂运算符号为“** .^ **”（注意运算符号中的小点），表示元素对元素的幂运算。而矩阵的幂运算符号为“** ^ **”。**

【例 3-13】

```
a=[1 3 4;2,6,5;3 2,4];
c=a.^2
c =
     1     9    16
     4    36    25
     9     4    16
```

为便于比较，下例示出矩阵的幂运算规则。

【例 3-14】

```
a=[1 3 4;2,6,5;3 2,4];
c=a^2
c =
    19    29    35
    29    52    58
    19    29    38
```

【例 3-15】

```
a=[1 3 4;2,6,5;3 2,4];
b=[2 3 1;4 1 2;4 5 3];
c=a.^b
```

```
c =
     1    27     4
    16     6    25
    81    32    64
```

上例中两数组的幂运算均为数组中各对应元素间的运算。

4. 数组的指数运算、对数运算与开方运算

在 MATLAB 中，由于数组的运算实质上是数组内部每个元素的运算，因此，数组的指数运算、对数运算及开方运算与标量的运算规则完全一样，运算函数分别为“exp”,“log”,“sqrt”等。下例为数组的指数运算。

【例 3-16】

```
a=[1 3 4;2,6,5;3 2,4];
c=exp(a)
c =
    2.7183   20.0855   54.5982
    7.3891  403.4288  148.4132
   20.0855    7.3891   54.5982
```

数组的对数运算、开方运算与数组的指数运算完全一样。

3.2.3 向量运算

向量的运算主要包括向量的点积、叉积、混合积运算。下面分别介绍。

1. 向量的点积运算

在高等数学中，两向量的点积是指两个向量在其中一个向量方向上的投影的乘积，通常用来定义向量的长度。

在 MATLAB 中，向量的点积由函数“dot”来实现，“dot”函数的调用格式如下。

- C = dot(A,B)　　表示返回向量 A 与 B 的点积，结果放在向量 C 中。需要说明的是，向量 A 与 B 必须长度相等。另外，当 A 与 B 都是列向量时，dot (A,B)等同于 A'*B。
- C=dot(A,B,DIM)　　表示返回向量 A 与 B 在维数为 DIM 的点积，结果放在向量 C 中。

【例 3-17】

```
A=[2 4 5 3 1];
B=[3 8 10 12 13];
C=dot(A,B)
C =
   137
```

【例 3-18】

```
A=[2 4 5 3 1];
B=[3 8 10 12 13];
C=dot(A,B,4)
C =
     6    32    50    36    13
```

2. 向量的叉积运算

在高等数学中，向量的叉积返回的是与两向量组成的平面垂直的向量。在 MATLAB 中，

向量的叉积由函数“cross”来实现。函数“cross”的调用格式如下。

- C = cross(A,B)　　表示返回向量 A 与 B 的叉积，即 $C=A\times B$。需要说明的是，向量 A 与 B 必须是 3 个元素的向量。
- C = cross(A,B,DIM)　　表示返回向量 A 与 B 在 DIM 维的叉积。需要说明的是，向量 A 与 B 必须要有相同的大小，size(A,DIM)和 size(B,DIM)的结果必须为 3。

【例 3-19】

```
A=[2 4 5];
B=[3 8 10];
C=cross(A,B)
C =
     0    -5     4
```

3. 向量的混合积运算

向量的混合积运算由上面介绍的“dot”和“cross”两个函数共同来实现。

【例 3-20】

```
A=[2 4 5];
B=[3 8 10];
C=[0 -5 4];
D=dot(A,cross(B,C))
D =
    41
```

上例表示，首先进行的是向量 B 与 C 的叉积运算，然后再把叉积运算的结果与向量 A 进行点积运算。

3.3 矩阵运算

如果说 MATLAB 的最大特色是强大的矩阵运算功能，此话毫不为过。事实上，MATLAB 中所有的计算都是以矩阵为基本单元进行的。MATLAB 对矩阵的运算功能最全面，也最强大。

3.3.1 矩阵构造

矩阵的表示形式与数组类似，用户只需以左方括号开始，以右方括号结束，矩阵同行之间以空格或逗号分隔，行与行之间以分号或回车符分隔。

【例 3-21】

```
a=[1 2 3 4 6 4 3 2 7 8]
a =
     1     2     3     4     6     4     3     2     7     8
```

【例 3-22】

```
a=[1 2 0;2 5 -1;4 10 -1]
a =
     1     2     0
     2     5    -1
     4    10    -1
```

有时候，用户可能需要输入大量的数据，以产生一个大型矩阵，此时就不太适合用手工方式在命令窗口输入了，因为这样不仅繁琐而且容易出错。为此，MATLAB 中专门提供了一个矩阵编辑器(Matrix Editor)来方便用户创建和修改大型矩阵。

单击 MATLAB 的 View 菜单，从中选择 Workspace 子菜单。Workspace 窗口中显示出已经创建了的矩阵变量(如图 3-1 所示)。双击其中任何一个已经创建的矩阵，就会弹出 MATLAB 的矩阵编辑器界面，如图 3-2 所示。用户可以在 Numric format 中设定输入数据的格式，在 Size 中设定矩阵的行数与列数，其中，第一个编辑框中指定行数，第二个编辑框中指定列数。例如，在图 3-2 中 2×4 矩阵的基础上，再增加 4 行 2 列，就变成了 6×6 矩阵，如图 3-3 所示。然后，用户就可以输入数据了。数据输入完毕，单击 File 主菜单中的 Close Array Editor 子菜单，就可以保存创建的矩阵。

需要说明的是，用户在使用矩阵编辑器时，最好预先定义一个矩阵变量。

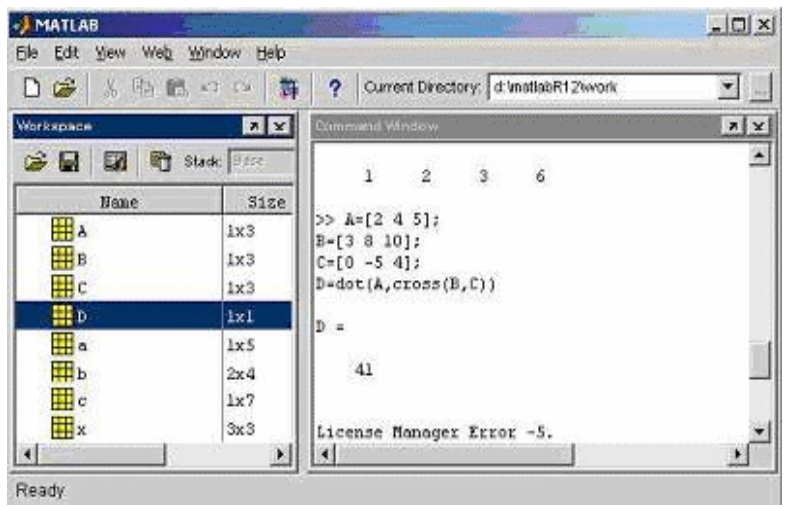


图 3-1　MATLAB 中的 Workspace 窗口界面

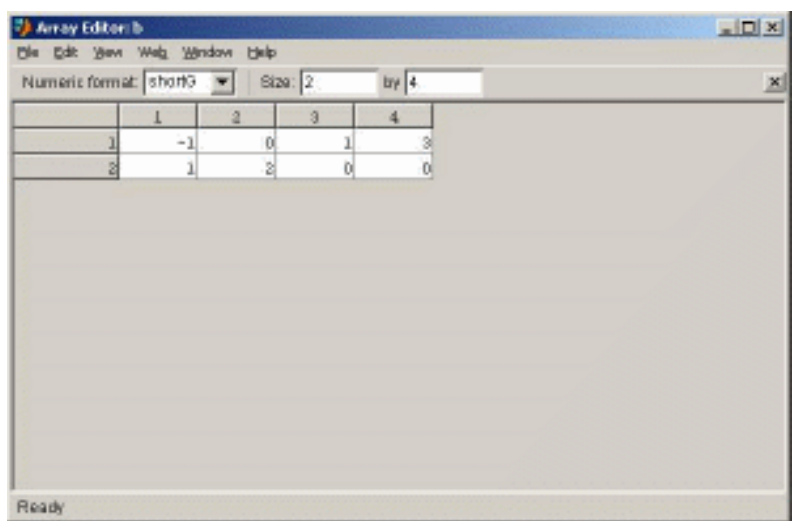


图 3-2　MATLAB 的矩阵编辑器

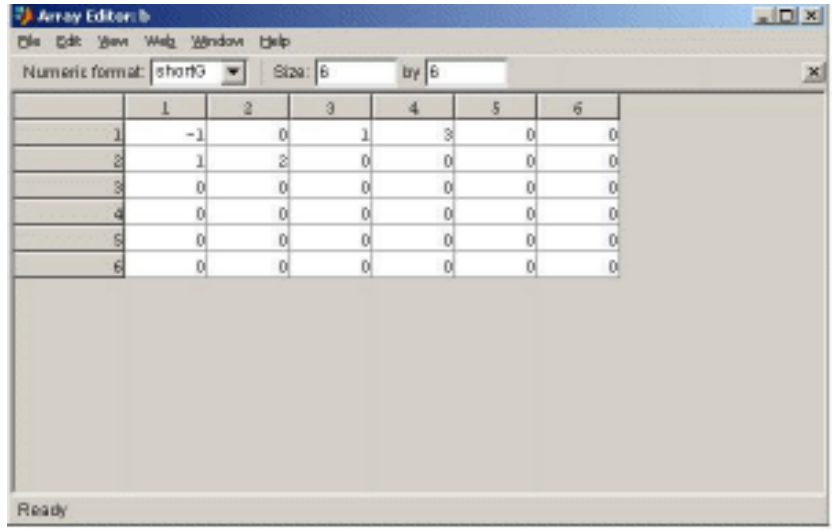


图 3-3　矩阵编辑器中扩充后的矩阵

使用矩阵编辑器，用户很容易修改矩阵。通过改变图 3-2 中 Size 标签后面两个文本框中的值，可以把原矩阵缩小为它左上方的子矩阵，也可以把原矩阵扩充为更大的矩阵。

3.3.2　矩阵的基本运算

矩阵的基本运算包括矩阵的四则运算，矩阵与标量的运算，矩阵的幂运算、指数运算、对数运算、开方运算以及矩阵的逆运算、行列式运算等。下面分别对上述运算进行说明。

1．矩阵的四则运算

矩阵的四则运算与前面介绍的数组的四则运算基本相同，但也有一些差别。其中，矩阵的加、减运算与数组的加、减运算完全相同，但要求进行运算的两个矩阵的大小完全相同，使用的运算符号也是“ + ”与“ - ”。

【例 3-23】

```
a=[1 2;3 5;2 6];
b=[2 4;1 8;9 0];
c=a+b
c =
     3     6
     4    13
    11     6
```

在线性代数中规定，矩阵的乘法运算要求进行相乘的两矩阵要有相同的公共维。设 A 矩阵为一个 $i\times j$ 大小的矩阵，则要求与之相乘的 B 矩阵必须是一个 $i\times k$ 大小的矩阵，此时 A 矩阵与 B 矩阵才能相乘。矩阵的乘法运算使用的运算符号是“ * ”。

【例 3-24】

```
a=[1 2;3 5;2 6];
b=[2 4 1;8 9 0];
c=a*b
c =
```

```
    18    22     1
    46    57     3
    52    62     2
```

当然，矩阵乘法也可以像数组乘法那样进行矩阵元素的相乘，此时要求相乘的两个矩阵大小完全相同，使用的运算符号为“ .* ”。

【例 3-25】

```
a=[1 2 0;2 5 -1;4 10 -1];
c=[1 2 4;2 5 10;0 -1 -1]
d=c.*a
d =
     1     4     0
     4    25   -10
     0   -10     1
```

在 MATLAB 中，矩阵的除法运算有两个运算符号，分别为左除符号“ \ ”与右除符号“ / ”。矩阵的右除运算速度要慢一点，而左除运算可以避免奇异矩阵的影响。对于方程 $Ax=b$，若此方程为超定方程，则使用除法运算符“ \ ”与“ / ”可以自动找到使 $Ax-b$ 的平方最小化的解。若此方程为不定方程，则使用除法运算符“ \ ”与“ / ”求得的解至多有 Rank(A)（矩阵 A 的秩）个非零元素，而且求得的解是这种类型的解中范数最小的一个。

【例 3-26】

```
a=[21 34 20;5 78 20;21 14 17;34 31 38];
b=[10 20 30 40]';
x=b\a
x =
    0.7667
    1.1867
    0.8767
```

上例中的方程 $Ax=b$ 为超定方程。注意，结果矩阵 x 是列向量形式。

【例 3-27】

```
a=[21 34 20 5;78 20 21 14;17 34 31 38];
b=[10 20 30]';
x=b\a
x =
    1.6286
    1.2571
    1.1071
    1.0500
```

上例中的方程 $Ax=b$ 为不定方程。

2．矩阵与标量间的四则运算

矩阵与标量间的四则运算和数组与标量间的四则运算完全相同，即矩阵中的每个元素与标量进行加、减、乘、除四则运算。需要说明的是，当进行除法运算时，标量只能做除数。

【例 3-28】

```
b=[21 34 20;78 20 21;17 34 31];
c=b+2
c =
    23    36    22
    80    22    23
    19    36    33
```

【例 3-29】

```
b=[21 34 20;78 20 21;17 34 31];
c=b/2
c =
   10.5000   17.0000   10.0000
   39.0000   10.0000   10.5000
    8.5000   17.0000   15.5000
```

3．矩阵的幂运算

矩阵的幂运算与数组的幂运算不同，数组的幂运算使用运算符“.^”，用来表示对数组中的元素进行幂运算。而矩阵的幂运算使用运算符“^”，它并不是对矩阵的每个元素进行幂运算，这与矩阵的某种分解方式有关，这一点需要注意。

【例 3-30】

```
b=[21 34 20;78 20 21;17 34 31];
c=b^2
c =
    3433    2074    1754
    3555    3766    2631
    3536    2312    2015
```

4. 矩阵的指数运算、对数运算与开方运算

矩阵的指数运算、对数运算与开方运算与数组的指数运算、对数运算、开方运算不同，它并不是对矩阵中的单个元素的运算，而是对整个矩阵的运算，这一点与矩阵的幂运算相类似。其中，矩阵的指数运算函数为“expm**”，“**expm1**”，“**exmp2**”，“**exmp3**”，矩阵的对数运算函数为“**logm**”，矩阵的开方运算函数为“**sqrtm**”。**

【例 3-31】

```
a=[1 3 4;2,6,5;3 2,4];
c=expm(a)
c =
   1.0e+004 *
    0.4668    0.7694    0.9200
    0.7919    1.3065    1.5613
    0.4807    0.7919    0.9475
```

【例 3-32】

```
a=[1 3 4;2,6,5;3 2,4];
c=logm(a)
c =
```

```
   0.5002 + 2.4406i   0.5960 - 0.6800i   0.7881 - 1.2493i
   0.4148 + 0.4498i   1.4660 - 0.1253i   1.0108 - 0.2302i
   0.5780 - 1.6143i   0.4148 + 0.4498i   1.0783 + 0.8263i
```

上面结果表示：产生的运算结果是复数形式，其中，符号“ i ”为复数的虚数单位。

【例 3-33】

```
a=[1 3 4;2,6,5;3 2,4];
c=sqrtm(a)
c =
   0.6190 + 0.8121i   0.8128 - 0.2263i   1.1623 - 0.4157i
   0.3347 + 0.1497i   2.3022 - 0.0417i   1.1475 - 0.0766i
   1.0271 - 0.5372i   0.3347 + 0.1497i   1.6461 + 0.2750i
```

5. 矩阵的转置、逆运算与行列式运算

与线性代数中一样，矩阵的转置只需用符号“ ’”来表示即可。

【例 3-34】

```
a=[1 2 0;2 5 -1;4 10 -1];
C=a'
C =
     1     2     4
     2     5    10
     0    -1    -1
```

线性代数中论述的求矩阵的逆运算非常复杂，而在 MATLAB 中，矩阵的逆运算只需用函数“ inv ”来实现，这大大简化了计算过程。例如，求矩阵 B 的逆：A=inv(B)。

【例 3-35】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=inv(a)
b =
     5     2    -2
    -2    -1     1
     0    -2     1
```

在 MATLAB 中，求矩阵的行列式大小，可用函数“ det ”来实现。

【例 3-36】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=det(a)
b =
     1
```

3.3.3 矩阵的特殊运算

矩阵的特殊运算包括矩阵的特征值运算、条件数运算、奇异值运算、范数运算、秩运算、正交化运算、迹运算、伪逆运算等，下面分别介绍。

1．矩阵的特征值运算

在线性代数中，计算矩阵的特征值过程相当复杂。而在 MATLAB 中，矩阵的特征值运

算只需用函数“eig”或“eigs”计算即可得到。

【例 3-37】

```
a=[1 2 0;2 5 -1;4 10 -1];
[b,c]=eig(a)
b =
   -0.2440   -0.9107    0.4472
   -0.3333    0.3333    0.0000
   -0.9107   -0.2440    0.8944
c =
    3.7321         0         0
         0    0.2679         0
         0         0    1.0000
```

上例中的矩阵 b，c 分别为特征向量矩阵和特征值矩阵。

2．矩阵的条件数运算

在 MATLAB 中，矩阵的条件数可分别由函数“cond(A)”,“condest(A)”或“rcond(A)”计算得到，它们分别代表计算矩阵的条件数值、计算 1-范数矩阵条件数值、计算矩阵的逆条件数值。学过矩阵分析的读者可能还记得，条件数的值代表矩阵“病态”程度的大小。

【例 3-38】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=cond(a)
b =
    78.1154
```

【例 3-39】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=condest(a)
b =
   119
```

【例 3-40】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=rcond(a)
b =
    0.0084
```

3．矩阵的奇异值运算

在 MATLAB 中，矩阵的奇异值一般通过函数“svd(A)”和“svds(A)”计算得到。

【例 3-41】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=svd(a)
b =
   12.3171
    0.5149
    0.1577
```

【例 3-42】

```
a=[1 2 0;2 5 -1;4 8 -1];
b=svds(a)
b =
    10.7543
     0.5645
     0.1647
```

4. **矩阵的范数运算**

在 MATLAB **中，矩阵的范数运算可由函数“** norm(A) **”,“** norm(A,1) **”,“** norm(A,2) **”,“** norm(A,inf) **”,“** norm(A,'fro') **”等来实现，它们分别执行矩阵的范数运算、**1-**范数运算、**3-**范数运算、无穷大范数运算和** F-**范数运算。此外，函数“** normest(A) **”也可以计算矩阵的** 3-**范数值。**

【例 3-43】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=norm(a,2)
b =
    12.3171
```

【例 3-44】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=normest(a)
b =
    12.3171
```

【例 3-45】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=norm(a,inf)
b =
    15
```

【例 3-46】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=norm(a,'fro')
b =
    12.3288
```

5. **矩阵的秩运算**

矩阵的秩在线性代数中的运用非常广泛，但在线性代数中的计算也非常复杂。而在 MATLAB **中，矩阵的秩只需用函数“** rank(A) **”来计算就可求得。**

【例 3-47】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=rank(a)
b =
    3
```

6. 矩阵的正交化运算

在 MATLAB 中，矩阵的正交化运算可由函数“ orth(A) ”来实现。下面的例子是求矩阵的一组正交基。

【例 3-48】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=orth(a)
b =
    -0.1799    0.5217   -0.8340
    -0.4434   -0.7998   -0.4047
    -0.8781    0.2970    0.3752
```

7. 矩阵的迹运算

矩阵的迹是指矩阵所有对角线元素的和。在 MATLAB 中，矩阵的迹可由函数“ trace(A) ”计算得到。

【例 3-49】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=trace(a)
b =
     5
```

8. 矩阵的伪逆运算

矩阵的伪逆运算对严重病态的矩阵求解有重要作用。在 MATLAB 中，矩阵的伪逆运算由函数“ pinv(A) ”来实现。

【例 3-50】

```
a=[1 2 0;2 5 -1;4 10 -1];
b=pinv(a)
b =
    5.0000    2.0000   -2.0000
   -2.0000   -1.0000    1.0000
    0.0000   -2.0000    1.0000
```

3.3.4 矩阵的分解运算

矩阵的分解运算主要包括矩阵的特征值分解、Cholesky 分解、QR 分解、QZ 分解、LU 分解、Schur 分解、奇异值分解等。在 MATLAB 中，分别由函数“ chol(A) ”,“ qr(A) ”,“ lu(A) ”,“ schur(A) ”和“ svd(A) ”来实现上述分解运算。

1. 矩阵的特征值分解

在线性代数中，很多情况下需要求矩阵的特征值。然而，其求解方法却比较复杂。现在，利用 MATLAB 中的求矩阵特征值的函数，就很容易求解。在 MATLAB 中，求矩阵特征值的函数是“ eig ”和“ eigs ”。其中，函数“ eigs ”主要应用于稀疏矩阵。函数“ eig ”的调用格式有如下几种。

- D= eig (A)　　D 为矩阵 A 的特征向量矩阵。
- D= eig(A,B)　　D 为矩阵 A，B 的广义特征向量矩阵。

- [V,D] = eig(A)　　矩阵 V，D 分别为特征向量矩阵和特征值矩阵，因此满足 AV=VD。
- [V,D] = eig(A,'nobalance')　　禁止“平衡”程序的运行。当在矩阵 A 中有的元素小到与截断误差相当时，这样做可以减少计算的误差。
- [V,D] = eig(A,B)　　矩阵 V，D 分别为矩阵 A，B 的广义特征向量矩阵和特征值矩阵，因此满足 A*V=B*V*D。
- [V,D] = eig(A,B,flag)　　使用某种分解算法来计算矩阵的特征值与特征向量，参数 flag 可以取“chol”,“qz”。矩阵 V，D 分别为矩阵 A，B 的广义特征向量矩阵与特征值矩阵。其中，参数 flag 取“chol”表示对 B 矩阵使用 Cholesky 分解算法来计算矩阵 A，B 的特征向量与特征值。如果矩阵 A 为对称矩阵，矩阵 B 为对称正定矩阵，则此算法为默认算法。参数 flag 取“qz”，则忽略对称性，即不管矩阵 A，B 是否为对称矩阵，均把它们当成非对称矩阵来计算。

【例 3-51】

```
B = [3 -2 -.9 2*eps; -2 4 -1 -eps; -eps/4 eps/2 -1 0; -.5 -.5 .1 1];
[VB,DB] = eig(B)
VB =
    0.6153   -0.4176   -0.0000   -0.3305
   -0.7881   -0.3261   -0.0000   -0.2949
   -0.0000   -0.0000    0.0000   -0.8136
    0.0189    0.8481    1.0000   -0.3765
DB =
    5.5616         0         0         0
         0    1.4384         0         0
         0         0    1.0000         0
         0         0         0   -1.0000
```

式中，eps 为计算机的最小值，不同的计算机，此值略有不同。比如，笔者的计算机，此值为 2.2204e-016。

【例 3-52】

```
a=[1 2 0;2 5 -1;4 10 -1];
[V,D] = eig(a,'nobalance')
V =
   -0.2679   -1.0000    0.5000
   -0.3660    0.3660    0.0000
   -1.0000   -0.2679    1.0000
D =
    3.7321         0         0
         0    0.2679         0
         0         0    1.0000
```

【例 3-53】

```
a=[1 3 4;2,6,5;3 2,4];
b=[1 2 0;2 5 -1;4 10 -1];
[V,D] = eig(a,b,'chol')
```

```
V =
    1.0000          0.8609 - 0.1391i          0.8609 + 0.1391i
   -0.4411         -0.1323 - 0.0229i         -0.1323 + 0.0229i
   -0.3686          0.0671 + 0.0486i          0.0671 - 0.0486i
D =
  -15.2537                0                         0
         0          1.1268 + 0.3271i                0
         0                0                  1.1268 - 0.3271i
```

2. **矩阵的** Cholesky **分解**

设矩阵 A **为** n **阶对称正定矩阵，则** A **矩阵可分解为** LL′**，即** A= LL′**。其中，矩阵** L **是上三角矩阵。此时，这种分解就称为** Cholesky **分解。在** MATLAB **中，**Cholesky **分解由函数“** chol **”实现。“** chol **”函数的调用格式有以下两种。**

- R = chol(X)　　**要求矩阵** X **是一个正定矩阵，否则返回出错信息。返回的矩阵** R **是一个上三角矩阵。**
- [R,p] = chol(X)　　**使用这种方式，不管矩阵** X **是否是一个正定矩阵，都不会返回出错信息。如果矩阵** X **是一个正定矩阵，则返回的矩阵** R **是一个上三角矩阵，**p **为零。如果矩阵** X **不是一个正定矩阵，则返回的矩阵** R **是一个上三角矩阵，**p **是一个正数。**

【例 3-54】

```
a=[3 -1 1;-1 5 2;1 2 4];
b=chol(a)
b =
    1.7321   -0.5774    0.5774
         0    2.1602    1.0801
         0         0    1.5811
```

下例中矩阵 B **不是一个正定矩阵。**

【例 3-55】

```
B = [3 -2 -.9 2*eps;  -2 4 -1 -eps;  -eps/4 eps/2 -1 0;  -.5 -.5 .1 1];
[b,p]=chol(B)
b =
    1.7321   -1.1547
         0    1.6330
p =
     3
```

此例中，若采用第一种调用方式，就会返回出错信息，如下例所示。

【例 3-56】

```
B = [3 -2 -.9 2*eps;  -2 4 -1 -eps;  -eps/4 eps/2 -1 0; -5 -.5 .1 1];
b=chol(B)
??? Error using ==> chol
Matrix must be positive definite.
```

3．**矩阵的** QR **分解**

一般地讲，实数矩阵 A **可进行** QR **分解，即** A=QR**。其中，**Q **为正交矩阵，**R **为上三角**

矩阵。

在 MATLAB 中，QR 分解可由函数“qr(A)”来实现。常用的调用格式如下。

- [B,C]=qr(A)　返回的矩阵 C 为上三角矩阵，矩阵 B 为满阵。
- [Q,R,E] = qr(A)　返回的矩阵 E 是置换矩阵，矩阵 R 是上三角矩阵，矩阵 Q 是满阵。上述矩阵满足关系 A*E = Q*R。

【例 3-57】

```
a=[1 2 0;2 5 -1;4 10 -1];
[b,c,e]=qr(a)
b =
    -0.1761    0.4598   -0.8704
    -0.4402   -0.8276   -0.3482
    -0.8805    0.3219    0.3482
c =
   -11.3578    1.3207   -4.5783
         0    0.5058    0.0920
         0         0   -0.1741
e =
     0     0     1
     1     0     0
     0     1     0
```

4．矩阵的 QZ 分解

在 MATLAB 中，QZ 分解可由函数“qz”来实现。“qz”函数常用的调用格式如下。

- [AA,BB,Q,Z,V] = qz(A,B)　要求矩阵 A，B 是方阵。产生的矩阵 AA，BB 是上三角矩阵，矩阵 Q，Z 是正交矩阵，矩阵 V 是特征向量矩阵。其中，满足 Q*A*Z = AA 与 Q*B*Z = BB。
- [AA,BB,Q,Z,V] = qz(A,B,flag)　方阵 A，B 的 QZ 分解取决于参数 flag。参数 flag 可取'complex'或'real'。其中，'complex'表示复分解产生三角矩阵 AA，'real'表示实分解产生三角矩阵 AA。

【例 3-58】

```
a=[1 2 0;2 5 -1;4 10 -1];
x=[1 3 4;2,6,5;3 2,4];
[b,c,e,z,v]=qz(a,x)
b =
   -0.1783 + 0.0000i   -1.4551 + 0.4813i   -3.0105 + 0.2715i
         0             5.6957 - 1.6531i   10.1589 - 1.1010i
         0                   0             0.9081 + 0.2636i
c =
    2.7199             -7.4009 + 0.2393i   -2.2295 + 1.2686i
         0              6.9587              0.4560 - 1.1530i
         0                   0              1.1095
e =
```

```
   -0.5730 - 0.0018i   -0.7935 - 0.0026i    0.2051 + 0.0007i
    0.0758 - 0.0217i    0.1978 + 0.0143i    0.9769 - 0.0053i
   -0.8158 + 0.0013i    0.5753 + 0.0057i   -0.0531 + 0.0255i
z =
    0.8670 - 0.0028i    0.4652 - 0.0196i    0.1583 - 0.0804i
   -0.3824 + 0.0012i    0.4177 - 0.1304i    0.8099 - 0.0801i
   -0.3196 + 0.0010i    0.7624 + 0.1027i   -0.5396 - 0.1223i
v =
    0.9968 - 0.0032i    0.9510 - 0.0490i    0.0519 - 0.9481i
   -0.4397 + 0.0014i   -0.1409 - 0.0406i    0.0400 + 0.1407i
   -0.3674 + 0.0012i    0.0671 + 0.0607i   -0.0603 - 0.0671i
```

5. 矩阵的 LU 分解

矩阵的 LU 分解是线性方程组求解方法中高斯消去法的基础，在 MATLAB 中由函数“lu(A)”来实现。“lu(A)”函数常用的调用格式如下。

- [L,U] = lu(X)　U **矩阵为上三角矩阵，满足** X = L*U。
- [L,U,P] = lu(X)　**返回的矩阵** P **是置换矩阵，矩阵** U **是上三角矩阵，矩阵** L **是满阵。上述矩阵满足关系** L*U = P*X。

【例 3-59】

```
a=[1 2 0;2 5 -1;4 10 -1];
[b,c,p]=lu(a)
b =
    1.0000         0         0
    0.2500    1.0000         0
    0.5000         0    1.0000
c =
    4.0000   10.0000   -1.0000
         0   -0.5000    0.2500
         0         0   -0.5000
p =
     0     0     1
     1     0     0
     0     1     0
```

6. 矩阵的 Schur 分解

在 MATLAB 中，矩阵的 Schur 分解由函数“Schur(A)”来实现，其调用格式为 [b,c]=schur(A)**，其中，矩阵** c **为** Schur **矩阵。**

【例 3-60】

```
a=[1 2 0;2 5 -1;4 10 -1];
[b,c]=schur(a)
b =
   -0.2440   -0.8797   -0.4082
   -0.3333    0.4714   -0.8165
```

```
    -0.9107    0.0632    0.4082
c =
    3.7321   -1.2247   11.5618
    0         0.2679   -1.3509
    0         0         1.0000
```

7. **矩阵的奇异值分解**

在 MATLAB **中，矩阵的奇异值分解由函数‘** svd(A) **’来实现，其调用格式为**[b,c,d]=svd(A)**。**

【例 3-61】

```
a=[1 2 0;2 5 -1;4 10 -1];
[b,c,d]=svd(a)
b =
    -0.1799    0.5217   -0.8340
    -0.4434   -0.7998   -0.4047
    -0.8781    0.2970    0.3752
c =
    12.3171    0         0
     0         0.5149    0
     0         0         0.1577
d =
    -0.3718    0.2136   -0.9034
    -0.9221    0.0275    0.3860
     0.1073    0.9765    0.1868
```

3.3.5 特殊矩阵

MATLAB **中提供了几个特殊矩阵，主要包括如下** 7 **个矩阵。**

1. **空矩阵**

空矩阵用“ [ ] **”表示，空矩阵的大小为零，但变量名却存在于工作空间内。另外，也可以用** clear **从内存中清除空矩阵变量。**

2. **单位矩阵**

在 MATLAB **中，单位矩阵可用函数“** eye(n,m) **”来实现，其中，**n **为要生成的矩阵的行数，**m **为要生成的矩阵的列数。**

【例 3-62】

```
a=eye(4,3)
a =
     1     0     0
     0     1     0
     0     0     1
     0     0     0
```

3. **全部元素是** 1 **的矩阵**

在 MATLAB **中，全部元素为** 1 **的矩阵可用函数“** ones(n,m) **”来实现，其中，**n **为要生成的矩阵的行数，**m **为要生成的矩阵的列数。**

【例 3-63】

```
a=ones(4,3)
a =
     1     1     1
     1     1     1
     1     1     1
     1     1     1
```

4. **全部元素是 0 的矩阵**

在 MATLAB **中，零矩阵可用函数“** zeros(n,m) **”来实现，其中，**n **为要生成的零矩阵的行数，**m **为要生成的零矩阵的列数。**

【例 3-64】

```
a=zeros(4,3)
a =
     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

5. **魔方矩阵**

在 MATLAB **中，魔方矩阵可用函数“** magic(n) **”来实现，其中，**n **为要生成的矩阵的行数与列数。**

【例 3-65】

```
a=magic(4)
a =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

6. **随机矩阵**

在 MATLAB **中，随机矩阵可由函数“** rand(n,m) **”或“** randn(n,m) **”来实现，它们分别表示生成元素服从** 0~1 **间均匀分布的随机矩阵和元素服从均值为零、方差为** 1 **的正态分布随机矩阵。**

【例 3-66】

```
a=rand(4,3)
a =
    0.9501    0.8913    0.8214
    0.2311    0.7621    0.4447
    0.6068    0.4565    0.6154
    0.4860    0.0185    0.7919
```

7. **伴随矩阵**

在 MATLAB **中，某个矩阵的伴随矩阵可用函数“** compan(A) **”来实现。**

【例 3-67】

```
u = [1   0   -7   6];
A = compan(u)
A =
     0     7    -6
     1     0     0
     0     1     0
```

注意，输入函数 compan **中的变量必须是向量形式，而不能是矩阵。**

3.3.6 稀疏矩阵

在许多应用场合，如数值计算中的有限单元法求解方程中，所处理的矩阵通常只有很少数的非零元素（有时，非零元素的个数只占矩阵元素总个数的 1%**左右），这种矩阵就称为稀疏矩阵。对这类矩阵，若像满矩阵那样存储所有的元素，那么对计算机资源是一种很大的浪费。为了避免这种浪费，一般只存储非零元素及与之相配的行号、列号。相应地，为了避免对零元素的运算，开发了针对这类矩阵的特殊算法。在** MATLAB **中，对稀疏矩阵的处理正是与上述思想相一致的。**

1. **稀疏矩阵的生成**

MATLAB **中，创建稀疏矩阵一般用函数“** sparse **”或“** spdiags **”来实现。其中，函数** sparse **的调用格式如下。**

- A=sparse(B)　　**表示将矩阵** B **转化为稀疏矩阵** A**。**
- A=sparse(i,j, s,m,n,nzmax)　　**表示生成** *m*×*n* **大小的矩阵。其中，**s **为所有非零元素构成的向量，**i**，**j **为非零元素的行下标与列下标。参数中的** nzmax **为存储非零元素的空间，它是一个正整数，其大小一定要大于等于非零元素的总数。**
- A= sparse(i,j, s,m,n)　　**参数** nzmax=length(s)**，其他的与上同。**
- A= sparse(i,j, s)　　**参数** m=max(i)**，**n=max(j)**，**nzmax=length(s)**，其他的与上同。**
- A= sparse(m,n)　　**这是** sparse([],[],[],m,n,0)**的简化形式。**

函数 spdiags **的调用格式如下。**

- [B,d] = spdiags(A)　　**从矩阵** A **中抽取所有非零元素，形成一个稀疏对角矩阵带。**d **是** A **矩阵中所有非零元素向量。**
- B = spdiags(A,d)　　**从矩阵** A **中抽取被** d **指定的对角矩阵。**
- A = spdiags(B,d,A)　　**用矩阵** B **的列代替被** d **指定的对角矩阵。输入仍然是稀疏矩阵。**
- A = spdiags(B,d,m,n)　　**生成一个** *m*×*n* **大小的矩阵** A**，**d **是长度为** *p* **的整数向量，它指定** A **的对角线位置。**B **矩阵用来给出矩阵** A **的对角线元素值，其大小为** min(*m*,*n*) ×*p***。**

【例 3-68】

```
a=sparse(1:8,1:8,3*ones(1,8))
a =
   (1,1)        3
   (2,2)        3
   (3,3)        3
   (4,4)        3
   (5,5)        3
```

(6,6) 3
(7,7) 3
(8,8) 3

【例 3-69】

```
n=8;
e = ones(n,1);
A = spdiags([e -2*e e], -1:1, n, n)
A =
   (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
   (6,5)        1
   (5,6)        1
   (6,6)       -2
   (7,6)        1
   (6,7)        1
   (7,7)       -2
   (8,7)        1
   (7,8)        1
   (8,8)       -2
```

另外，MATLAB **还提供了生成其他形式稀疏矩阵的函数，如表** 3-2 **所示。**

表 3-2　**生成常用的稀疏矩阵的函数**

函　数　名	说　　明
sparse	最常用的稀疏矩阵生成函数
spdiags	以对角带生成稀疏矩阵
spconvert	由外部数据输入生成稀疏矩阵
find	求出非零元素在稀疏矩阵中的索引
speye	生成稀疏单位矩阵
sprand	生成稀疏的均匀分布随机矩阵
sprandn	生成稀疏的正态分布随机矩阵
sprandsym	生成稀疏的对称随机矩阵
full	把稀疏矩阵转化为满矩阵

上述函数的具体调用格式，请详见 MATLAB 的用户指南与联机帮助。

2. 稀疏矩阵的运算

由于稀疏矩阵的存储空间变小，而且 MATLAB 专门开发了针对它的算法，因此对稀疏矩阵的运算非常快。下面两例是用稀疏矩阵求解与用满矩阵求解的运算速度的比较。

【例 3-70】

```
n=1500;
e = ones(n,1);
A = spdiags([e -2*e e],  -1:1, n, n);
B=ones(n,1);
Tic;
x=A\B;
t1=toc
t1 =       %运算时间
      1.7720
```

【例 3-71】

```
C=full(A);
tic;
x=B\C;
t2=toc
t2 =        %运算时间
     41.6800
```

可见，采用稀疏矩阵的运算速度比采用满矩阵的运算速度快得多。

3.4 多项式运算

多项式运算是数学中最基本的运算之一。在高等代数中，多项式一般可表示为以下形式：$f(x)=a_0x^n+a_1x^{n-1}+a_2x^{n-2}+a_3x^{n-3}+a_4x^{n-4}+\cdots+a_{n-1}x+a_n$。对于这种表示形式，很容易用一个行向量来表示，即 $T=[a_0,a_1,a_2\cdots,a_{n-1},a_n]$。在 MATLAB 中，多项式正是用这样一个行向量表示的，它的系数是按降序排列的。

3.4.1 多项式构造

由上述可知，多项式可以直接用向量表示。因此，构造多项式最简单的方法是直接输入向量，如下例所示。

【例 3-72】　直接输入向量构造下面的多项式。

$$f(x)=2x^5+5x^4+4x^2+x+4$$

```
T=[2 5 0 4 1 4];
poly2sym(T)
ans =
    2*x^5+5*x^4+4*x^2+x+4
```

式中，函数 poly2sym 是符号工具箱中的函数（具体内容可参阅第 4 章）。注意，用此种方式构造多项式，必须把多项式各项的系数写完整，而不管此项的系数是否为零。

另外，在 MATLAB 中构造多项式，也可用多项式的根生成方式。

【例 3-73】　用多项式的根构造下面的多项式。

$$f(x)= 2x^5+ 5x^4+ 4x^2+ x+4$$

```
T=[2 5 0 4 1 4];
r=roots(T)
poly(r)
r =
   -2.7709
    0.5611 + 0.7840i
    0.5611 - 0.7840i
   -0.4257 + 0.7716i
   -0.4257 - 0.7716i
ans =
    1.0000    2.5000   -0.0000    2.0000    0.5000    2.0000
```

式中，函数 roots(T)用来求多项式的根，函数 poly(r)用来生成该多项式。

注意，当用根生成多项式时，如果某些根有虚部，由于截断误差的存在，用函数 poly 生成的多项式可能有一些小的虚部。如果要消除这些虚部，只需使用函数 real 提取出实部即可。

3.4.2　多项式的运算

多项式的运算主要包括多项式的四则运算、导数运算、估值运算、求根运算以及多项式的拟合等，下面分别介绍。

1. 多项式的四则运算

多项式的四则运算主要是多项式的加、减、乘、除运算。其中，多项式的加、减运算要求两个相加、减的多项式向量的大小必须相等，此时加、减法才有效。当两个相加、减的多项式阶次不同时，低阶多项式必须用首零填补，使其与高阶多项式有相同的阶次。

【例 3-74】　对下面的两个多项式相加。

$f_1(x)= 2x^5+ 5x^4+ 4x^2+ x+4$，$f_2(x)= 5x^3+ x^2+ 3x+2$。

```
T1=[2 5 0 4 1 4];
T2=[0 0 5 1 3 2];
T=T1+T2;
poly2sym(T)
ans =
   2*x^5+5*x^4+5*x^3+5*x^2+4*x+6
```

多项式的乘法运算用函数 conv(T1,T2)来实现。conv 函数相当于执行两个数组的卷积。

【例 3-75】　对下面的两个多项式执行乘法运算。

$f_1(x)= 2x^5+ 5x^4+ 4x^2+ x+4$，$f_2(x)= 5x^3+ x^2+ 3x+2$。

```
T1=[2 5 0 4 1 4];
T2=[0 0 5 1 3 2];
T=conv(T1,T2);
poly2sym(T)
ans =
```

```
10*x^8+27*x^7+11*x^6+39*x^5+19*x^4+33*x^3+15*x^2+14*x+8
```

需要说明的是，当对多个多项式执行乘法运算时，需要重复使用 conv **函数。**

多项式的除法运算用函数 deconv(T1,T2)**来实现。**deconv **函数相当于执行两个数组的解卷。**

【例 3-76】　对下面的两个多项式执行除法运算。

$f_1(x)= 2x^5+ 5x^4+ 4x^2+ x+4$，$f_2(x)= 5x^3+ x^2+ 3x+2$。

```
T1=[2 5 0 4 1 4];
T2=[5 1 3 2];
[T r]=deconv(T1,T2)
T =
    0.4000    0.9200   -0.4240
r =
         0         0         0    0.8640    0.4320    4.8480
```

式中，T **为多项式相除后的商向量，**r **为多项式除法的余数向量。**

2. **多项式的导数运算**

多项式的导数运算即多项式的微分运算。在 MATLAB **中，多项式的导数运算可用函数** polyder **来实现。**

【例 3-77】　对多项式 $f(x)= 2x^5+ 5x^4+ 4x^2+ x+4$ **进行导数（微分）运算。**

```
T1=[2 5 0 4 1 4];
h=polyder(T1);
poly2sym(h)
ans =
    10*x^4+20*x^3+8*x+1
```

3. **多项式的估值运算**

多项式的估值运算，即求在给定点的多项式函数值。在 MATLAB **中，多项式的估值运算可用函数** polyval **或** polyvalm **来实现。函数** polyval **的调用格式为** polyval(p,s)**，其中，**p **为多项式，**s **为矩阵，它是按数组运算规则来计算多项式的值。函数** polyvalm **的调用格式为** polyval(p,s)**，其中，**p **为多项式，**s **为方阵，它是按矩阵运算规则来计算多项式的值。**

【例 3-78】　求多项式 $f(x)= 2x^5+ 5x^4+ 4x^2+ x+4$ **在给定点** x=[3 4]**处的值。**

```
T1=[2 5 0 4 1 4];
h=polyval(T1,[3 4])
h =
         934        3400
>>T1=[2 5 0 4 1 4];
h=polyvalm(T1,[3 4;4 6])
h =
       43414       62640
       62640       90394
```

注意，此时 s **一定为方阵。**

4. **多项式的求根运算**

求多项式的根实际上已在前面论述到了，即用函数 roots **来求多项式的根。**

【例 3-79】　求方程 $f(x)=2x^5+5x^4+6x^3+4x^2+x+8=0$ 的根。

```
T1=[2 5 6 4 1 8];
h=roots(T1)
h =
   -1.7640
   -0.8679 + 1.3394i
   -0.8679 - 1.3394i
    0.5000 + 0.8001i
    0.5000 - 0.8001i
```

3.4.3　多项式的拟合

在 MATLAB 中，多项式的拟合可用函数 polyfit 来实现。函数 polyfit 的调用格式如下。

- polyfit(x,y,n)　　表示用最小二乘法来对已知数据 x,y 进行拟合，以求得 n 阶多项式的系数向量，输入参数中的 n 即为要拟合的多项式的阶次。
- [p,s] = polyfit(x,y,n)　　输入参数与上面的相同，它返回要拟合的多项式的系数向量 p，向量 s 为使用函数 polyval 获得的错误预估计值。

一般来说，多项式拟合中阶数越大，拟合的精度就越高。

【例 3-80】　用 6 阶多项式对[0,2*pi]上的余弦函数进行拟合。

```
x=linspace(0,2*pi,100);
y=cos(x);
T=polyfit(x,y,6);
y1=polyval(T,x);
plot(x,y,'go',x,y1,'b--')
```

拟合结果如图 3-4 所示。

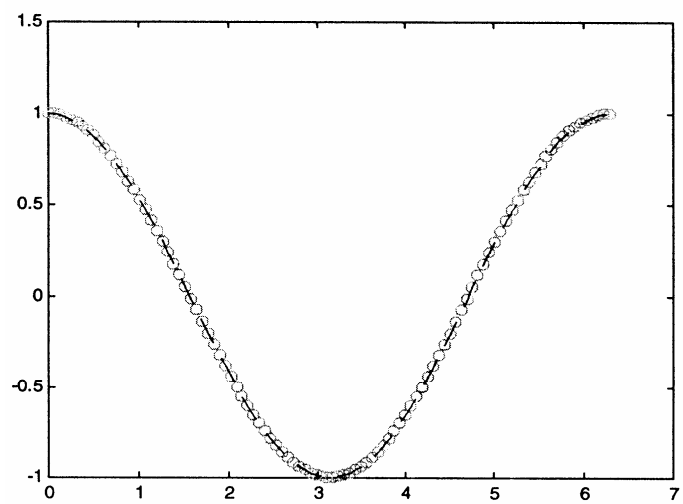


图 3-4　多项式的拟合图

由图 3-4 可知，由多项式拟合生成的图形与原始曲线能很好地吻合，这说明多项式的拟合效果非常好。

3.4.4 多项式的插值

多项式的插值包括一维线性插值、二维线性插值、三维线性插值、三次样条插值等。

1．一维线性插值

对于一维线性数据，可以通过插值或查表来求得离散点之间的数据值。在 MATLAB 中，一维线性插值可以用函数 interp1 来实现。函数 interp1 的调用格式如下。

- Yi=interp1(X, Y, Xi)　　返回在插值向量 Xi 处的函数值向量 Yi，它是根据向量 X 与 Y 插值而来。如果 Y 是一个矩阵，那么对 Y 的每一列进行插值，返回的矩阵 Yi 的大小为 length(Xi)×length(Y,2)。如果向量 Xi 中有元素不在 X 的范围内，则与之相对应的 Yi 返回为 NaN 。
- Yi =interp1(Y, Xi)　　省略 X，表示 X=1:N，此处的 N 为向量 Y 的长度或为矩阵 Y 的行数，即 size(Y,1)，其余同上。
- Yi =interp1(X Y, Xi,'method')　　表示用'method'指定的插值方法进行插值。参数'method'可取以下值：

  'nearest'—— 最近插值

  'linear' —— 线性插值

  'spline' —— 三次样条插值

  'cubic' —— 三次插值

'method'的默认值为线性插值。

需要注意的是，上述的所有插值调用格式，都要求向量 X 为单调。当 X 为单调且等间距时，可以使用快速插值法，此时可以将'method'参数的值设置为'linear', 'cubic'或'nearest'。当 X 为不等间距单调时，可以使用 MATLAB 中的函数 interp1q 来快速插值。

【例 3-81】

```
x=linspace(0,2*pi,10);
y=[5 7 8 10 13 14 15 17 19 20];
x1=[1.2 2.1 3];
y1=interp1(x,y,x1)
```

插值结果为：

```
y1 =
    7.7189    10.0241    13.2972
```

【例 3-82】

```
y=[5 7 8 10 13 14 15 17 19 20];
x1=[1.2 2.1 3];
y1=interp1(y,x1)
```

插值结果为：

```
y1 =
    5.4000    7.1000    8.0000
```

【例 3-83】

```
x=linspace(0,2*pi,100);
y=sin(x);
x1=[0.5 1.4 2.1 2.6 4.2];
```

```
y1=interp1(x,y,x1,'cubic');
plot(x,y,'-',x1,y1,'o')
```

插值结果如图 3-5 所示。

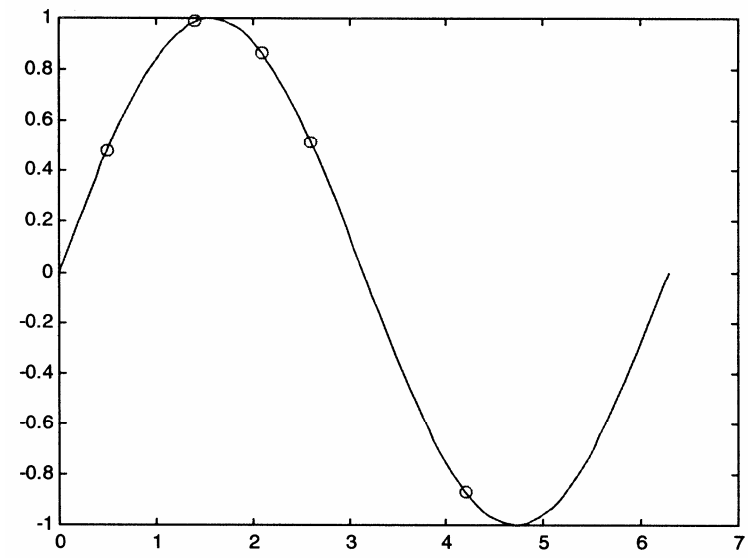


图 3-5　一维线性插值所得的插值曲线

2．二维线性插值

与一维线性插值一样，二维线性插值也可以通过插值或查表来求得离散点之间的数据值。在 MATLAB 中，二维线性插值可以用函数 interp2 来实现。函数 interp2 的调用格式如下。

- Zi =interp2(X, Y, Z, Xi, Yi)　　返回在插值向量 Xi，Yi 处的函数值向量 Zi，它是根据向量 X，Y 与 Z 插值而来，这里 X，Y 与 Z 也可以是矩阵形式。如果向量 Xi，Yi 中有元素不在 X，Y 的范围内，则与之相对应的 Zi 返回为 NaN 。
- Zi =interp2(Z, Xi, Yi)　　省略 X，Y，表示 X=1:N，Y =1:M，此处的 M 与 N 为矩阵 Z 的行数与列数，即[M,N]=size(Z)，其余同上。
- Zi =interp2(Z,ntimes)　　表示在 Z 的各点之间插入数据点来扩展 Z，依次执行 ntimes 次迭代。默认的 ntimes 为 1 次。interp2(Z) 与 interp2(Z,1)的作用一样。
- Zi =interp2(X, Y, Z, Xi, Yi, 'method')　　表示用'method'指定的插值方法进行插值。参数'method'可取以下值：

  'nearest'—— 最近插值

  'linear' —— 线性插值

  'spline' —— 三次样条插值

  'cubic' —— 三次插值

  'method'的默认值为线性插值。

需要注意的是，上述的所有插值调用格式，都要求向量 X 与 Y 为单调且具有相同的格式。当 X 与 Y 为单调且等间距时，可以使用快速插值法，此时可以将'method'参数的值设置为'linear', 'cubic'或'nearest'。

【例 3-84】

```
z=[3 5 7 9 11 10 4 15;1 3 2 6 11 5 7 13];
interp2(z,[1 3],[1 2])
ans =
     3     2
```

【例 3-85】

```
z=[3 5 7 9 11 10 4 15;1 3 2 6 11 5 7 13];
interp2(z,1)
ans =
  Columns 1 through 7
    3.0000    4.0000    5.0000    6.0000    7.0000    8.0000    9.0000
    2.0000    3.0000    4.0000    4.2500    4.5000    6.0000    7.5000
    1.0000    2.0000    3.0000    2.5000    2.0000    4.0000    6.0000
  Columns 8 through 14
   10.0000   11.0000   10.5000   10.0000    7.0000    4.0000    9.5000
    9.2500   11.0000    9.2500    7.5000    6.5000    5.5000    9.7500
    8.5000   11.0000    8.0000    5.0000    6.0000    7.0000   10.0000

  Column 15
   15.0000
   14.0000
   13.0000
```

【例 3-86】　用 method 参数指定的插值方法来进行插值。

```
[x,y,z] = peaks(10);
[xi,yi] = meshgrid(-3:.1:3,  -3:.1:3);
zi = interp2(x,y,z,xi,yi,'spline');
mesh(xi,yi,zi)
```

插值结果如图 3-6 所示。

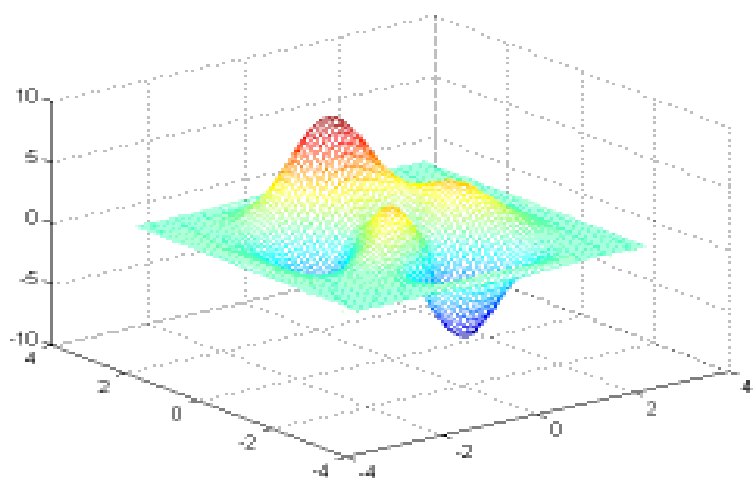


图 3-6　对 peak 函数进行二维线性插值的结果图

3．三维线性插值

与一维线性插值、二维线性插值一样，三维线性插值也可以通过插值或查表来求得离散点之间的数据值。在 MATLAB 中，三维线性插值可以用函数 interp3 来实现。函数 interp3 的调用格式如下。

- Vi = interp3(X,Y,Z,V, Xi, Yi, Zi)　　返回三维函数 V 在插值向量 Xi, Yi, Zi 处的函数值向量 Vi，向量 Xi, Yi, Zi 的大小必须相同。并且，要么全都是行向量形式，要么全都是列向量形式。其中，三维函数 V 由向量 X,Y,Z 插值而来。这里，X,Y,Z 也可以是矩阵形式。如果向量 Xi, Yi, Zi 中有元素不在 X,Y,Z 的范围内，则与之相对应的 Vi 返回 NaN 。
- Vi = interp3(V, Xi, Yi, Zi)　　省略 X,Y,Z，表示 X=1:N，Y =1:M，Z=1:P，此处的 M,N 与 P 为三维函数 V 的大小，即[M,N,P]=size(V)，其余同上。
- Vi = interp3(V,ntimes)　　表示在 V 的各点之间插入数据点来扩展 V，依次执行 ntimes 次迭代。默认的 ntimes 为 1 次。命令 interp3(V) 与 interp3(V,1)的作用一样。
- Vi = interp3(...,'method')　　表示用'method'参数指定的插值方法进行插值。参数'method'可以取以下值：

  'nearest'——最近插值

  'linear'——线性插值

  'spline'——三次样条插值

  'cubic'——三次插值

'method'的默认值为线性插值。

需要注意的是，上述的所有插值调用格式，都要求向量 X,Y, Z 为单调且具有相同的格式。当 X,Y, Z 为单调且等间距时，可以使用快速插值法，此时可以将'method'参数的值设置为'linear', 'cubic'或'nearest'。

【例 3-87】

```
[x,y,z,v] = flow(10);
[xi,yi,zi] = meshgrid(.1:.25:10, -3:.25:3, -3:.25:3);
vi = interp3(x,y,z,v,xi,yi,zi); % V is 31-by-41-by-27
slice(xi,yi,zi,vi,[6 9.5],2,[-2 .2]), shading flat
```

插值结果如图 3-7 所示。

4．三次样条插值

众所周知，使用高阶多项式的插值常常会产生病态的结果，而三次样条插值就能消除这种病态。在三次样条插值中，要寻找 3 次多项式，以逼近每对数据点之间的曲线。因为两点只能决定一条直线，而在两点间的曲线可以用无限多的 3 次多项式进行逼近。因此，为使结果具有惟一性，在三次样条插值中，增加了 3 次多项式的约束条件。通过限定每个 3 次多项式的一阶与二阶导数，使其在插值点处相等，就可以较好地确定所有的 3 次多项式。需要注意的是，第一个与最后一个 3 次多项式在第一个与最后一个插值点外，没有伴随多项式。所以，对于第一个与最后一个插值点，需应用别的约束条件。这里，采用使第一个与第二个 3 次多项式的三阶导数相等，最后一个与最后第二个 3 次多项式的三阶导数也相等作为约束条件。实际上，这也正是 MATLAB 提供的三次样条插值函数 spline 所采用的约束条件。

MATLAB 提供的三次样条插值函数有 spline 与 interp1 两个。其中，函数 interp1 在前面已经有过介绍，下面重点介绍 spline 函数。spline 函数的调用格式如下。

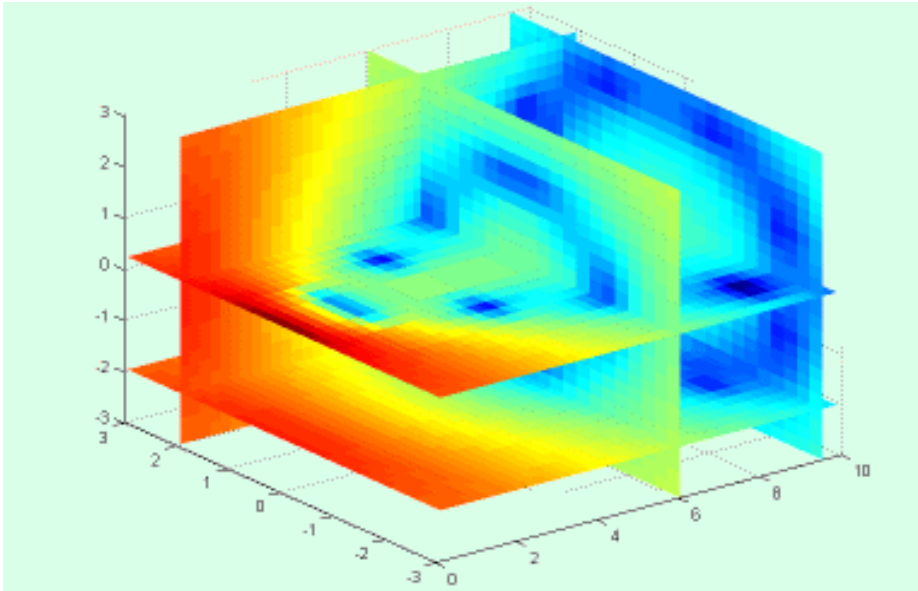


图 3-7　对 flow 函数进行三维线性插值的结果图

- yy=spline(X,Y, xx)　　利用三次样条插值法寻找在插值点 xx 处的插值函数值 yy，插值函数根据输入参数 X 与 Y 的关系得来。其中，X 与 Y 为向量形式，而 xx 可以为向量形式，也可以是标量形式。此函数的作用等同于 interp1(X,Y,xx,'spline')。
- pp=spline(X,Y)　　返回三次样条插值的分段多项式形式的向量，以后可以使用函数 ppval 来进行插值计算。

【例 3-88】

```
circle = spline( 0:4, [0 1 0 -1 0 1 0; pi/2 0 1 0 -1 0 pi/2] );
xx = 0:0.5:4;
cc = ppval(circle, xx)
cc =
   Columns 1 through 7
    1.0000    0.6875         0   -0.6875   -1.0000   -0.6875         0
         0    0.6989    1.0000    0.6837         0   -0.6837   -1.0000
   Columns 8 through 9
    0.6875    1.0000
   -0.6989         0
```

【例 3-89】

```
>>x = 0:10;
y = sin(x);
xx = 0:.25:10;
yy = spline(x,y,xx);
plot(x,y,'o',xx,yy)
```

插值结果如图 3-8 所示。

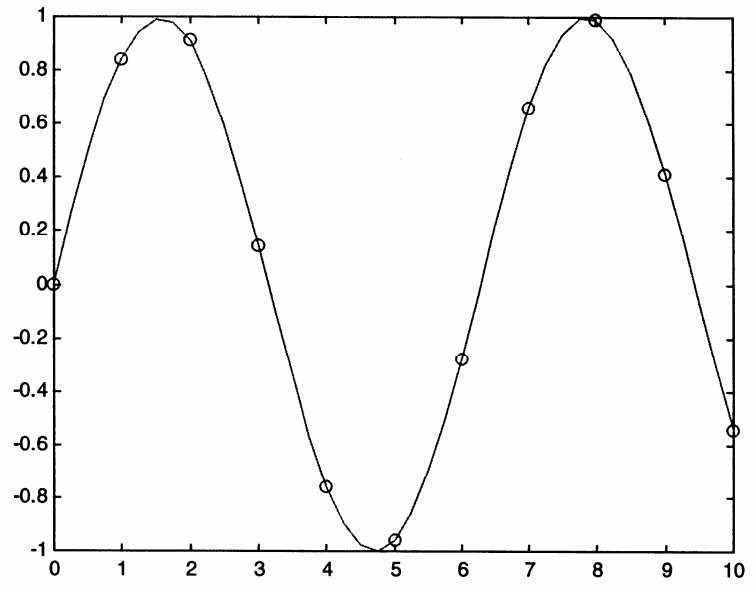


图 3-8　三次样条插值曲线图

3.5　关系和逻辑运算

MATLAB 除了支持上面所述的数学运算外，还支持关系和逻辑运算。关系和逻辑运算主要是提供求解真（假）命题的答案。对于所有输入的关系与逻辑表达式，MATLAB 与一般的编程语言相类似，它把所有非零数值当成真，把零当成假。对于所有输出的关系与逻辑表达式，MATLAB 是这样处理的：对于真值，输出为 1；对于假值，输出为 0。

关系与逻辑运算主要包括关系与逻辑操作符、关系与逻辑函数以及测试函数等，下面分别介绍。

3.5.1　关系与逻辑操作符

1. 关系操作符

MATLAB 中的关系操作符如表 3-3 所示。

表 3-3　关系操作符

关系操作符	说　　明	相对应的函数
= =	等于	eq(A,B)
~ =	不等于	ne(A,B)
<	小于	lt(A,B)
>	大于	gt(A,B)
<=	小于等于	le(A,B)
>=	大于等于	ge(A,B)

表 3-3 中，函数中的 A 与 B 可以都是矩阵或数组。此时，要求它们的大小完全相同。关系操作符对矩阵或数组中各自相对应的元素进行比较，返回的结果与两相比较的矩阵或数组

的大小相同。A 与 B 也可以是：一个为数组或矩阵，另一个为标量。此时，标量与数组或矩阵中的每一个元素进行比较，返回的结果与数组或矩阵的大小相同。

【例 3-90】

```
A=[1 4 3 5 7];
B=[2 6 9 0 7];
C=eq(A,B)
C =
     0     0     0     0     1
```

【例 3-91】

```
A=[1 4 3 5 7];
B=[2 6 9 0 7];
A==B
ans =
     0     0     0     0     1
```

由于关系运算的输出为 0 或 1，所以它们也能用在数学运算中。请看下例。

【例 3-92】

```
A=[1 4 3 5 7];
B=[2 6 9 0 7];
C=eq(A,B);
D=A-C
D =
     1     4     3     5     6
```

2. 逻辑操作符

MATLAB 中的逻辑操作符如表 3-4 所示。

表 3-4　逻辑操作符

逻辑操作符	说　　明	相对应的函数
&	逻辑与	and(A,B)
|	逻辑或	or(A,B)
~	逻辑非	not(A)
	逻辑异或。A 与 B 都是非零或都是零返回 0；有一个非零返回 1	xor(A,B)
	向量 A 中有非零元素时返回 1；矩阵 A 中某一列有非零元素时，此列返回 1	any(A)
	向量 A 中所有元素非零时返回 1；矩阵 A 中某一列所有元素非零时，此列返回 1	all(A)

表 3-4 中，函数中的 A 与 B 可以都是矩阵或向量。此时，要求它们的大小完全相同。逻辑操作符对矩阵或向量中各自相对应的元素进行比较，返回的结果与两相比较的矩阵或向量的大小相同。A 与 B 也可以是：一个为向量或矩阵，另一个为标量。此时，标量与向量或矩阵中的每一个元素进行比较，返回的结果与向量或矩阵的大小相同。

【例 3-93】

```
A=[0 0 2 1];
B=[1 2 0 1];
c=xor(A,B)
```

```
c =
     1     1     1     0
```

【例 3-94】

```
A=[0 0 2 1];
B=1;
c=or(A,B)
c =
     1     1     1     1
```

3.5.2 测试函数

测试函数主要用于测试特殊值或条件的存在，返回的是逻辑值 1 或 0。在 MATLAB 中，测试函数如表 3-5 所示。

表 3-5 测试函数

函 数 名	说 明
isempty(A)	若参量 A 为空，返回 1；否则，返回 0
isglobal(A)	若参量 A 为全局变量，返回 1；否则，返回 0
ishold	当前绘图状态保持是 ON，返回 1；否则，返回 0
isieee	计算机执行 IEEE 运算，返回 1；否则，返回 0
isinf(A)	若参量 A 无穷大，返回 1；否则，返回 0
isletter(A)	若参量 A 为字母，返回 1；否则，返回 0
isnan(A)	若参量 A 为不定值，返回 1；否则，返回 0
isreal(A)	若参量 A 无虚部，返回 1；否则，返回 0
isspace(A)	若参量 A 为空格字符，返回 1；否则，返回 0
isstr(A)或 ischar(A)	若参量 A 为一个字符串，返回 1；否则，返回 0
isstudent(A)	若 MATLAB 为学生版，返回 1；否则，返回 0
isunix(A)	若计算机为 UNIX 系统，返回 1；否则，返回 0
isvms(A)	若计算机为 VMS 系统，返回 1；否则，返回 0

【例 3-95】

```
isstr('aa')
ans =
     1
```

【例 3-96】

```
A=[];
isempty(A)
ans =
     1
```

3.6 数据分析

MATLAB 强大的矩阵运算功能，决定了它很容易对一大批数据进行一般的数据分析，比

如求平均值、最大值、标准差等。下面对在 MATLAB 中出现的通用数据分析函数进行简单的介绍。

3.6.1 基本数据操作函数

MATLAB 中出现的通用数据分析函数如表 3-6 所示。

表 3-6 通用的数据分析函数

函 数 名	说 明
max	求最大元素
min	求最小元素
mean	求元素平均值或列元素的平均值
median	求列元素的中值
std	求元素标准差
sum	求各列元素的和
prod	求列元素的积
hist	直方图
trapz	梯形法求数值积分
cumprod	求列元素的累计积
cumsum	求列元素的累计和
cumtrapz	梯形法求累积数值积分
sort	按升序对元素进行排序
sortrows	按升序排列矩阵各行

下面，对几个常用的通用数据分析函数做简单的介绍。

1. max 函数

max 函数用于求向量或矩阵中的最大元素。在 MATLAB 中，max 函数的调用格式有如下几种。

- max(X)　　输入参数 X 可以为向量或矩阵。若为向量，则返回最大的向量元素。若为矩阵，则返回一行向量，包括矩阵每列的最大元素。
- [Y,I]=max(X)　　同上，只是同时返回最大元素在向量中的下标值或最大元素在矩阵各列中的下标值。
- max(X,Y)　　要求向量或矩阵 X 与 Y 的大小一样，返回的是一个包含最大元素的、与它们大小一样的矩阵或向量。另外，X 或 Y 也可以有一个是标量，此时那个标量与另一向量或矩阵中的每个元素进行比较，返回的是与那个向量或矩阵大小一样的向量或矩阵。
- [Y,I] = max(X,[],DIM)　　返回向量或矩阵 X 中维数为 DIM 的最大值及其下标值。

当 X 为复数形式时，通过计算 max(ABS(X))返回结果。另外，在进行最大值计算时，NaNs（即 MATLAB 中的不定值表示形式）被忽略。

【例 3-97】

```
a=[1 2 3;4 5 3];
[y,i]=max(a)
```

```
y =
     4     5     3
i =
     2     2     1
```

【例 3-98】

```
a=[1 2 3;4 5 3];
b=[1 2 4;4 3 6];
max(a,b)
ans =
     1     2     4
     4     5     6
```

【例 3-99】

```
a=[1 2 3;4 5 3];
[y,i]=max(a,[],2)
y =
     3
     5
i =
     3
     2
```

2．min 函数

min 函数用于求向量或矩阵中的最小元素。在 MATLAB 中，min 函数的调用格式与 max 函数完全一样，这里就不再赘述。

【例 3-100】

```
a=[1 2 3;4 5 3];
b=4;
min(a,b)
ans =
     1     2     3
     4     4     3
```

3．mean 函数

mean 函数用于求向量或矩阵中元素的平均值。在 MATLAB 中，mean 函数的调用格式如下。

- mean (X)　　输入参数 X 可为向量或矩阵。若为向量，则返回向量元素的平均值。若为矩阵，则返回一行向量，包括矩阵每列元素的平均值。
- mean(X,DIM)　　返回向量 X 中维数为 DIM 的平均值。

【例 3-101】

```
a=[1 2 3;4 5 3];
mean(a,2)
ans =
     2
```

4

【例 3-102】

```
a=[1 2 3;4 5 3];
mean(a,1)
ans =
    2.5000    3.5000    3.0000
```

4．median 函数

median 函数用于求向量或矩阵中元素的中值。median 函数的调用格式与 mean 函数完全一样，这里就不再赘述。

【例 3-103】

```
a=[1 2 3;4 5 3];
median(a)
ans =
    2.5000    3.5000    3.0000
```

【例 3-104】

```
a=[1 2 3;4 5 3];
median(a,2)
ans =
     2
     4
```

5．std 函数

std 函数用于求向量或矩阵中元素的标准差。在 MATLAB 中，std 函数的调用格式如下。

- std (X)或 std(X,0)　　输入参数 X 可为向量或矩阵。若为向量，则返回向量元素的标准差。若为矩阵，则返回一行向量，包括矩阵每列元素的标准差。其中，标准差用 $n-1$ 正态化。
- std(X,1)　　输入参数 X 可为向量或矩阵。若为向量，则返回向量元素的标准差。若为矩阵，则返回一行向量，包括矩阵每列元素的标准差。其中，标准差用 n 正态化。
- std(X,Flag,DIM)　　输入参数 X 可为向量或矩阵。返回向量中维数为 DIM 的标准差，其中，当 Flag=0 时，标准差用 $n-1$ 正态化；当 Flag 不等于 0 时，标准差用 n 正态化。

【例 3-105】

```
a=[1 2 3;4 5 3];
std(a,0,2)
ans =
     1
     1
```

【例 3-106】

```
a=[1 2 3;4 5 3];
std(a,1)
ans =
    1.5000    1.5000         0
```

6．sum 函数

sum 函数用于求向量或矩阵中元素的和。在 MATLAB 中，sum 函数的调用格式如下。

- sum (X)　　输入参数 X 可为向量或矩阵。若为向量，则返回向量元素的总和。若为矩阵，则返回一行向量，包括矩阵每列元素的和。
- Sum (X,DIM)　返回向量 X 中维数为 DIM 的元素和。

【例 3-107】

```
x=[0 1 2;3 4 5];
sum(x,1)
ans =
     3     5     7
```

【例 3-108】

```
x=[0 1 2;3 4 5];
sum(x)
ans =
     3     5     7
```

【例 3-109】

```
x=[0 1 2;3 4 5];
sum(x,2)
ans =
     3
     12
```

7．prod 函数

prod 函数用于求向量或矩阵中元素的积。prod 函数的调用格式与 sum 函数完全一样，这里就不再赘述。

【例 3-110】

```
x=[0 1 2;3 4 5];
y=prod(x,2)
y =
     0
    60
```

【例 3-111】

```
x=[0 1 2;3 4 5];
y=prod(x)
y =
     0     4    10
```

8．trapz 函数

trapz 函数用于梯形法求数值积分。在 MATLAB 中，trapz 函数的调用格式如下。

- Z=trapz(Y)　　通过梯形法求 Y 的近似积分值。若 Y 为向量，则返回 Y 的近似积分值；若 Y 为矩阵，则返回一行向量，此行向量各元素为矩阵 Y 对应列向量的近似积分值。
- Z=trapz(X，Y)　　通过梯形法计算 Y 对 X 的积分值。需要注意的是，向量 X 和 Y 必须长度相等；或者，X 为一列向量，而 Y 为第一个非独立维是 X 的长度值的数组，此

时函数沿此维开始计算。

- Z=trapz(X,Y,DIM)或 trapz(Y,DIM)　表示对 Y 的交叉维 DIM 积分。X 的长度必须等于 size(Y,DIM)。

【例 3-112】

```
Y=[0 1 2;3 4 5];
trapz(Y,1)
ans =
    1.5000    2.5000    3.5000
```

【例 3-113】

```
Y=[0 1 2;3 4 5];
trapz(Y,2)
ans =
     2
     8
```

【例 3-114】

```
Y=[0 1 2;3 4 5];
trapz(Y)
ans =
    1.5000    2.5000    3.5000
```

9. sort 函数

sort 函数用于对元素按升序进行排序。在 MATLAB 中，sort 函数的调用格式如下。

- sort (X)　输入参数 X 可为向量、矩阵或字符串。若为向量，则返回对向量中的元素按升序进行排序后的结果；若为矩阵，则返回一矩阵，该矩阵的每列元素是按升序进行排序后的结果；若为字符串，则返回对 X 中的字母按 ASCII 先后顺序进行排序后的结果。
- [Y,I] = sort(X)　同上，但它同时返回排序后各元素的下标值。如果 X 是一个向量，则 Y = X (I)；如果 X 是一个 $m \times n$ 大小的矩阵，则当 $j = 1:n$ 时，Y(:,j) = X (I(:,j),j)。
- sort (X,DIM)　返回对向量 X 中维数为 DIM 的元素按升序进行排序后的结果。

【例 3-115】

```
x=[3 7 5;0 4 2];
sort(x,1)
ans =
    0    4    2
    3    7    5
```

【例 3-116】

```
x=[3 7 5;0 4 2];
sort(x,2)
ans =
     3    5    7
     0    2    4
```

【例 3-117】

```
x=[3 7 5;0 4 2];
sort(x)
ans =
     0     4     2
     3     7     5
```

3.6.2 有限差分类函数

MATLAB 中出现的有限差分类函数如表 3-7 所示。

表 3-7 有限差分类函数

函 数 名	说 明
diff	求差分和近似导数
gradient	求近似梯度
del2	求离散 Laplace 算子

下面，对几个常用的有限差分类函数做简单的介绍。

1．diff 函数

diff 函数用于求差分和近似导数。diff 函数的调用格式如下。

- diff(X)　　输入参数 X 可为向量或矩阵。若 X 为向量，则返回的是[X (2) - X (1), X (3) - X (2),…, X (n) - X (n-1)]；若 X 为矩阵，则返回的是矩阵每列的差分，即[X (2:n,:) - X (1:n-1,:)]。
- diff(X,N)　　返回的是向量或矩阵中第一个非独立维的 N 阶差分或导数值。若 N >= size(X,DIM)，则 diff 返回下一个非独立维的连续差分或导数值。
- diff(X,N,DIM)　　返回的是向量或矩阵中第 DIM 维的 N 阶差分或导数值。若 N >= size(X,DIM)，则 diff 返回一空向量。

【例 3-118】

```
x=[3 7 5;0 4 2];
diff(x)
ans =
    -3    -3    -3
```

【例 3-119】

```
x=[3 7 5;0 4 2];
diff(x,2)
ans =
     0     0
```

【例 3-120】

```
x=[3 7 5;0 4 2];
diff(x,2,1)
ans =
empty matrix: 0-by-3
```

2．gradient 函数

gradient 函数用于求近似梯度。gradient 函数的调用格式如下。

- [FX,FY] = gradient (F)　　返回矩阵 F 的数值梯度。其中，FX 相当于 d*F* /d*x*，为 X 方向的差分值。FY 相当于 d*F*/d*y*，为 Y 方向的差分值。每个方向的点间隔假设为 1。当 F 为向量时，DF =gradient (F)为一维梯度形式。
- [FX,FY] = gradient (F,H)　　当 H 为标量时，使用 H 作为各个方向的点间隔值，其他同上。
- [FX,FY] = gradient (F,HX,HY)　　当 F 为二维时，使用 HX 与 HY 作为 X 和 Y 方向的点间隔。HX 与 HY 可为标量或向量。若 HX 与 HY 为向量，则要求它们的长度必须与 F 的长度相匹配。
- [FX,FY,FZ] = gradient (F)　　返回矩阵 F 的三维数值梯度。其中，FZ 相当于 d*F* /d*z*，为 Z 方向的差分值。同理，gradient(F,H)当 H 为标量时，使用 H 作为各个方向的点间隔值。
- [FX,FY,FZ] = gradient (F, HX,HY,HZ)　　使用 HX,HY,HZ 作为各个方向的点间隔值。

【例 3-121】

```
[x,y] = meshgrid(-2:.2:2, -2:.2:2);
z = x .* log(-x.^2 - y.^2);
[px,py] = gradient(z,.2,.2);
contour(z),
hold on,
quiver(px,py),
hold off
```

结果如图 3-9 所示。

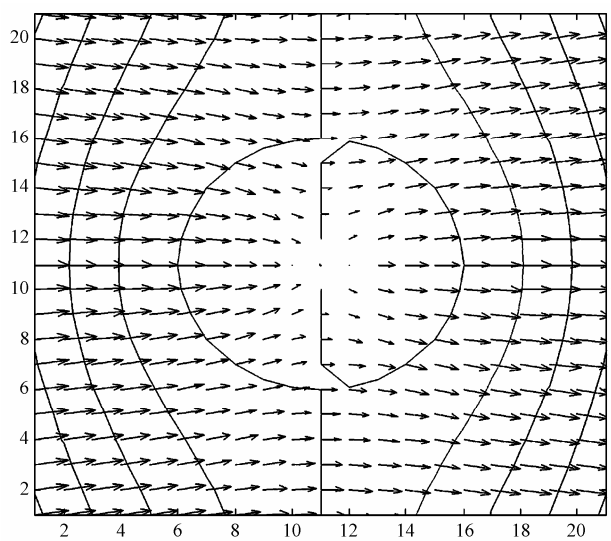


图 3-9　梯度函数矢量图

3.6.3 相关关系类函数

MATLAB 中出现的相关关系类函数如表 3-8 所示。

表 3-8 相关关系类函数

函 数 名	说 明
corrcoef	求相关系数
cov	求协方差矩阵
subspace	求 2 个子空间之间的夹角

下面，对几个常用的相关关系类函数做简单的介绍。

1．corrcoef 函数

corrcoef 函数用于求相关系数，corrcoef 函数的调用格式如下。

- corrcoef(X)　返回从矩阵 X 形成的一个相关系数矩阵。此相关系数矩阵的大小与矩阵 X 一样。它把矩阵 X 的每行作为一个观测量，每列作为一个变量，然后求它们的相关系数。
- corrcoef (X,Y)　在这里，X 与 Y 是一个列向量，它们与 corrcoef ([X Y])的作用一样。

如果 C 是一个协方差矩阵，即 C = cov(X)，则 corrcoef (C) 是这样一个矩阵，它的第（i,j）个元素由下式求得：C(i,j)/SQRT(C (i,i)* C (j,j))。

【例 3-122】

```
x=[1 2 3;4 5 6;7 8 9];
y=corrcoef(x)
y =
     1     1     1
     1     1     1
     1     1     1
```

【例 3-123】

```
x=[1 2 3;4 5 6;7 10 9];
y=cov(x);
z=corrcoef(y)
z =
     1     1     1
     1     1     1
     1     1     1
```

2．cov 函数

cov 函数用于求协方差矩阵，cov 函数的调用格式如下。

- cov(X)或 cov(X,0)　输入参数 X 可为向量或矩阵。若 X 为向量，则返回的是一个标量值；若 X 为矩阵，则返回的协方差矩阵的大小与矩阵 X 一样。它把矩阵 X 的每行作为一个观测量，每列作为一个变量，然后求它们的协方差。协方差用 $n-1$ 正态化。
- cov (X,1)　除了协方差用 n 正态化外，其他同上。
- cov (X,Y)或 cov (X,Y,0)　在这里，X 与 Y 是两个同样长度的向量。它们等同于 cov([X (:) Y(:)])。其中，协方差用 $n-1$ 正态化。

- cov (X,Y,1)　**除了协方差用 *n* 正态化外，其他同上。**

【例 3-124】

```
x=[2 3 4 5 6 7];
y=cov(x)
y =
    3.5000
```

【例 3-125】

```
x=[2 3 4;5 6 7];
y=cov(x)
y =
    4.5000    4.5000    4.5000
    4.5000    4.5000    4.5000
    4.5000    4.5000    4.5000
```

第4章　符 号 运 算

MATLAB 的强大之处不仅在于其强大的数值运算功能，而且也在于其强大的符号运算功能。MATLAB 的符号运算是通过集成在 MATLAB 中的符号数学工具箱（Symbolic Math Toolbox）来实现的。实际上，MATLAB 中的符号数学工具箱是建立在功能强大的由加拿大滑铁卢大学（Waterloo University）开发的 Maple 软件的基础上。当进行 MATLAB 符号运算时，它就请求 Maple 软件去计算并将结果返回给 MATLAB。MATLAB 的符号数学工具箱用途广泛，它可用于数学、物理、力学等各种学科的科研、工程应用中。更为重要的是，它使用字符串来进行符号分析与运算，而不是基于矩阵的数值分析与运算。

MATLAB 的符号数学工具箱可完成几乎所有的符号运算功能。这些功能主要包括：符号表达式的运算，符号表达式的复合、化简，符号矩阵的运算，符号微积分，符号函数画图，符号代数方程求解，符号微分方程求解等。此外，符号数学工具箱还支持可变精度运算，即支持符号运算并以指定的精度返回结果。本章将对上面所述内容做详细的介绍。

4.1　符号表达式

在 MATLAB 符号工具箱中，符号表达式是代表数字、函数和变量的 MATLAB 字符串或字符串数组，它不要求变量要有预先确定的值。

符号表达式包括符号表达式的生成与符号表达式的运算。

在 MATLAB 符号工具箱中，符号表达式的运算主要是通过符号函数对其进行运算的。所有的符号函数作用到符号表达式和符号数组，返回的仍是符号表达式或符号数组（即字符串）。可以运用 MATLAB 中的函数 isstr 来判断返回的表达式是数字还是字符串。如果是字符串，isstr 函数返回 1，否则，返回 0。符号表达式的运算主要包括：提取分子、分母，符号表达式的基本代数运算，符号表达式的高级运算，符号数值函数的创建等。

4.1.1　符号表达式的生成

符号表达式可以是符号函数或符号方程。其中，符号函数没有等号，而符号方程必须要带有等号。MATLAB 在内部把符号表达式表示成字符串，以与数字相区别。符号表达式的创建可通过以下几种方法进行。

1. 用单引号来生成符号表达式

在 MATLAB 中，所有的字符串都用单引号来设定输入或输出。为此，符号表达式也可用单引号来生成。

【例 4-1】

```
f='exp(x)'
f =
    exp(x)
```

【例 4-2】

```
f='a*x^2+bx+c=0'
f =
    a*x^2+bx+c=0
```

【例 4-3】

```
f='D2y-2Dy-3y=0'
f =
    D2y-2Dy-3y=0
```

上面的第一个例子生成的是一般的符号函数，第二个例子生成的是符号代数方程，第三个例子生成的是符号微分方程。

2. **用函数** sym **来生成符号表达式**

在 MATLAB **自己可以确定变量类型的情况下，可以不用** sym **函数来显式生成的符号表达式。但在某些情况下，特别是在建立符号数组时，必须要用** sym **函数来将字符串转换成符号表达式。**

【例 4-4】

```
A=sym('[a b c;e f g]')
A =
    [ a, b, c]
    [ e, f, g]
```

【例 4-5】

```
f=sym('ax+b=0')
f =
    ax+b=0
```

3. **用函数** syms **来生成符号表达式**

用函数 syms **只能生成符号表达式，而不能生成符号方程。**

【例 4-6】

```
syms y u;
p = exp(-y/u)
p =
    exp(-y/u)
```

在符号表达式中，要了解哪些变量是符号变量，可用函数 symvar **来获知。有意义的是，函数** symvar **会自动把**'i'，'j'，'pi'，'inf'，'nan'，'eps' **等特殊字母不当成符号变量。**

【例 4-7】

```
symvar('sin(omega)')
ans =
    'omega'
```

【例 4-8】

```
symvar('a*x+y')
ans =
    'a'
    'x'
    'y'
```

当利用符号函数进行运算时，若没有指定独立变量，则 MATLAB 会把 x 当成独立变量。

【例 4-9】

```
diff('x^n')
ans =
    x^n*n/x
```

上例表示，以 x 作为默认的独立变量，对 x 求导。

【例 4-10】

```
diff('x^n','n')
ans =
    x^n*log(x)
```

上例表示，以 n 作为默认的独立变量，对 n 求导。

4.1.2 符号表达式的提取分子、分母运算

如果符号表达式为有理分式形式或可展开为有理分式的形式，则可通过函数 numden 来提取符号表达式中的分子与分母。numden 函数可将符号表达式合并、有理化，并返回所得的分子与分母。numden 函数的调用格式如下。

- [n,d] =numden(a)　提取符号表达式 a 中的分子与分母，并分别将其存放在 n 与 d 中。
- n =numden(a)　提取符号表达式 a 的分子与分母，但只把分子存放在 n 中。

【例 4-11】

```
 f=sym('a*x^2/(b-x)');
[n,d]=numden(f)
n =
    -a*x^2
d =
    -b+x
```

【例 4-12】

```
f=sym('a*x^2/(b-x)');
n=numden(f)
n =
    -a*x^2
```

4.1.3 符号表达式的基本代数运算

符号表达式的加、减、乘、除四则运算及幂运算等基本的代数运算，与矩阵的数值运算几乎完全一样。其中，符号表达式的加、减、乘、除运算可分别由函数 symadd, symsub, symmul, symdiv 来实现，也可与矩阵的数值运算一样，用“+”，“-”，“*”，“/”符号进行运算，而符号表达式的幂运算可以由函数 sympow 来实现，也可以由幂运算符号“^”来实现。

【例 4-13】

```
f='4*x+5';
g='2*x^2+5*x+6';
symadd(f,g)
```

```
ans =
    9*x+11+2*x^2
```

【例 4-14】

```
B=sym('x+1');
C=sym('x^2-1');
D=B+C
D =
    x+x^2
```

【例 4-15】

```
f='4*x+5+6*y';
g='2*x^2+5*x+6';
symsub(f,g)                    %求 f-g 的表达式
ans =
    -x-1+6*y-2*x^2
```

【例 4-16】

```
f='4*x+5+6*y';
g='2*x^2+5*x+6';
symmul(f,g)                    %求 f*g 的表达式
ans =
    (4*x+5+6*y)*(2*x^2+5*x+6)
```

【例 4-17】

```
f='4*x+5+6*y';
g='2*x^2+5*x+6';
symdiv(f,g)                    %求 f/g 的表达式
ans =
    (4*x+5+6*y)/(2*x^2+5*x+6)
```

【例 4-18】

```
f='4*x+5+6*y';
g='2*x^2+5*x+6';
sympow(f,g)                    %求 f^g 的表达式
ans =
    (4*x+5+6*y)^(2*x^2+5*x+6)
```

【例 4-19】

```
B=sym('x+1');
C=sym('x^2-1');
D=B^C
D =
    (x+1)^(x^2-1)
```

4.1.4 符号表达式的高级运算

符号表达式的高级运算主要是指符号表达式的复合函数运算、反函数运算以及求表达式

的符号和，下面分别予以介绍。

1. 符号表达式的复合函数运算

在 MATLAB 中，符号表达式的复合函数运算主要是通过函数 compose 来实现的。compose 函数的调用格式如下。

- compose(f,g)　返回复合函数 $f(g(y))$。在这里，$f = f(x)$，$g = g(y)$。其中，x 是 findsym 定义的 f 函数的符号变量，y 是 findsym 定义的 g 函数的符号变量。
- compose (f,g,z)　返回自变量为 z 的复合函数 $f(g(z))$。在这里，$f = f(x)$, $g = g(y)$，x, y 分别是 findsym 定义的 f 函数和 g 函数的符号变量。
- compose (f,g,x,z)　返回复合函数 $f(g(z))$，并且使 x 成为 f 函数的独立变量。即，如果 $f = \cos(x/t)$, 则 compose (f,g,x,z)返回 $\cos(g(z)/t)$ ，而 compose (f,g,t,z) 返回 $\cos(x/g(z))$。
- compose (f,g,x,y,z)　返回复合函数 $f(g(z))$，并且使 x 与 y 分别成为函数 f 与 g 的独立变量。即，如果 $f = \cos(x/t)$，$g = \sin(y/u)$, compose (f,g,x,y,z) 返回 $\cos(\sin(z/u)/t)$，而 compose (f,g,x,u,z)返回 $\cos(\sin(y/z)/t)$。

【例 4-20】

```
syms x y;
f = 1/x^3;
g = tan(y);
compose(f,g)
ans =
    1/tan(y)^3
```

【例 4-21】

```
syms x y t;
f = 1/x^3;
g = tan(y);
compose(f,g,t)
ans =
    1/tan(t)^3
```

【例 4-22】

```
syms x y t z;
g = tan(y);
h=x^t;
compose(h,g,x,z)
ans =
    tan(z)^t
```

【例 4-23】

```
syms x y t z;
g = tan(y);
h=x^t;
compose(h,g,x,y,z)
ans =
    tan(z)^t
```

2. **符号表达式的反函数运算**

在 MATLAB **中，符号表达式的反函数运算主要是通过函数** finverse **来实现的。**finverse **函数的调用格式如下。**

- g = finverse (f)　**返回符号函数** f **的反函数** g**。其中，**f **是一个符号函数表达式，其变量为** x**。求得的反函数** g **是一个满足** $g(f(x)) = x$ **的符号函数。**
- g = finverse (f,v)　**返回自变量为** v **的符号函数** f **的反函数。求得的反函数** g **是一个满足** $g(f(v)) = v$ **的符号函数。当** f **包含不止一个符号变量时，往往使用这种求反函数的调用格式。**

需要注意的是，当函数 finverse **求得的解不惟一时，**MATLAB **会给出警告信息。**

【例 4-24】

```
f=sym(1/sin(x));
finverse(f)
ans =
    asin(1/x)
```

【例 4-25】

```
f=sym(x^2+1);
finverse(f)
Warning: finverse(x^2+1) is not unique.
> In E:\MATLAB\toolbox\symbolic\@sym\finverse.m at line 43
ans =
    (-1+x)^(1/2)
```

上例就是因为求得的解不惟一而给出了警告。

3. **求表达式的符号和**

在 MATLAB **中，求表达式的符号和主要是通过函数** symsum **来实现的。**symsum **函数的调用格式如下。**

symsum (S)　**返回** $\sum_{0}^{x-1} s(x)$ **的结果。**

symsum (S,v)　**返回** $\sum_{0}^{x-1} s(v)$ **的结果。**

symsum (S,a,b)　**返回** $\sum_{a}^{b} s(x)$ **的结果。**

symsum (S,v,a,b)　**返回** $\sum_{a}^{b} s(v)$ **的结果。**

【例 4-26】

```
k=sym('k');
symsum(k)
ans =
    1/2*k^2-1/2*k
```

【例 4-27】

```
k=sym('k');
```

```
n=sym('n');
symsum(k,0,n-1)
ans =
    1/2*n^2-1/2*n
```

【例 4-28】

```
k=sym('k');
symsum(1/k^2,1,inf)
ans =
    1/6*pi^2
```

4.1.5 符号数值函数的创建

可以通过两种方法来创建符号数值函数，第一种方法是使用符号表达式来创建，另一种方法是通过编写一个 M **文件来创建。**

1. **符号表达式创建符号函数**

【例 4-29】

```
syms x y;
f=sin(x*y)/(x*y)
f=
    sin(x*y)/x/y
```

对于产生的符号数值函数 f**，可以用符号工具箱中的** diff, int, subs **等符号函数来进行其他的运算。**

2. M **文件创建符号函数**

通过 M **文件，可以创建更通用的符号函数。**

【例 4-30】　创建能对任何输入参数进行处理的符号函数 $z=\sin(x)/x$**。**

permit a more general use of functions. Suppose, for example, you want to create the sinc function sin(x)/x. To do this, create an M-file in the @sym directory:

```
function z = sinc(x)
%SINC 符号函数名；sin(x)/x 是符号函数
if isequal(x,sym(0))
    z = 1;
else
    z = sin(x)/x;
end
```

对于 M **文件的详细说明请参阅第** 7 **章。**

4.2 符号与数值间的转换及符号的可变精度运算

在 MATLAB **中，有许多函数可实现将符号表达式转换成数值表达式或将数值表达式转换成符号表达式。其中，将符号表达式转换成数值表达式可以通过函数** numeric **或** eval **来实现；而将数值表达式转换成符号表达式可以通过函数** sym **来实现。另外，函数** sym2poly **实现将符号多项式转换成它的** MATLAB **等价系数向量；而函数** poly2sym **的功能正好相反，实**

现将 MATLAB 等价系数向量转换成符号多项式。MATLAB 的符号数学工具箱还支持可变精度运算，即支持符号运算并以指定的精度返回结果，这可通过函数 digits 与 vpa 来实现。下面对上述几个函数做简单的介绍。

4.2.1 将符号表达式转换成数值表达式

将符号表达式转换成数值表达式主要是通过函数 numeric 或 eval 来实现。

【例 4-31】

```
p='1+sqrt(2)/2'
p =
    1+sqrt(2)/2
```

【例 4-32】

```
numeric(p)
ans =
    1.7071
```

【例 4-33】

```
eval(p)
ans =
    1.7071
```

另外，函数 sym2poly 可实现将符号多项式转换成它的 MATLAB 等价系数向量。

【例 4-34】

```
sym2poly(3*x^4+4*x^3+2*x^2+x+1)
ans =
     3     4     2     1     1
```

4.2.2 将数值表达式转换成符号表达式

将数值表达式转换成符号表达式主要通过函数 sym 来实现。

【例 4-35】

```
p=1.7071;
n=sym(p)
n =
    17071/10000
```

另外，函数 poly2sym 可实现将 MATLAB 等价系数向量转换成它的符号多项式。

【例 4-36】

```
a=[1 2 3 4 5 1];
poly2sym(a)
ans =
    x^5+2*x^4+3*x^3+4*x^2+5*x+1
```

4.2.3 可变精度运算

因为数值的精度受每次操作所保留的位数限制，所以，数值的任何运算都会引入误差。

而符号表达式的运算由于不涉及到数值运算，因此，它的运算结果是很精确的。MATLAB 对数值的运算完全依靠计算机的浮点运算，一般地讲，MATLAB 数值运算的相对精度大约是 16 位。而 MATLAB 的符号数学工具箱可实现任何位数的运算。符号数学工具箱的默认位数为 16 位，其默认位数可以由函数 digits 与 vpa 来调整。其中，digits 函数的调用会对所有的符号表达式都有影响，而 vpa 函数的调用只对单个的符号表达式有影响。

【例 4-37】

```
digits    %查看现在系统中的算术运算精度
digits = 25
```

【例 4-38】

```
vpa('pi')
ans =
    3.14159265358979323846264
```

【例 4-39】

```
digits(15);
vpa('pi')
ans =
    3.14159265358979
```

【例 4-40】

```
vpa('pi',50)
ans =
    3.1415926535897932384626433832795028841971693993751
```

4.3 符号表达式的化简与替换

当通过 MATLAB 的符号函数运算生成的符号表达式难于看懂时，可以通过 MATLAB 的符号数学工具箱中提供的函数，来对符号表达式进行化简与替换，使其成为易于看懂的形式。

4.3.1 符号表达式的化简

MATLAB 的符号数学工具箱中提供的符号表达式化简函数主要有：pretty, collect, horner, factor, expand, simple, simplify。下面对上述函数分别予以介绍。

- pretty　函数 pretty 与高等数学课本上显示符号表达式的形式相类似。

【例 4-41】

```
syms x;
f=taylor(exp(-x))    %显示指数函数的 Taylor 多项式
f =
    1-x+1/2*x^2-1/6*x^3+1/24*x^4-1/120*x^5
pretty(f)
    1 – x^2 + 1/2 x^2 - 1/6 x^3 + 1/24 x^4 - 1/120 x^5
```

- collect　collect 函数用于合并符号表达式的同类项。

【例 4-42】

```
syms x;
```

```
f=sym('(x-1)^2*(x-3)*(x-5)*(x-7)');
collect(f)
ans =
x^5-17*x^4+102*x^4-262*x^2+281*x-105
```

- horner　horner **函数用于将一般的符号表达式转换成嵌套形式的符号表达式。**

【例 4-43】

```
syms x;
f=sym('x^5-17*x^4+102*x^4-262*x^2+281*x-105');
horner(f)
ans =
    -105+(281+(-262+(102+(-17+x)*x)*x)*x)*x
```

- factor　factor **函数用于对符号表达式进行因式分解。**

【例 4-44】

```
syms x;
f=sym('x^12-1');
factor(f)
ans =
    (-1+x)*(x+1)*(x^2+x+1)*(x^2-x+1)*(x^2+1)*(x^4-x^2+1)
```

- expand　expand **函数用于对符号表达式进行展开。**

【例 4-45】

```
syms x;
f=sym('(x-1)^12');
expand(f)
ans =
1-12*x+66*x^2-220*x^3+495*x^4-792*x^5+924*x^6-792*x^7+495*x^8-220*x^9+66*x^10-12*x^1
1+x^12
```

- simplify　simplify **函数用于对符号表达式进行化简。它利用各种类型的代数恒等式，包括求和、积分、三角函数、指数函数、**Bessel **函数等来化简符号表达式。**

【例 4-46】

```
syms x;
f=sym('(x-1)^3/(x-1)');
simplify(f)
ans =
    (-1+x)^2
```

- simple　simple **函数通过各种方式对符号表达式进行化简，以显示长度最短的符号表达式化简形式。**simple **函数的调用格式有如下两种。**

simple(S)　**对符号表达式尝试用多种不同的算法进行化简，以显示长度最短的符号表达式化简形式。**

[r,how]=Simple(s)　**返回的** r **为对符号表达式进行化简后的形式，**how **为所采用的化简方法。**

【例 4-47】

```
syms x;
f=sym('(x-1)^3/(x-1)');
simple(f)

simplify:
     (-1+x)^2
radsimp:
     (-1+x)^2
combine(trig):
     1-2*x+x^2
factor:
     (-1+x)^2
expand:
     1-2*x+x^2
combine:
     (-1+x)^2
convert(exp):
     (-1+x)^2
convert(sincos):
     (-1+x)^2
convert(tan):
     (-1+x)^2
collect(x):
     1-2*x+x^2
ans =
     (-1+x)^2
```

上例中，第一个输出参数为化简所采用的方法，后一个输出参数为符号表达式化简后的结果。最后返回的结果是在各种方法中长度最短的符号表达式形式。

4.3.2 符号表达式的替换

在 MATLAB **的符号数学工具箱中，有两个函数可以进行符号表达式的替换：**subexpr **和** subs**。下面对这两个函数予以介绍。**

- subexpr　subexpr **函数的作用是，用一个** MATLAB **默认的符号变量** sigma **替换符号函数求解中产生的复杂符号表达式，使结果看起来简洁。**

【例 4-48】

```
syms a x
s = solve(x^3+a*x+1);
r = subexpr(s)
sigma =
     -108+12*(12*a^3+81)^(1/2)
```

```
r =
[                                                    1/6*sigma^(1/3)   -2*a/sigma^(1/3)]
[-1/12*sigma^(1/3)+a/sigma^(1/3)+1/2*i*3^(1/2)*(1/6*sigma^(1/3)+2*a/sigma^(1/3))]
[-1/12*sigma^(1/3)+a/sigma^(1/3)-1/2*i*3^(1/2)*(1/6*sigma^(1/3)+2*a/sigma^(1/3))]
```

当然，也可以用一个自定义的符号变量来代替默认的符号变量 sigma**。比如，上面的例子中，用符号变量** S **来代替默认的** sigma**，则有下例的结果。**

【例 4-49】

```
syms a x
s = solve(x^3+a*x+1);
r = subexpr(s, 'S')
S =
      -108+12*(12*a^3+81)^(1/2)

r =
[                                                  1/6*S^(1/3)-2*a/S^(1/3)]
[-1/12*S^(1/3)+a/S^(1/3)+1/2*i*3^(1/2)*(1/6*S^(1/3)+2*a/S^(1/3))]
[-1/12*S^(1/3)+a/S^(1/3)   -1/2*i*3^(1/2)*(1/6*S^(1/3)+2*a/S^(1/3))]
```

- subs　subs **函数除具有与** subexpr **函数一样的可以用一个** MATLAB **符号变量替换复杂符号表达式的作用外，还可以求解被替换的复杂符号表达式的值。**

【例 4-50】

某符号矩阵的特征值与特征向量求解如下。

```
syms a b c
A = [a b c; b c a; c a b];
[v,E] = eig(A)
v =
[1,                                                 -(a+(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)-b)/(a-c),
-(a-(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)-b)/(a-c)]
[1,                                                 -(b-c-(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2))/(a-c),
-(b-c+(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2))/(a-c)]
[1,              1,              1]

E =
[b+a+c, 0, 0]
[0,   (b^2-b*a-c*b-c*a+a^2+c^2)^(1/2), 0]
[0, 0, -(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)]
```

此时，v = subexpr(v,'S')

```
S =
(b^2-b*a-c*b-c*a+a^2+c^2)^(1/2)

v =
[1, -(a+S-b)/(a-c), -(a-S-b)/(a-c)]
```

```
[1, - (b-c-S)/(a-c), - (b-c+S)/(a-c)]
[1,                  1,              1]
```

下面，用符号变量 S **替换** E，E = subs(E,S,'S')

```
E =
[      S,      0,      0]
[      0,     -S,      0]
[      0,      0,  b+c+a]
```

假若被替换的符号表达式 S **中的符号变量** a，b，c **的值如下：**

```
a = 10; b = 2; c = 10;
```

则 subs(S)

```
ans =
     8
```

就可以计算出被替换的符号表达式 S **的值。**

如果在计算出被替换的符号表达式 S **的值的同时，仍然保留** a，b，c **为符号变量，结果如下所示：**

```
syms a b c
subs(S,{a,b,c},{10,2,10})

ans =

     8
```

4.4 符号矩阵

符号矩阵和符号向量中的元素都是符号表达式。

4.4.1 符号矩阵的生成

符号矩阵可通过函数 sym **来生成。符号矩阵中的元素是任何不带等号的符号表达式，各符号表达式的长度可以不同；符号矩阵中，以空格或逗号分隔的元素指定的是不同列的元素，而以分号分隔的元素指定的是不同行的元素。**

【例 4-51】

```
syms x;
A=sym('[cos(x),sin(x),x;  -x+1 x^2+x+1 tan(x)]')
A =
    [  cos(x),  sin(x),          x]
    [   -x+1, x^2+x+1,  tan(x)]
```

用函数 size **可求得符号矩阵的大小。但是，**size **函数返回的是数值或向量，而不是符号表达式。**

【例 4-52】

```
A=sym('[a b c;e f g]');
```

```
d=size(A)
d =
     2     3
```

【例 4-53】

```
a=[1 2 3 4;4 5 6 7];
b=sym(a)
b =
    [ 1, 2, 3, 4]
    [ 4, 5, 6, 7]
```

4.4.2 符号矩阵的运算

符号矩阵的运算主要包括符号矩阵的一些基本运算以及符号矩阵的常用函数运算。下面分别做简单介绍。

1. **符号矩阵的四则运算**

在 MATLAB **中，进行符号矩阵的四则运算非常方便。实际上，它与数值矩阵的四则运算完全相同，这大大地方便了用户的操作。符号矩阵的加、减、乘、除运算可分别由函数** symadd, symsub, symmul, symdiv **来实现，也可与一般的数值运算一样，用“+”，“-”，“*”，“/”符号进行运算，而符号矩阵的幂运算既可由函数** sympow **来实现，也可由幂运算符号“^”来实现。**

【例 4-54】

```
A=sym('[cos(x),sin(x);x^2+x+1 tan(x)]');
B=sym('[x+1,x^2-1;sin(x),log(x)]');
C=A+B
C =
[         cos(x)+x+1,      sin(x)+x^2-1]
[ x^2+x+1+sin(x),    tan(x)+log(x)]
```

【例 4-55】

```
A=sym('[cos(x),sin(x);x^2+x+1 tan(x)]');
B=sym('[x+1,x^2-1;sin(x),log(x)]');
C=symmul(A,B)
C =
[                  cos(x)*(x+1)+sin(x)^2,        cos(x)*(x^2-1)+sin(x)*log(x)]
[     (x^2+x+1)*(x+1)+sin(x)*tan(x), (x^2+x+1)*(x^2-1)+tan(x)*log(x)]
```

【例 4-56】

```
A=sym('[cos(x),sin(x);x^2+x+1 tan(x)]');
B=sym('[x+1,x^2-1;sin(x),log(x)]');
C=A/B
C =
[-1/(-x*log(x)   -log(x)+sin(x)*x^2-sin(x))*(cos(x)*log(x)   -sin(x)^2),
(cos(x)*x-cos(x)-sin(x))/(sin(x)*x-log(x)-sin(x))]
[-1/(-x*log(x)-log(x)+sin(x)*x^2-sin(x))*(x^2*log(x)+x*log(x)+log(x)-tan(x)*sin(x)),
  (x^4-1-tan(x))/(sin(x)*x-log(x)-sin(x))]
```

【例 4-57】

```
B=sym('[x+1,x^2-1;sin(x),log(x)]');
sympow(A,2)
ans =
[              cos(x)^2+sin(x)*(x^2+x+1),          cos(x)*sin(x)+sin(x)*tan(x)]
[ (x^2+x+1)*cos(x)+tan(x)*(x^2+x+1),              sin(x)*(x^2+x+1)+tan(x)^2]
```

需要注意的是，符号矩阵的幂运算函数 sympow(A,B)中的 B 必须是标量值。

【例 4-58】

```
A=sym('[cos(x),sin(x);x^2+x+1 tan(x)]');
B=sym('[x+1,x^2-1;sin(x),log(x)]');
sympow(A,B)
??? Error using ==> sym/mpower
Either base or exponent must be a scalar.
Error in ==> D:\matlabR12\toolbox\symbolic\sympow.m
On line 8    ==> c = sym(a) ^ sym(b);
```

上例中，由于幂运算函数 sympow(A,B)中的 B 是符号矩阵，所以返回出错信息。

2. 符号矩阵的其他一些基本运算

符号矩阵的其他一些基本运算包括符号矩阵的转置、行列式、逆、秩、指数运算等。下面分别做简要介绍。

- 符号矩阵的转置运算　符号矩阵的转置运算由函数 transpose 来实现。

【例 4-59】

```
A=sym('[cos(x),sin(x);x^2+x+1 tan(x)]');
B=transpose(A)
B =
   [    cos(x), x^2+x+1]
   [    sin(x),    tan(x)]
```

- 符号矩阵的行列式运算　符号矩阵的行列式运算由函数 determ 或 det 来实现。

【例 4-60】

```
A=sym('[cos(x),sin(x);x^2+x+1 tan(x)]');
B=determ(A)
B =
-sin(x)*x^2-sin(x)*x-sin(x)+tan(x)*cos(x)
```

【例 4-61】

```
A=sym('[cos(x),sin(x);x^2+x+1 tan(x)]');
B=det(A)
B =
-sin(x)*x^2-sin(x)*x-sin(x)+tan(x)*cos(x)
```

- 符号矩阵求逆运算　符号矩阵求逆运算与数值矩阵的求逆运算一样，由函数 inv 来实现。

【例 4-62】

```
A=sym('[cos(x),sin(x);x^2+x+1 tan(x)]');
B=inv(A)
```

```
B =
    [ -tan(x)/(sin(x)*x^2+sin(x)*x+sin(x)-tan(x)*cos(x)),
    sin(x)/(sin(x)*x^2+sin(x)*x+sin(x)-tan(x)*cos(x))]
    [(x^2+x+1)/(sin(x)*x^2+sin(x)*x+sin(x)-tan(x)*cos(x)),
    -1/(sin(x)*x^2+sin(x)*x+sin(x)-tan(x)*cos(x))*cos(x)]
```

- **符号矩阵求秩运算　符号矩阵求秩运算与数值矩阵的求秩运算一样，由函数** rank **来实现。**

【例 4-63】

```
A=sym('[cos(x),sin(x);x^2+x+1 tan(x)]');
B=rank(A)
B =
    2
```

3. **符号矩阵的常用函数运算**

符号矩阵的常用函数运算包括符号矩阵的特征值、特征向量运算，符号矩阵的奇异值运算，符号矩阵的约当标准型运算等。

- **符号矩阵的特征值、特征向量运算　在** MATLAB **中，符号矩阵的特征值、特征向量运算可通过函数“** eig **”，“** eigensys **”来实现。**

【例 4-64】

```
A=sym('[1,3/2;2 4]');
[B,C]=eig(A)
B =
    [ -3/4+1/4*21^(1/2),          -3/4-1/4*21^(1/2)]
    [                 1,                       1]
C =
    [ 5/2+1/2*21^(1/2),                    0]
    [                  0, 5/2-1/2*21^(1/2)]
```

【例 4-65】

```
A=sym('[1,3/2;2 4]');
[B,C]=eigensys(A)
B =
    [                  1,                    1]
    [ 1+1/3*21^(1/2),       1-1/3*21^(1/2)]
C =
    [ 5/2+1/2*21^(1/2),                    0]
    [                  0, 5/2-1/2*21^(1/2)]
```

上面两例中的 B **为特征向量，**C **为特征值。**

【例 4-66】

```
A=sym('[1,3/2;2 4]');
B=eigensys(A)
B =
    [ 5/2+1/2*21^(1/2)]
    [ 5/2-1/2*21^(1/2)]
```

- **符号矩阵的奇异值运算**　**符号矩阵的奇异值运算可以通过函数“**svd**”,“**singvals**”来实现。**

【例 4-67】

```
A=sym('[1,3/2;2 4]');
B=svd(A)
B =
    [ 1/4*101^(1/2)+1/4*85^(1/2)]
    [ 1/4*101^(1/2)-1/4*85^(1/2)]
```

【例 4-68】

```
A=sym('[1,3/2;2 4]');
B=singvals(A)
B =
    [ 1/4*101^(1/2)+1/4*85^(1/2)]
    [ 1/4*101^(1/2)-1/4*85^(1/2)]
```

- **符号矩阵的约当标准型运算**　**符号矩阵的约当标准型运算可以通过函数“**jordan**”来实现。**

【例 4-69】

```
A=sym('[1,3/2;2 4]');
[B,C]=jordan(A)
B =
    [ -1/14*21^(1/2)+1/2,  1/14*21^(1/2)+1/2]
    [          2/21*21^(1/2),        -2/21*21^(1/2)]
C =
    [ 5/2+1/2*21^(1/2),                    0]
    [                    0, 5/2-1/2*21^(1/2)]
```

上例中的 B **为转换矩阵，其列是特征向量；**C **为约当标准型，它是特征值的对角矩阵，即它的对角线元素是特征值。**

4.5　符号微积分

微积分是高等数学中最重要的基础内容之一，它被广泛地应用于许多的工程学科中。MATLAB **的符号数学工具箱为我们提供了快速、简便地计算微积分的工具。**MATLAB **符号数学工具箱中的符号微积分包括符号极限、符号微分、符号积分等。**

4.5.1　符号极限

极限是高等数学的出发点，同时它也是微积分学的基础。在 MATLAB **中，符号极限由函数“**limit**”来实现。**limit **函数的调用格式如下。**

- limit (F,x,a)　　**返回符号表达式** F **当** x→a **时的极限。**
- limit (F,a)　　**返回符号表达式** F **由** findsym(F) **返回的独立变量趋向于** a **时的极限。**
- limit (F)　　**返回符号表达式** F **由** findsym(F) **返回的独立变量在** $a = 0$ **处的极限值。**
- limit (F,x,a,'right') **或** limit (F,x,a,'left')　　**参数中的'right'，'left'表明取极限的方向。**

【例 4-70】

```
syms x;
limit((x-2)/(x^2-4),2)
ans =
     1/4
```

【例 4-71】

```
syms x;
limit(1/x,x,0,'left')
ans =
     -inf
```

【例 4-72】

```
syms x h;
limit((sin(x+h)-sin(x))/h,h,0)
ans =
     cos(x)
```

【例 4-73】

```
syms x a v;
v = [(1 + a/x)^x, exp(-x)];
limit(v,x,inf,'left')
ans =
     [ exp(a),          0]
```

4.5.2 符号微分

微分是高等数学中最基础的内容之一。在 MATLAB 中，符号微分由函数“ diff ”来实现。实际上，在数值运算一章中已介绍过 diff 函数了。diff 函数可同时计算数值微分与符号微分。当输入的参数是数值时，MATLAB 能非常巧妙地对其进行数值微分；当输入的参数是符号字符串时，MATLAB 同样能非常巧妙地对其进行符号微分。diff 函数的调用格式如下。

- diff(F)　　对 findsym 函数返回的独立变量求微分，F 为符号表达式。
- diff(F,'a')　　对 a 变量求微分，F 为符号表达式。
- diff(F,n)　　对 findsym 函数返回的独立变量求 n 次微分，F 为符号表达式。
- diff(F, 'a',n)或 diff(F,n, 'a')　　对变量 a 求 n 次微分，F 为符号表达式。

【例 4-74】

```
syms x;
f=sym('(x-1)^3/(x-1)');
B=diff(f)
B =
     2*x-2
```

【例 4-75】

```
syms x;
f=sym('(x-1)^3/(x+1)');
B=diff(f,2)
```

```
B =
        6*(x-1)/(x+1)-6*(x-1)^2/(x+1)^2+2*(x-1)^3/(x+1)^3
```

同样的，函数 diff **也可对符号矩阵进行运算。此时，它是对符号矩阵中的每个元素进行微分。**

【例 4-76】

```
A=sym('[cos(x),sin(x);x^2+x+1 tan(x)]');
B=diff(A)
B =
        [       -sin(x),         cos(x)]
        [         2*x+1, 1+tan(x)^2]
```

4.5.3 符号积分

由高等数学可知，积分比微分复杂得多。很多情况下，积分不一定能成功。当在 MATLAB **中进行符号积分，找不到原函数时，它将返回未经计算的函数。符号积分由函数“** int **”来实现。**int **函数的调用格式如下所示。**

- int (F)　　**对** findsym **函数返回的独立变量求不定积分，**F **为符号表达式。**
- int (F,v)　　**对** v **变量求不定积分，**F **为符号表达式。**
- int (F,a,b)　　**对** findsym **函数返回的独立变量求从** a **到** b **的定积分，**F **为符号表达式。**
- int (F,v,a,b)　　**对** v **变量求从** a **到** b **的定积分，**F **为符号表达式。**

【例 4-77】

```
syms u alpha;
int(sin(alpha*u),alpha)
ans =
        -1/u*cos(alpha*u)
```

【例 4-78】

```
int(1/(1+x^2))
ans =
        atan(x)
```

【例 4-79】

```
int('log(x)/exp(x^2)')
Warning: Explicit integral could not be found.
> In E:\MATLAB\toolbox\symbolic\@sym\int.m at line 58
  In E:\MATLAB\toolbox\symbolic\@char\int.m at line 9
ans =
        int(log(x)/exp(x^2),x)
```

上例中，当找不到原函数时，返回未经计算的函数。

与函数 diff **一样，函数** int **也可对符号矩阵进行运算。此时，它是对符号矩阵中的每个元素进行积分。**

【例 4-80】

```
syms t alpha;
int([exp(t),exp(alpha*t)])
```

```
ans =
    [exp(t), 1/alpha*exp(alpha*t)]
```

4.6 符号函数画图

MATLAB 的符号工具箱也为用户采用符号函数进行画图提供了方便。符号函数画图可以通过函数“ ezplot ”或“ fplot ”来实现。其中，函数 ezplot 的调用格式如下。

- ezplot (f)　　表示在默认区间 -2*pi < x < 2*pi 绘制 $f = f(x)$ 的函数图。
- ezplot (f, [a,b])　　表示在区间 $a < x < b$ 绘制 $f = f(x)$ 的函数图。对于隐函数，$f = f(x,y)$。
- ezplot (f)　　表示在默认区间 -2*pi < x < 2*pi 和 -2*pi < y < 2*pi 绘制 $f(x,y) = 0$ 的函数图。
- ezplot (f, [xmin,xmax,ymin,ymax])　　表示在区间 xmin < x < xmax 和 ymin < y < ymax 绘制 $f(x,y) = 0$ 的函数图。
- ezplot (f, [a,b])　　表示在区间 $a < x < b$ 和 $a < y < b$ 绘制 $f(x,y) = 0$ 的函数图。
- ezplot (x,y)　　表示在区间 0 < t < 2*pi 绘制 $x = x(t)$和 $y = y(t)$的函数图。
- ezplot (x,y, [tmin,tmax])　　表示在区间 tmin < t < tmax 绘制 $x = x(t)$和 $y = y(t)$的函数图。
- ezplot (f, [a,b], FIG), ezplot (f, [xmin,xmax,ymin,ymax], FIG), ezplot (x,y, [tmin,tmax], FIG)　　表示在指定的图形窗口 FIG 中画图，以代替当前的图形窗口。

【例 4-81】

```
ezplot('cos(x)', [0, pi])
```

画图结果如图 4-1 所示。

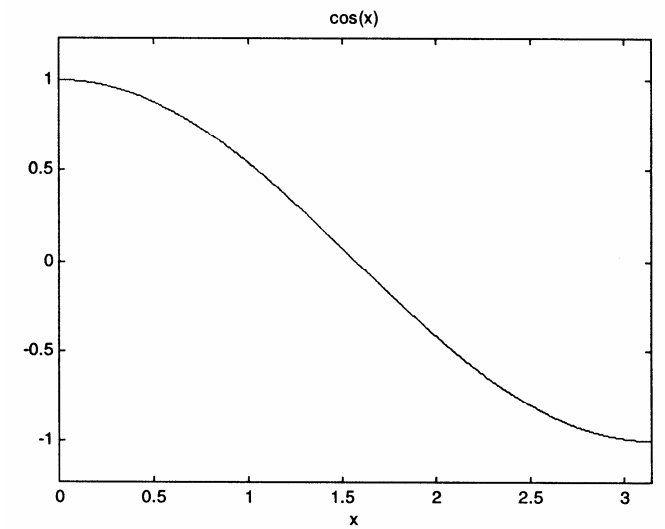


图 4-1　余弦函数图

【例 4-82】

```
ezplot('sin(3*t)*cos(t)','sin(3*t)*sin(t)',[0,pi])
```

画图结果如图 4-2 所示。

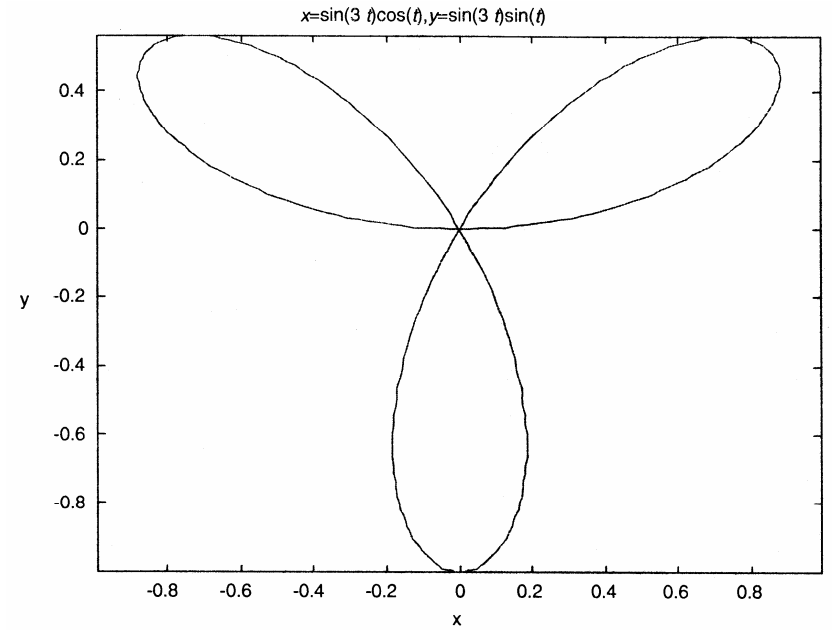


图 4-2　**参数形式函数图**

fplot **函数的调用格式如下。**

- fplot (fun,lims)　　**表示绘制字符串** fun **指定的函数在区间** lims=[xmin,xmax]**的图形。如果** lims=[xmin,xmax,ymin,ymax]**，则** y **被限制在区间**[ymin,ymax]**内。**fun **必须是** M **文件的函数名或是独立变量为** x **的字符串，此字符串被送入函数** eval**。函数** fun(x)**必须对向量中的每个元素** x **返回一行向量。例如，如果** fun **为**[f1(x),f2(x),f3(x)]**，则当输入**[x1;x2]**后，返回的是矩阵。**

  [f1(x1) f2(x1) f3(x1)]
  [f1(x2) f2(x2) f3(x2)]

- fplot(fun,lims,tol)　　**输入参数** tol **用来表示相对误差精度，默认的** tol **是** 0.002**。其他同上。**
- fplot (fun,lims,n)　　**输入参数** n **表示以** n+1 **个点来画图，**n **的默认值是** 1**，最大步长被限制在**(1/n)*(xmax-xmin)**。**
- fplot ((fun,lims,'LineSpec')　　**表示以指定线型** LineSpec **来画图。**
- fplot ((fun,lims,…)　　**表示包括输入参数** tol, n, LineSpec**，参数的顺序可以任意。**
- [x,y]= fplot ((fun,lims,…)　　**代替在屏幕上画出图形，返回绘制函数图** y=Fun(x)**的向量值** x **与** y**。**
- fplot(fun,lims,tol,n,'LineSpec',p1,p2,…)　　**表示画函数** y=fun(x,p1,p2,…)**的图形，其他同上。**

【例 4-83】

```
fplot('abs(exp(-j*x*(0:9))*ones(10,1))',[0 2*pi])
```

画图结果如图 4-3 **所示。**

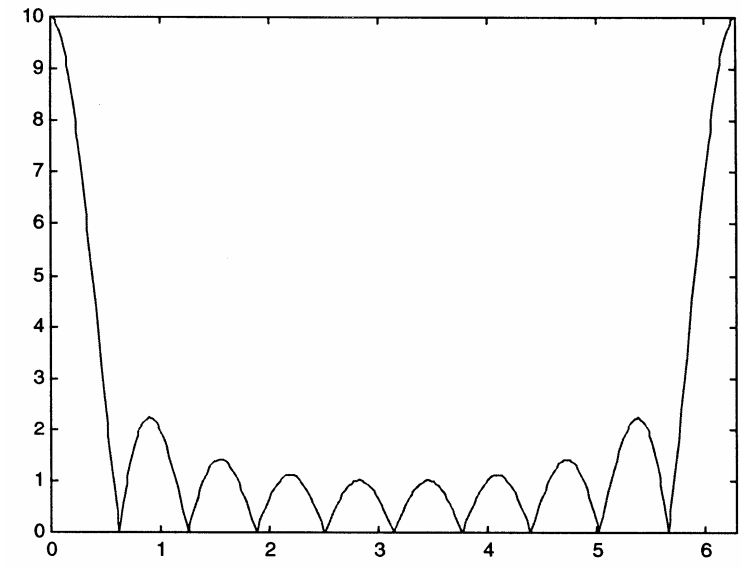


图 4-3　fplot 函数绘制的函数图

4.7　符号方程求解

在 MATLAB 的符号数学工具箱中，符号方程是含有等号的符号表达式。符号方程的求解包括符号代数线性方程的求解、符号代数非线性方程的求解和符号微分方程的求解等。

4.7.1　符号代数线性方程求解

符号代数线性方程的求解可以通过函数“ solve ”，“ linsolve ”来实现。solve 函数的求解如下例所示。

【例 4-84】

```
solve('p*sin(x) = r')
ans =
    asin(r/p)
```

如果符号表达式不含等号，则函数 solve 会自动将表达式转换成等号右端为 0 的符号方程，如下例所示。

【例 4-85】

```
solve('p*tan(x)-r')
ans =
    atan(r/p)
```

如果想对非默认变量求解，则 solve 函数必须指定变量，如下例所示。

【例 4-86】

```
solve('a*x^2+b*x+c','a')
ans =
    - (b*x+c)/x^2
```

另外，solve 函数也可解线性方程组。

【例 4-87】

```
[x,y] = solve('x^2 + x*y + y = 3','x^2 - 4*x + 3 = 0')
x =
     [ 1]
     [ 3]
y =
     [     1]
     [-3/2]
```

linsolve 函数的求解如下例所示。

【例 4-88】

```
a=sym('[10 -1 0;-1 10 -2]');
b=sym('[1;2]');
linsolve(a,b)
ans =
     [     1/9]
     [ 19/72]
```

实际上，x=linsolve(A,B)与 x= sym(A)\ sym(B)的求解结果相同。

【例 4-89】

```
a=sym('[10 -1 0;-1 10 -2]');
b=sym('[1;2]');
a\b
ans =
     [     1/9]
     [ 19/72]
```

4.7.2 符号代数非线性方程求解

符号代数非线性方程的求解可以通过函数“ fsolve ”来实现。fsolve 函数以最小二乘法求解非线性方程，其调用格式如下。

- x=fsolve(fun,x0)　　输入参数中的 x0 为所求解方程的初始向量或矩阵，fun 为所要求解的符号方程，它通常是以 M 文件的形式给出。
- x=fsolve(fun,x0,options)　　输入参数中的 options 是可选参数，它可以通过 optimset 函数生成（详情可见 optimset 函数）。可选参数 options 可以是 Display, TolX, TolFun, DerivativeCheck, Diagnostics, Jacobian, JacobPattern, LineSearchType, venbergMarquardt, xFunEvals, xIter, ffMinChange, ffMaxChange, rgeScale, xPCGIter, econdBandWidth, TolPCG, picalX 等。
- x=Fsolve(fun,x0,options,p1,p2,…)　　p1,p2,…直接赋给函数 fun，即 fun(x,p1,p2,…)。此时，若 options 使用默认值，则需要输入空矩阵。

【例 4-90】

首先编写 M 文件如下：

myfun.m

```
function F = myfun(x)
F = [2*x(1) - x(2) - exp(-x(1));
-x(1) + 2*x(2) - exp(-x(2))];
```

然后，在 MATLAB **主窗口中输入：**

```
x0 = [-5; -5];
options=optimset('Display','iter');
[x,fval] = fsolve('myfun',x0,options)
```

运行后结果如下：

Iteration	Func-count	f(x)	Norm of step	First-order optimality	CG-iterations
1	4	47071.2	1	2.29e+004	0
2	7	6527.47	1.45207	3.09e+003	1
3	10	918.372	1.49186	418	1
4	13	127.74	1.55326	57.3	1
5	16	14.9153	1.57591	8.26	1
6	19	0.779051	1.27662	1.14	1
7	22	0.00372453	0.484658	0.0683	1
8	25	9.21617e-008	0.0385552	0.000336	1
9	28	5.66133e-017	0.000193707	8.34e-009	1

```
Optimization terminated successfully:
Relative function value changing by less than OPTIONS.TolFun
x =
    0.5671
    0.5671
fval =
    1.0e-008 *
   -0.5320
   -0.5320
```

4.7.3 符号微分方程求解

符号微分方程的求解可以通过函数“ dsolve **”来实现，**dsolve **函数的调用格式如下。**

dsolve ('eqn1','eqn2',…) **输入参数**'eqn1', 'eqn2'**代表微分方程与初始条件，几个微分方程与初始条件可以一起输入，只要以逗号分隔开即可。默认的独立变量是** t**，用户也可以使用别的变量来代替** t**，只要把别的变量放在输入变量的最后即可。字母** D **代表微分算子，即** d/d*t*，**字母** D **后面所跟的数字代表几阶微分，如** D2 **代表** d^2/dt^2**。跟在微分算子后面的字母是被微分的变量，如** D3y **代表对** y(t)**的三阶微分。注意，在符号变量中不能再出现字母** D**。初始条件可用这样的形式给出：**'y(a)=b' **或**'Dy(a) = b'**。此处的** y **是被微分变量，**a **和** b **是常量。如果初始条件的个数少于被微分变量的个数，则解中会出现** C1, C2 **这样的不定常数。**

函数的输出结果可能存在如下 3 **种情况：**

（1**）一个方程和一个输出，则返回符号向量中非线性方程的联立解。**

（2**）几个方程与相同个数的输出，返回的结果是按字母顺序排序，并且分配给输出参数。**

（3）几个方程和一个输出，则返回解的结构。

如果解不是显式形式，则被认为是隐式形式。当一个隐式形式的解返回时，会给出警告。如果解既不是显式形式，也不是隐式形式，也会给出警告，同时返回的是一个空字符串。在一些非线性方程中，输出结果可能是降阶的微分方程或积分方程。

【例 4-91】

```
y = dsolve('(Dy)^2 + y^2 = 1','y(0) = 0')
y =
    [-sin(t)]
    [ sin(t)]
```

【例 4-92】

```
S = dsolve('Df = f + g','Dg = -f + g','f(0) = 1','g(0) = 2')
S =
    f: [1x1 sym]
    g: [1x1 sym]
```

上例中，可通过如下方式查看结果：

```
S.f
  ans =
  exp(t)*(cos(t)+2*sin(t))
S.g
  ans =
  -exp(t)*(sin(t)-2*cos(t))
```

【例 4-93】

```
Y = dsolve('Dy = y^2*(1-y)')
Warning: Explicit solution could not be found; implicit solution returned.
> In E:\MATLAB\toolbox\symbolic\dsolve.m at line 292
Y =
    t+1/y-log(y)+log(-1+y)+C1=0
```

上例返回的是隐式形式的解。

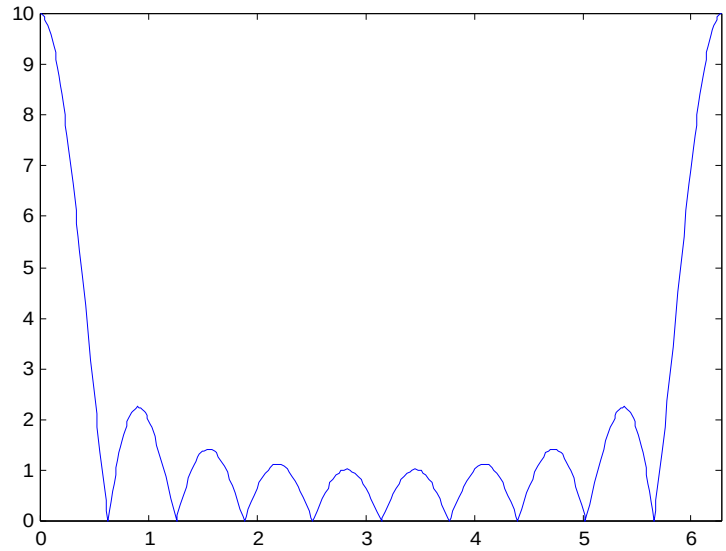


图 4-3　fplot 函数绘制的函数图

4.7　符号方程求解

在 MATLAB 的符号数学工具箱中，符号方程是含有等号的符号表达式。符号方程的求解包括符号代数线性方程的求解、符号代数非线性方程的求解和符号微分方程的求解等。

4.7.1　符号代数线性方程求解

符号代数线性方程的求解可以通过函数“solve”，“linsolve”来实现。solve 函数的求解如下例所示。

【例 4-84】

```
solve('p*sin(x) = r')
ans =
    asin(r/p)
```

如果符号表达式不含等号，则函数 solve 会自动将表达式转换成等号右端为 0 的符号方程，如下例所示。

【例 4-85】

```
solve('p*tan(x)-r')
ans =
    atan(r/p)
```

如果想对非默认变量求解，则 solve 函数必须指定变量，如下例所示。

【例 4-86】

```
solve('a*x^2+b*x+c','a')
ans =
    -(b*x+c)/x^2
```

第 5 章　一般图形功能

MATLAB 的图形绘制和编辑功能非常强大。在 MATLAB 中，既可以绘制线形图、条形图等基本图形，也可以绘制不同坐标系下的图形，如对数坐标图、极坐标图等；还可以绘制不同专业用到的功能图形，如玫瑰花图、阶梯图等。此外，MATLAB 还有很强的三维可视化功能，能够绘制具有高度真实感的实体模型图（这部分内容在第 6 章具体介绍）。

5.1　基本图形绘制

本节介绍常见的线形图、条形图、带形图、面积图、误差条图、饼图、散点图和直方图等的绘制。

5.1.1　线形图

1．二维线形图

利用 plot 函数可以绘制二维线形图，其调用格式如下。

- plot(Y)　　若 Y 的值为实数，则根据 Y 各列的数据与它们对应的编号绘二维线形图。若 Y 的值为复数，则 plot(Y)与 plot(real(Y),imag(Y))等价，即根据 Y 各元素的实部和虚部数据绘图。
- plot(X1,Y1,…)　　绘制所有由 Xn 和 Yn 确定的直线。若 Xn 和 Yn 中只有一个为矩阵，则根据向量与矩阵的行或列的配对数据绘图，取行还是取列决定于向量的行或列的维数是否与矩阵匹配。
- plot(X1,Y1,LineSpec,…)　　绘 Xn, Yn 和 LineSpec 定义的所有直线。其中，LineSpec 指定直线的线型、标记和颜色。
- plot(…,'PropertyName',PropertyValue,…)　　给所有由 plot 函数创建的图形对象设置属性值。
- h = plot(…)　　返回一个句柄列向量到直线图形对象，一个句柄代表一条直线。

绘图时，可以指定标记的颜色和大小，也可以用图形属性指定其他线条特征，这些属性包括：

- LineWidth　　指定线条的粗细（以点为单位）。
- MarkerEdgeColor　　指定标记的颜色或图形封闭的标记（圆形、方形、金刚石形等）的边缘色。
- MarkerFaceColor　　指定图形封闭的标记表面的颜色。
- MarkerSize　　以点集为单位指定标记的大小。

【例 5-1】　下面的程序段生成一条曲线，并设定曲线的属性。

```
x=-pi:pi/10:pi;
y=sin(x);
plot(x,y,'--rs','LineWidth',2,…
```

'MarkerEdgeColor','k'…)

上面的程序段所生成的曲线如图 5-1 所示。

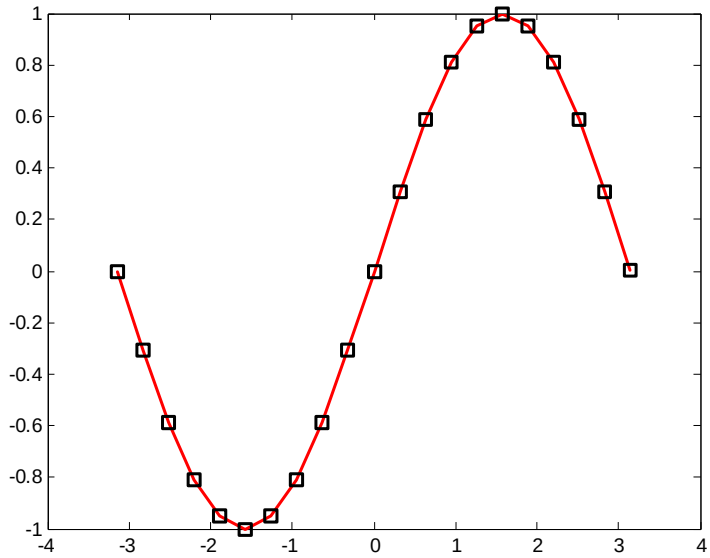


图 5-1　曲线图

可以调整坐标轴度量标记的位置和标签。图 5-2 中改变正弦曲线的标记形式。

```
t = 0:pi/20:2*pi;
plot(t,sin(t),'-.r*')
hold on
plot(sin(t-pi/2),'--mo')
plot(sin(t-pi),':bs')
hold off
```

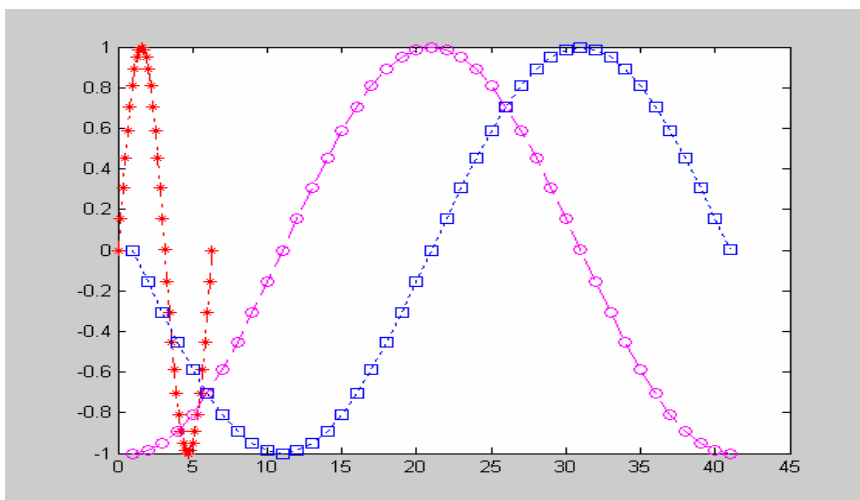


图 5-2　改变正弦曲线的标记形式

2．三维线形图

用 plot3 函数绘制三维线形图。该函数的调用格式如下。

- plot3(X1,Y1,Z1,…)　　绘一系列数据点的三维线形图，其中 X1, Y1 和 Z1 为向量或矩阵，为各点的三维坐标，通过这些点在三维空间中绘一条或多条直线。
- plot3(X1,Y1,Z1,LineSpec,…)　　创建和显示由 Xn, Yn, Zn 和 LineSpec 等定义的所有直线，其中 LineSpec 决定直线的线型、标记和颜色。
- plot3(…,'PropertyName',PropertyValue,…)　　为所有由 plot3 函数创建的直线图形对象设置属性值。
- h = plot3(…)　　返回句柄列向量到直线图形对象，一个句柄对应一条直线。

【例 5-2】　下面程序绘制一个三维线形图。

```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
grid on
axis square
```

绘制结果如图 5-3 所示。

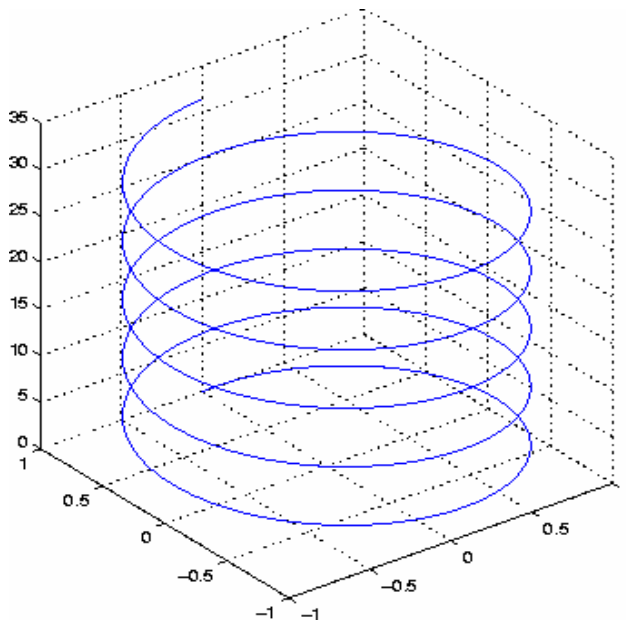


图 5-3　三维线形图

如果 plot3 函数的变量为大小相同的矩阵，则 MATLAB 利用矩阵 X, Y 和 Z 的列绘图。例如：

```
[X,Y] = meshgrid([-2:0.1:2]);
Z = X.*exp(-X.^2-Y.^2);
plot3(X,Y,Z)
grid on
```

绘制结果如图 5-4 所示。

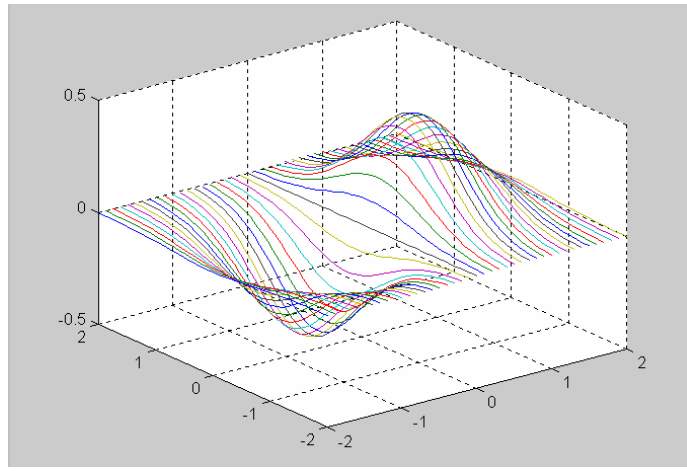


图 5-4　矩阵数据绘图

5.1.2　带形图

用 ribbon **函数生成带状图，其调用格式如下。**

- ribbon(Y)　　**用** X = 1:size(Y,1)**将** Y **的各列绘成单独的三维带状图。**
- ribbon(X,Y)　　**绘** X **对** Y **的各列的三维条带图。**X **和** Y **为大小相同的向量或矩阵。另外，**X **可以是一个行向量或列向量，**Y **是有** length(X)**行的矩阵。**
- ribbon(X,Y,width)　　**指定条带的宽度，默认值为** 0.75**。**
- h = ribbon(…)　　**返回表面图对象的句柄向量。**ribbon **函数为每个条带返回一个句柄。**

【例 5-3】　创建一个 peaks **函数的条带图。**

```
[x,y] = meshgrid(-3:.5:3,-3:.1:3);
z = peaks(x,y);
ribbon(y,z)
colormap hsv
```

绘制结果如图 5-5 **所示。**

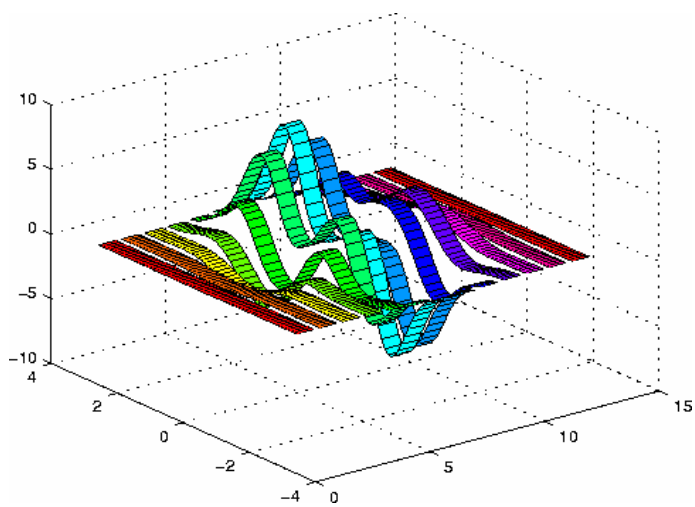


图 5-5　**条带图**

5.1.3 条形图

1．二维条形图

条形图用水平条形或垂直条形来表示向量或矩阵中的值。用函数 bar 或 barh 生成条形图，其调用格式如下。

- bar(Y)　　为 Y 的每一个元素绘一个条形。若 Y 为一矩阵，则 bar 函数对每一行元素生成的条形进行分组。当 Y 为一向量时，*x* 轴的度量为 1 到 length(Y)，若 Y 为一矩阵，则 *x* 轴的度量为 1 到 size(Y,1)。
- bar(x,Y)　　在 x 指定的位置上绘 Y 中每个元素的条形。
- bar(…,width)　　设置相邻条形的宽度并控制组内条形的间隔，默认宽度为 0.8。所以，若不指定 x，则组内的条形只有细微的差别。若宽度为 1，则组内的条形一个紧挨一个。
- bar(…,'style')　　指定条形的类型。'style'可以是'group'或'stack'。'group'为默认的显示模式。'group'显示 *n* 个条形组，每一个条形组中有 *m* 个垂直条形。其中，*n* 为 Y 的行数，*m* 为列数。条形组中包含 Y 中每一列对应的条形。'stack'为 Y 中的每一行显示一个条形。条形高度为行中元素的和。每一个条形都用多种颜色表示，颜色对应于不同种类的元素并显示每行元素对总和的相对贡献。
- bar(…,LineSpec)　　用 LineSpec 指定的颜色显示所有条形。
- [xb,yb] = bar(…)　　返回用 plot(xb,yb)或 patch(xb,yb,C)绘制的向量。它使用户对图形外观有了更多的控制。
- h = bar(…)　　返回一个句柄向量。bar 函数为 Y 中的每一列数据创建一个图形对象。
- barh(…), [xb,yb] = barh(…)和 h = barh(…)　　创建水平条形。Y 决定条形长度。

【例 5-4】　绘制钟形图。

```
x = -2.9:0.2:2.9;
bar(x,exp(-x.*x))
colormap hsv
```

绘制结果如图 5-6 所示。

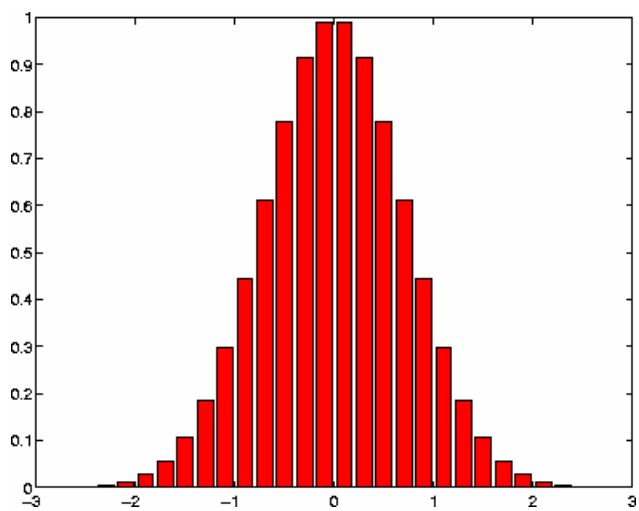


图 5-6　钟形图

2．三维条形图

用函数 bar3 和 bar3h 绘制三维条形图，其调用格式如下。

- bar3 函数和 bar3h 函数　　绘制三维垂直条形图和三维水平条形图。
- bar3(Y)　　绘制三维条形图，其中，Y 中的每一个元素对应于一个条形。当 Y 为向量时，*x* 轴的度量从 1 变到 length(Y)。当 Y 为矩阵时，*x* 轴的度量从 1 变到 size(Y,2)，做为列数，每行中的元素进行了分组。
- bar3(x,Y)　　在 x 指定的位置上绘 Y 中元素的条形图。
- bar3(…,width)　　设置条形的宽度并控制同组条形的间隔。默认时，条形宽度为 0.8。所以，如果不指定 x，则组内条形的间隔非常小。若宽度为 1，则组内条形紧密相连。
- bar3(…,'style')　　指定条形的类型。'style'可以是 'detached'，'grouped'或'stacked'。'detached' 为默认的显示模式。

  'detached' 在 x 的方向上用单独的条块显示 Y 中每一行的元素。

  'grouped' 显示 *n* 组条形集，每个条形集中有 *m* 个条形。其中，*n* 为 Y 的行数，*m* 为列数。

  'stacked' 为 Y 中的每一行显示一个条形。条形高度为行中元素的和。每个条形都用多种颜色表示，各种颜色对应于不同种类的元素并显示每行元素对总和的相对贡献。
- bar3(…,LineSpec)　　用 LineSpec 指定的颜色显示所有条形。
- h = bar3(…)　　返回阴影图形对象的句柄向量。bar3 为 Y 中的每一列创建一个阴影对象。
- bar3h(…)和 h = bar3h(…)　　创建水平条形。Y 决定条形长度。

【例 5-5】　本例创建 6 个子图，显示 bar3 函数不同变量设置的效果。数据 Y 为用 cool 颜色图创建的 7×3 矩阵。

```
Y = cool(7);
subplot(3,2,1)
bar3(Y,'detached')
title('Detached')
subplot(3,2,2)
bar3(Y,0.25,'detached')
title('Width = 0.25')

subplot(3,2,3)
bar3(Y,'grouped')
title('Grouped')

subplot(3,2,4)
bar3(Y,0.5,'grouped')
title('Width = 0.5')
subplot(3,2,5)
bar3(Y,'stacked')
title('Stacked')
```

```
subplot(3,2,6)
bar3(Y,0.3,'stacked')
title('Width = 0.3')

colormap([1 0 0;0 1 0;0 0 1])
```

绘制结果如图 5-7 **所示。**

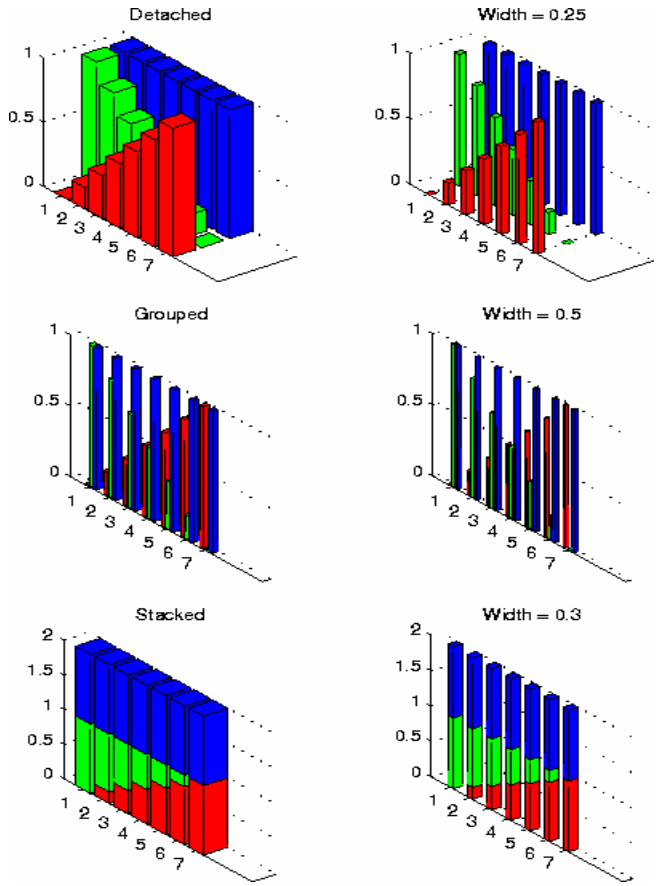


图 5-7　**三维条形图**

5.1.4　面积图

面积图将向量（或矩阵）Y **中的元素显示为一条或多条曲线，并填充每条曲线以下的面积。当** Y **为矩阵时，曲线堆栈显示，显示每个** x **区间内每行元素对曲线总高度的贡献。**

用函数 area **进行二维图的面积填充，其调用格式如下。**

- area(Y)　　**如果** Y **为向量，则根据它的值绘图，如果** Y **为矩阵，则根据它每一列的值绘制面积图。**x **轴自动根据** length(Y)**（当** Y **为向量时）或** size(Y,1)**（当** Y **为矩阵时）确定比例。**
- area(X,Y)　　**绘** X **的对应点处的** Y **数据的图。如果** X **为一向量，则** length(X)**必须等于** length(Y)**，**X **必须是单调的。如果** X **为一矩阵，则** size(X)**必须等于** size(Y)**，且** X **的每一列必须是单调的。使用** sort **函数可以使向量或矩阵单调化。**

- area(…,ymin)　　对面积填充，指定 y 方向上的下限。默认时，ymin 等于 0。
- area(…,'PropertyName',PropertyValue,…)　　为 area 函数创建的阴影图形对象指定属性名和属性值。
- h = area(…)　　返回阴影图形对象的句柄。area 函数为 Y 中的每一列创建一个阴影对象。

注意：area 函数为向量中的所有元素或矩阵中的每个列创建一条曲线。曲线颜色从整个色谱图范围内等间距选取。

【例 5-6】　利用 Y 的值绘制堆栈面积图。

```
Y = [   1, 5, 3;
        3, 2, 7;
        1, 5, 3;
        2, 6, 1];
area(Y)
grid on
colormap summer
set(gca,'Layer','top')
title 'Stacked Area Plot'
```

绘制结果如图 5-8 所示。

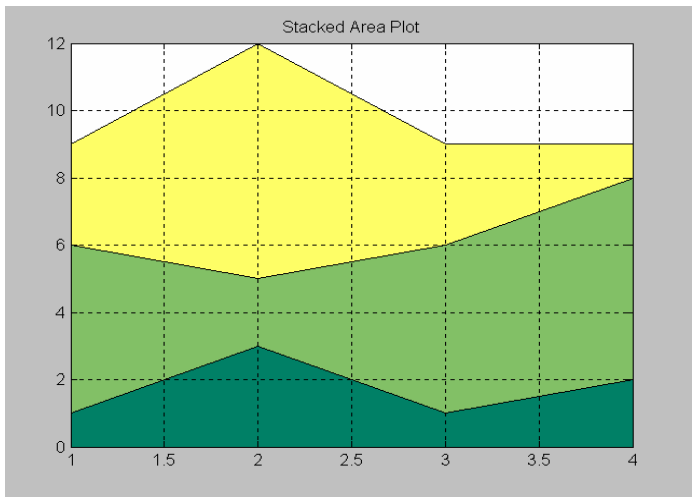


图 5-8　堆栈面积图

5.1.5　饼图

1．二维饼图

饼图显示某向量或矩阵中各元素所占的比例。pie 函数和 pie3 函数创建二维饼图和三维饼图。

用 pie 函数绘饼图，其调用格式如下。

- pie(X)　　使用 X 中的数据绘制饼图。X 中的每一个元素用饼图中的一个扇区表示。
- pie(X,explode)　　将一个扇区从饼图中分离。explode 为 X 对应的零或非零矩阵。非零值对应的扇区将从饼图中分离，所以若 explode(i,j)非零，则 X(i,j)对应扇区从中心分

离。explode 必须与 X 有相同的大小。

- h = pie(…)　　返回句柄向量到阴影和文本图形对象。

注意：X 中的值通过 X/sum(X)进行了正态化，以决定饼图中每个扇区的面积。若 sum(X)=1，则 X 中的值直接指定扇区的面积。若 sum(X)<1，则只绘出一个不完整的饼图。

【例 5-7】　下面用 pie 函数生成饼图，显示 3 种产品的销量分别占总销量的比例。给定矩阵 X，其中，X 的每一列包含 5 年内指定产品的年销量图。

```
X = [19.3 22.1 51.6;
     34.1 70.3 82.4;
     61.4 82.9 90.8;
     50.5 54.9 59.1;
     29.4 36.3 47.0];
x = sum(X);
```

可以使用 explode 输入变量将饼图中贡献最大的一个扇区分离出来。该变量是一个包含零和非零值的向量。

首先，创建一个包含零值的向量。

```
explode = zeros(size(x));
```

然后找到各扇区中贡献最大的一个，并将其对应的 explode 元素设置为 1。

```
[c,offset] = max(x);
explode(offset) = 1;
```

explode 向量包含元素[0 0 1]。使用下面的命令创建分离的饼图。

```
h = pie(x,explode);
```

生成如图 5-9 所示的饼图。

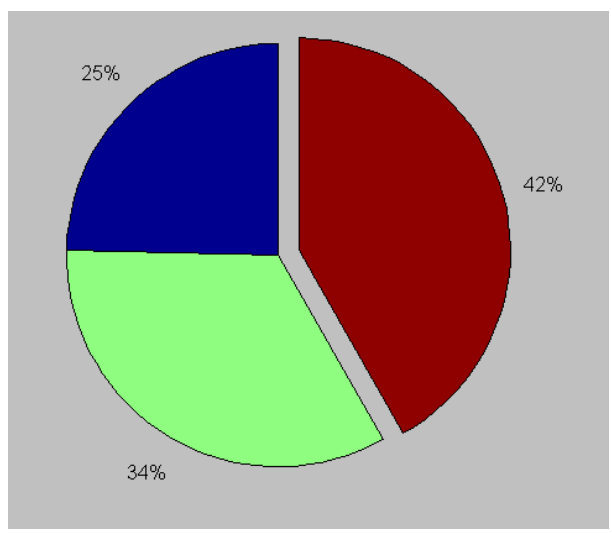


图 5-9　分离的饼图

饼图的标签为文本图形对象。要修改文本字符串和它们的位置，首先获得对象的字符串和内容。包围一个属性名可以保证输出为一单元数组，当用多对象进行绘图时这一点很重要。

```
textObjs = findobj(h,'Type','text');
oldStr = get(textObjs,{'String'});
```

```
val = get(textObjs,{'Extent'});
oldExt = cat(1,val{:});
```

创建新的字符串，然后将文本对象的字符串属性设置为新的字符串。

```
Names = {'Product X: ';'Product Y: ';'Product Z: '};
newStr = strcat(Names,oldStr);
set(textObjs,{'String'},newStr)
```

在新文本字符串和老文本字符串之间寻找差异并改变 Position **属性的值。**

```
val1 = get(textObjs, {'Extent'});
newExt = cat(1, val1{:});
offset = sign(oldExt(:,1)).*(newExt(:,3)-oldExt(:,3))/2;
pos = get(textObjs, {'Position'});
textPos =    cat(1, pos{:});
textPos(:,1) = textPos(:,1)+offset;
set(textObjs,{'Position'},num2cell(textPos,[3,2]))
```

标注结果如图 5-10 **所示。**

2．三维饼图

用 pie3 **函数绘三维饼形图，其调用格式如下。**

- pie3(X)　　**用** X **中的数据绘一个三维饼形图。**X **中的每一个元素代表饼图中的一个扇区。**
- pie3(X,explode)　　**确定是否从饼图的中心分离一个扇区。若** explode(i,j)**非零，则** X(i,j) **对应的扇区从饼图中心分离。**explode **必须与** X **具有相同的大小。**
- h = pie(…)　　**返回阴影、表面和文本图形对象的句柄向量。**

【例 5-8**】　通过将对应的分离元素设置为** 1 **来分离扇区。**

```
x = [1 3 0.5 2.5 2]
explode = [0 1 0 0 0]
pie3(x,explode)
colormap hsv
```

绘制结果如图 5-11 **所示。**

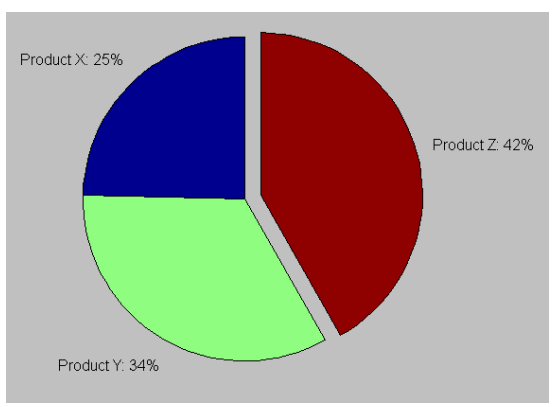


图 5-10　**饼图标注**

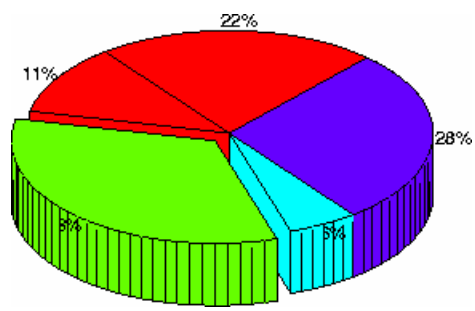


图 5-11　**三维饼图**

5.1.6 误差条图

误差条图显示数据的置信区间或沿曲线的偏差。用 errorbar 函数绘误差条图，其调用格式如下。

- errorbar(Y,E)　　根据 Y 的数据绘图并在 Y 的每个元素处绘一误差条。误差条两端距离曲线上下均为 E(i)长度。
- errorbar(X,Y,E)　　根据 X 和 Y 的误差条图，误差条的长度为 2*E(i)长度。 X, Y 和 E 必须大小相同。当它们为向量时，每个误差条均为由(X(i),Y(i))定义的距离曲线上、下各 E(i)的误差条。当它们为矩阵时，每个误差条由(X(i,j),Y(i,j))定义。
- errorbar(X,Y,L,U)　　用 L(i)+U(i)指定误差条上、下的长度，绘制误差条图。X,Y,L 和 U 必须大小相同。当它们为向量时，每个误差条由(X(i),Y(i))定义，用 L(i)定义下面的距离，用 U(i)定义上面的距离。当它们为矩阵时，每个误差条由(X(i,j),Y(i,j))定义，用 L(i,j)定义下面的距离，用 U(i,j)定义上面的距离。
- errorbar(…,LineSpec)　　用 LineSpec 指定线型、标记和颜色，绘制误差条图。
- h = errorbar(…)　　返回直线图形对象的句柄向量。

【例 5-9】　绘对称的误差条图，误差条的长度为两倍标准误差。

```
x=-pi:pi/20:pi;
y=cos(x);
E=std(y)*ones(size(x));
errorbar(x,y,E)
```

绘制结果如图 5-12 所示。

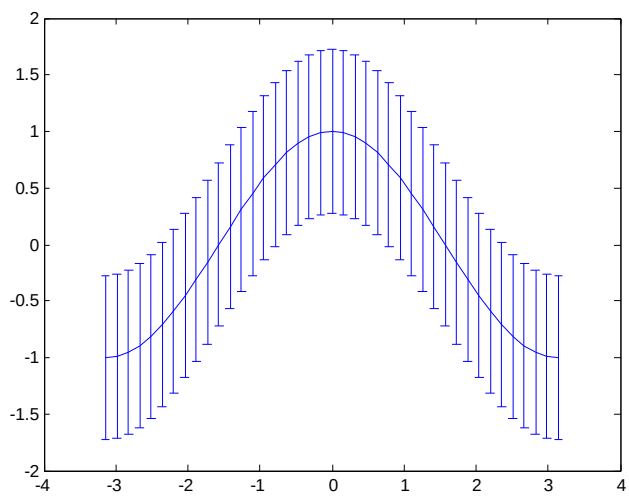


图 5-12　误差条图

5.1.7 散点图

1．二维散点图

用 scatter 函数绘二维散点图，其调用格式如下。

- scatter(X,Y,S,C)　　在向量 X 和 Y 指定的位置显示彩色圆圈。X 和 Y 必须大小相同。

S 确定标记的大小。它可以是与 X 和 Y 长度相同的向量，也可以是一个标量。若 S 为标量，则 MATLAB 将所有的标记绘成相同大小。C 确定每个标记的颜色。当 C 为与 X 和 Y 长度相同的向量时，将根据 C 中的值在当前彩色图中进行线性着色。当 C 为 length(X)×3 的矩阵时，它用 RGB 值指定标记的颜色。C 也可以是一个颜色字符串。

- scatter(X,Y)　　用大小和颜色的默认设置绘标记。
- scatter(X,Y,S)　　用单一的颜色和指定的大小绘标记。
- scatter(…,markertype)　　用指定的标记类型替代'o'。
- scatter(…,'filled')　　填充标记。
- h = scatter(…)　　返回 scatter 函数创建的直线对象的句柄。

【例 5-10】　根据下面 x 和 y 的数据绘散点图。

```
x=[40.080  27.254  33.667  38.477  38.477  49.699  48.096  62.525  68.938  65.731
   67.334  59.318  40.08   57.715  41.683  72.144  43.286  44.890  41.683  59.318
   56.112  40.08   51.302  43.286 107.414  80.160  56.112  44.890  33.667  65.731
   70.541  48.096  64.128];
y=[37.904  40.825  47.875  39.114  38.785  38.426  24.612  33.086  119.754  91.640
   91.205  89.652  90.846  21.154   9.332  25.893  39.485  42.492  39.228  53.697
   35.839  38.039  60.513  53.332  26.373 178.730   5.649  22.867   3.056  91.580
    1.248  27.999  80.148];
scatter(x,y,5)
```

绘制结果如图 5-13 所示。

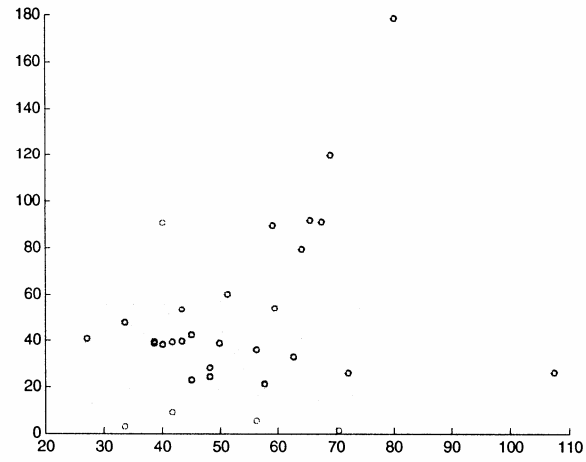


图 5-13　二维散点图

2．散点矩阵图

用 plotmatrix 函数绘散点图，其调用格式如下。

- plotmatrix(X,Y)　　根据 X 和 Y 的列数据绘散点图。若 X 为 $p\times m$ 的矩阵，Y 为 $p\times n$ 的矩阵，则 plotmatrix 函数生成一个 $n\times m$ 的坐标轴矩阵。除了对角线元素被(Y(:,i))所替代以外，plotmatrix(Y)函数的绘制效果与 plotmatrix(Y,Y)的相同。
- plotmatrix(…,'LineSpec')　　使用 LineSpec 参数创建散点图。默认符号为“ . ”。
- [H,AX,BigAx,P] = plotmatrix(…)　　把句柄矩阵返回到 H 中。

【例 5-11】　创建一个随机数散点图。

```
x = randn(50,3); y = x*[-1 2 1;2 0 1;1 -2 3;]';
plotmatrix(y,'*r')
```

绘制结果如图 5-14 所示。

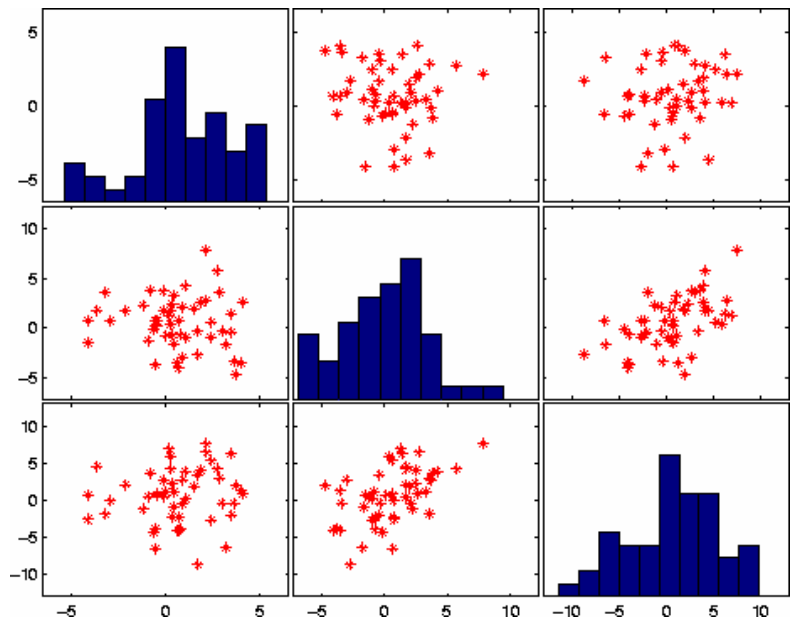


图 5-14　随机数散点图

3．三维散点图

用 scatter3 函数绘三维散点图，其调用格式如下。

- scatter3(X,Y,Z,S,C)　　**在向量 X，Y 和 Z 指定的位置上显示彩色圆圈。X，Y 和 Z 的大小必须相同。S 和 C 的意义同前（请参见二维散点图的内容）。**
- scatter3(X,Y,Z)　　**用大小和颜色的默认设置绘标记。**
- scatter3(X,Y,Z,S)　　**用单色和指定的大小绘标记。**
- scatter3(…,markertype)　　**使用指定的标记类型代替'o'。**
- scatter3(…,'filled')　　**填充标记。**
- h = scatter3(…)　　**返回由 scatter3 函数创建的直线对象的句柄。**

【例 5-12】

```
[x,y,z] = sphere(16);
X = [x(:)*.5 x(:)*.75 x(:)];
Y = [y(:)*.5 y(:)*.75 y(:)];
Z = [z(:)*.5 z(:)*.75 z(:)];
S = repmat([1 .75 .5]*10,prod(size(x)),1);
C = repmat([1 2 3],prod(size(x)),1);
scatter3(X(:),Y(:),Z(:),S(:),C(:),'filled'), view(-60,60)
```

绘制结果如图 5-15 所示。

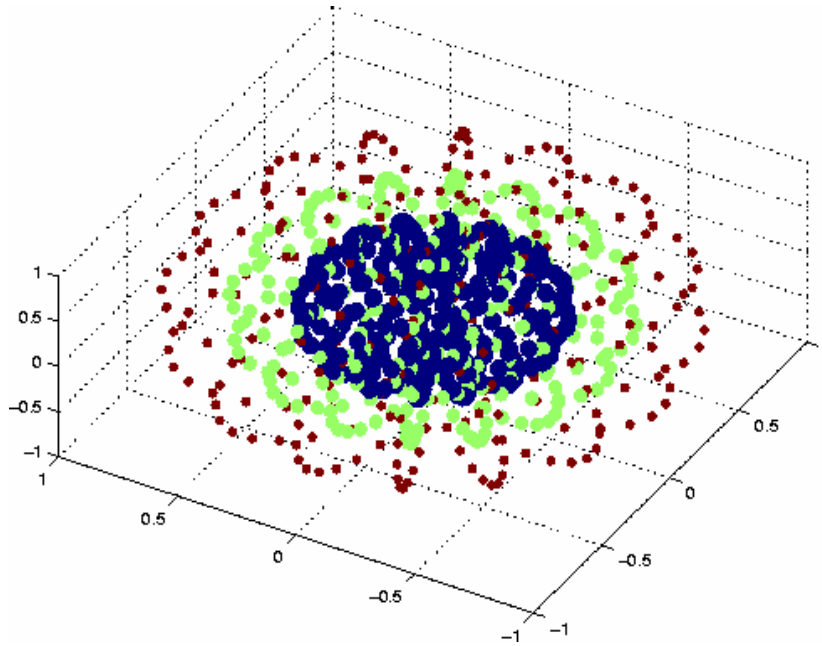


图 5-15　三维散点图

5.1.8　直方图

用 hist 函数绘直方图，显示数据的分布特征，其调用格式如下。

- n = hist(Y)　　将 Y 中的元素分到 10 个间隔相同的条形中，并返回每个条形中元素的个数。若 Y 是矩阵，则 hist 函数对每一列生成直方图。
- n = hist(Y,x)　　x 为向量，返回 Y 的分布。例如，若 x 为一个 5 元素向量，则 hist 函数将 Y 中的元素分配到 5 组条形中。
- n = hist(Y,nbins)　　nbins 为标量，使用 nbins 组条形。
- [n,xout] = hist(…)　　返回包含频数和条形位置的向量 n 和 xout。可以使用 bar(xout,n) 来绘直方图。
- hist(…)　　创建一个上面描述的直方图。hist 函数在 Y 的最小值和最大值之间沿 *x* 轴分配条形。

注意：Y 向量或 Y 矩阵某列中所有元素根据它们的数值范围进行分组。每一个组用一组条形表示。

直方图的 *x* 轴对应 Y 中值的范围。直方图的 *y* 轴显示落到组中的元素个数；所以，在任意条形组中，*y* 轴包括 0 到最大元素个数的范围。

直方图用添加阴影的图形对象创建。若希望改变图形的颜色，可以设置阴影属性。

【例 5-13】　创建服从高斯分布的数据的钟形直方图。

```
x = -2.9:0.1:2.9;
y = randn(1000,1);
hist(y,x)
```

绘制结果如图 5-16 所示。

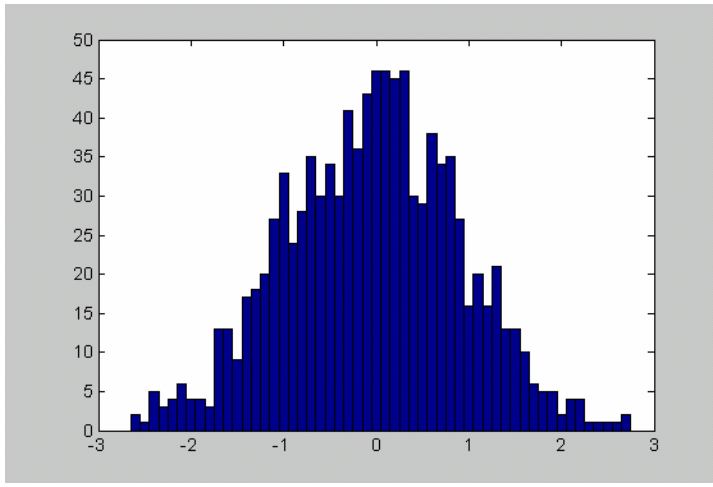


图 5-16　钟形直方图

5.2　功能图形绘制

部分图形主要应用于一定的专业，这里将它们称为功能图形。MATLAB 可以绘制彗星图、函数曲线图、帕累托图、玫瑰花图、罗盘图、火柴杆图、阶梯图、羽列图等多种功能图形。

5.2.1　彗星图

1．二维彗星图

彗星图为一快照图形，图中的小圆圈（代表彗星头部）跟踪屏幕上的数据点，彗星体为小圆圈后面的线段（就象彗星的尾部），尾部为跟踪整个函数的实线。用 comet 函数绘制二维彗星图。

用 comet 函数绘二维彗星图，其调用格式如下。

- comet(y)　　显示向量 y 的彗星图。
- comet(x,y)　　显示向量 y 和向量 x 的彗星图。
- comet(x,y,p)　　指定长度为 p*length(y)的彗星体。p 的默认值为 0.1。

【例 5-14】　创建一个简单的彗星图。

```
t = 0:.001:5*pi;
x = cos(3*t).*(cos(t).^2);
y = sin(3*t).*(sin(t).^2);
comet(x,y);
```

绘制结果如图 5-17 所示。

2．三维彗星图

用 comet3 函数绘三维彗星图，其调用格式如下。

- comet3(z)　　显示向量 z 的三维彗星图。
- comet3(x,y,z)　　显示穿过点[x(i),y(i),z(i)]的曲线的彗星图。
- comet3(x,y,z,p)　　指定彗星体的长度为 p*length(y)。

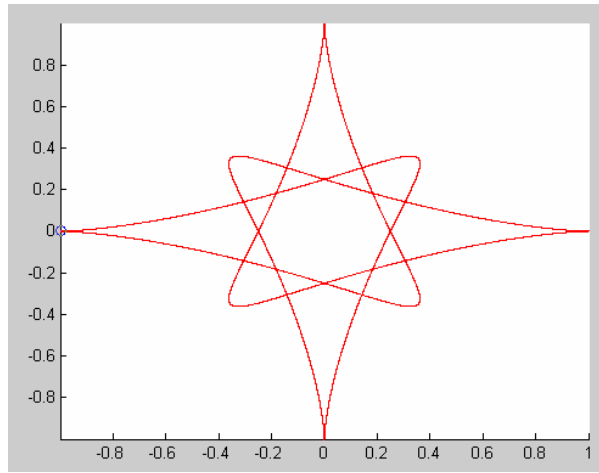


图 5-17　二维彗星图

【例 5-15】　创建一个三维彗星图。

```
t = -10*pi:pi/250:10*pi;
comet3((sin(t).^2).*cos(t),(cos(t).^2).*sin(t),t);
```

绘制结果如图 5-18 所示。

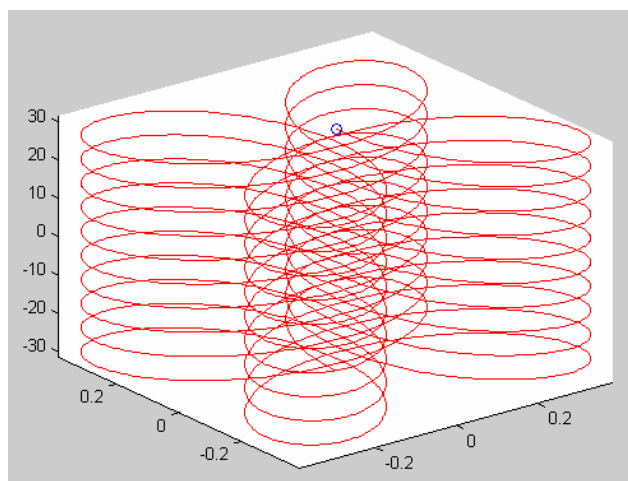


图 5-18　三维彗星图

5.2.2　函数曲线图

用 fplot 函数在指定的区间内绘函数的曲线图，其调用格式如下。

- fplot　　fplot 函数在指定区间内绘函数的曲线图。该函数必须具有 $y = f(x)$的形式，其中 x 为向量，它指定坐标范围，y 向量的大小与 x 的大小相同，包含 x 中点处的函数值。如果函数为给定的 x 返回一个以上的值，则 y 为矩阵，它的列包含 $f(x)$的每个元素。
- fplot('function',limits)　　在 limits 指定的范围内绘“ function ”指定的曲线图。limits 为向量，指定 x 轴的范围([xmin xmax])或 x 轴与 y 轴的范围([xmin xmax ymin ymax])。“ function ”必须为 M 文件的文件名或带有变量 x 的字符串。变量 x 将传给 eval，如“ sin(x) ”,“ diric(x,10) ”或“ [sin(x),cos(x)] ”。

函数 *f*(*x*)必须为 x 向量中的每个元素返回一个行向量。例如，若 *f*(*x*)返回[f1(x),f2(x),f3(x)]，则对于输入[x1;x2]，函数将返回下面的矩阵：

```
f1(x1) f2(x1) f3(x1)
f1(x2) f2(x2) f3(x2)
```

- fplot('function',limits,LineSpec)　　用 LineSpec 参数指定的线型绘“ function ”的图形。
- fplot('function',limits,tol)　　用 tol 指定的相对误差容限绘“ function ”函数指定的函数曲线图。其默认值为 2e-3，即 0.2%的精度。
- fplot('function',limits,tol,LineSpec)　　用 tol 指定的相对误差容限和 LineSpec 指定的线型、标记和颜色绘函数的曲线图。
- fplot('function',limits,n)　　用至少 n+1 个点绘函数的曲线图。n 的默认值为 1。最大步长值限定为(1/n)*(xmax-xmin)。
- fplot(fun,lims,…)　　接受选项参数 tol，n 和 LineSpec 的自由组合。
- [X,Y] = fplot('function',limits,…)　　将“ function ”的绝对值和阶次返回到 X 和 Y 中。
- […] = plot('function',limits,tol,n,LineSpec,P1,P2,…)　　可以将参数 P1, P2 等传递给函数“function”，即

  Y = function(X,P1,P2,…)

【例 5-16】　下面在[-2，2]范围内绘函数 tanh 的图形。

```
fplot('tanh',[-2 2])
```

创建一个 M 文件 graphf.m，返回如下两个列矩阵：

```
function Y = myfun(x)
Y(:,1) = 200*sin(x(:))./x(:);
Y(:,2) = x(:).^2;
```

用下面的命令行绘函数的图形：

```
fplot('graphf.m',[-20 20])
```

绘制结果如图 5-19 所示。

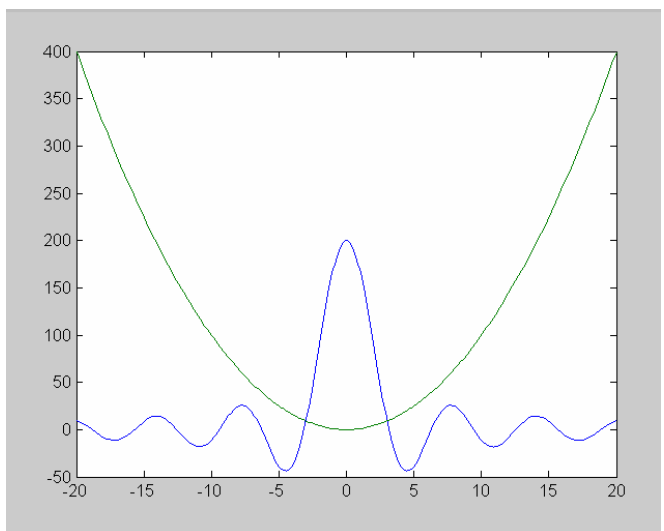


图 5-19　函数曲线图

5.2.3 帕累托图

用 pareto 函数绘帕累托图，其调用格式如下。

- pareto(Y)　　用 Y 中元素的编号标注条形。
- pareto(Y,names)　　用字符串矩阵或单元数组名标注条形。
- pareto(Y,X)　　用与 X 相关的值标注条形。
- H = pareto(…)　　返回阴影对象和直线对象句柄的组合。

【例 5-17】　根据有不同类型缺陷的产品个数数据绘制帕累托图。

```
defects = ['pits    ';'cracks';'holes ';'dents '];
quantity = [5 3 19 25];
pareto(quantity,defects)
```

绘制结果如图 5-20 所示。

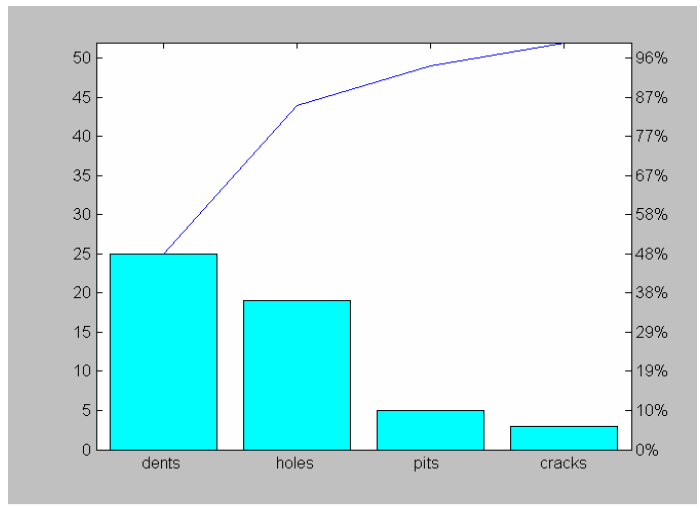


图 5-20　帕累托图

5.2.4 玫瑰花图

玫瑰花图是一个角度直方图，它是一个极坐标图，显示根据数值范围进行分组的数据的分布，每一组显示为一个扇区。

用 rose 函数绘玫瑰花图（角度直方图），其调用格式如下。

- rose(theta)　　生成一个角度直方图，显示扇区中心角为 20 度或更小时的数据分布。向量 theta 用弧度表示，确定从每个扇区开始的角度。每个扇区的长度反映 theta 中元素落入组内的个数，组的范围从 0 到所有扇区内元素的最大个数。
- rose(theta,x)　　使用向量 x 指定扇区的个数和位置。length(x)是扇区的个数，x 的值指定每个扇区的中心角。如，若 x 是一个 5 元素向量，则 rose 函数将 5 个扇区中 theta 的元素值集中到指定的 x 值处。
- rose(theta,nbins)　　在[0, 2*pi]的范围内绘 nbins 个间距相等的扇区。默认值为 20。
- [tout,rout] = rose(…)　　返回向量 tout 和 rout，这样 polar(tout,rout)函数生成数据的直方图。

【例 5-18】　创建一个玫瑰花图，显示 50 个随机数的分布。

```
theta = 2*pi*rand(1,50);
rose(theta)
```

绘制结果如图 5-21 所示。

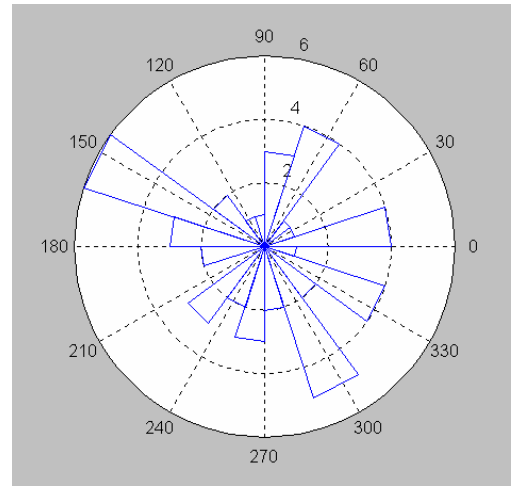


图 5-21　玫瑰花图

5.2.5　火柴杆图

1．二维火柴杆图

二维火柴杆图将数据显示为从 x 轴向外延伸的直线。直线末端有一个小圆圈（默认设置）或其他标记，其 y 坐标代表每个火柴杆终点的数据值。

用 stem 函数绘离散序列数据的火柴杆图，其调用格式如下。

- stem(Y)　　将数据序列 Y 绘成沿 x 轴等间距排列的火柴杆图。当 Y 为矩阵时，stem 函数用行中的所有元素对相同的 x 值绘图。
- stem(X,Y)　　用 X 与 Y 的列绘图。X 和 Y 为大小相同的向量或矩阵。另外，X 可以是行或列向量，Y 为具有 length(X)行的矩阵。
- stem(…,'fill')　　指定是否对火柴杆末端的小圆圈进行颜色填充。
- stem(…,LineSpec)　　指定火柴杆图的线型、标记和颜色。
- h = stem(…)　　返回直线图形对象的句柄。

【例 5-19】　创建 10 个随机数的火柴杆图。

```
y = linspace(0,2,10);
stem(exp(-y),'fill','-.')
axis ([0 11 0 1])
```

绘制结果如图 5-22 所示。

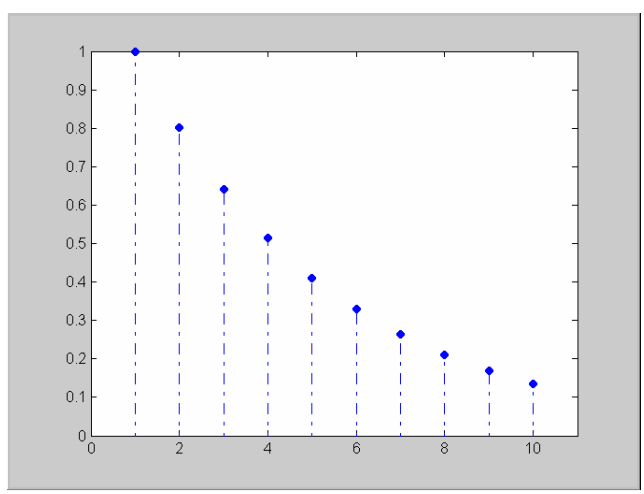


图 5-22　二维火柴杆图

2．三维火柴杆图

三维火柴杆图用 x-y 平面向上扩展的直线来表示数据，图中直线末端的小圆圈（默认设置）或其他标记表示每个火柴杆终止时的数据点。

用 stem3 函数绘离散序列数据的三维火柴杆图，其调用格式如下。

- stem3(Z)　　将数据序列 Z 表示为从 x-y 平面向上延伸的火柴杆。x 和 y 自动生成。
- stem3(X,Y,Z)　　在 X 和 Y 指定的数值点处绘数据序列 Z 的火柴杆图。X，Y 和 Z 必须都是相同长度的向量或相同大小的矩阵。
- stem3(…,'fill')　　指定是否在火柴杆末端的圆圈内进行颜色填充。
- stem3(…,LineSpec)　　为火柴杆指定线型、标记和颜色。
- h = stem3(…)　　返回直线图形对象的句柄。

【例 5-20】　创建一个三维火柴杆图，使下面的二变量函数可视化。

$$z = \sin(X) + \cos(Y)$$

```
X = linspace(0,1,10);
Y = X./2;
Z = sin(X) + cos(Y);
stem3(X,Y,Z,'fill')
view(-25,30)
```

绘制结果如图 5-23 所示。

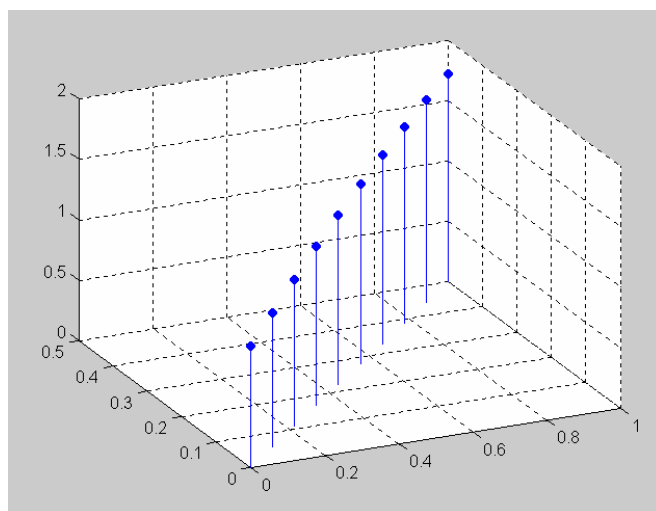


图 5-23　三维火柴杆图

5.2.6　阶梯图

用 stairs 函数生成阶梯图，其调用格式如下。

- stairs(Y)　　绘制 Y 中元素的阶梯图。当 Y 为向量时，x 轴的比例从 1 到 size(Y)。当 Y 为矩阵时，x 轴度量从 1 到 Y 的行数。
- stairs(X,Y)　　绘 X 与 Y 的列的阶梯图。X 和 Y 为相同大小的向量或相同大小的矩阵。另外，X 可以是行向量或列向量，Y 是一个有 length(X)行的矩阵。
- stairs(…,LineSpec)　　指定线型、标记和图形颜色。

- [xb,yb] = stairs(Y)和[xb,yb] = stairs(x,Y)　　不绘图形，但返回向量 xb 和 yb，然后用 plot(xb,yb)绘阶梯图。

【例 5-21】　创建余弦波的阶梯图。

```
x = 0:.25:10;
stairs(x,cos(x))
```

绘制结果如图 5-24 所示。

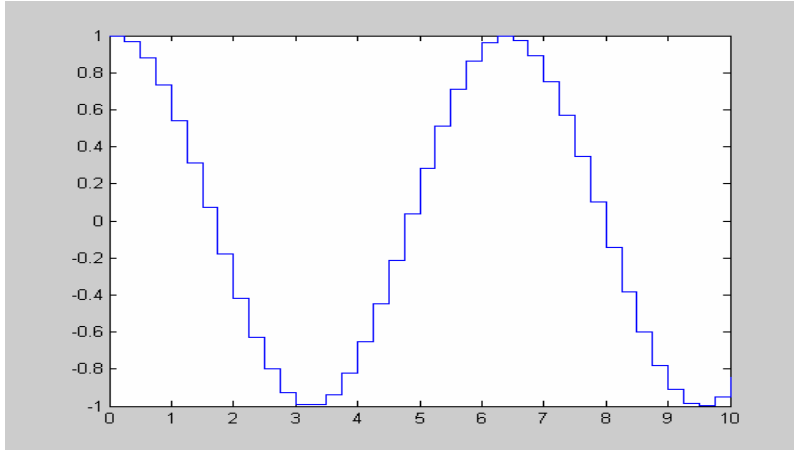


图 5-24　余弦波的阶梯图

5.2.7　罗盘图

罗盘图用从原点出发的箭头表示方向或速率向量。

用 compass 函数绘罗盘图，其调用格式如下。

- compass(X,Y)　　显示一个具有 *n* 个箭头的罗盘图，其中，*n* 为 X 或 Y 中的元素个数。每个箭头的起点位置为原点。箭头的顶点由[X(i),Y(i)]决定。
- compass(Z)　　显示具有 *n* 个箭头的罗盘图，其中，*n* 为 Z 的元素个数。箭头的起点为原点，顶点由 Z 的实部或虚部确定。该语法等价于 compass(real(Z),imag(Z))。
- compass(…,LineSpec)　　使用 LineSpec 参数指定的线型、标记和颜色绘罗盘图。
- h = compass(…)　　返回直线对象的句柄。

【例 5-22】　绘矩阵特征值的罗盘图。

```
Z = eig(randn(20,20));
compass(Z)
```

绘制结果如图 5-25 所示。

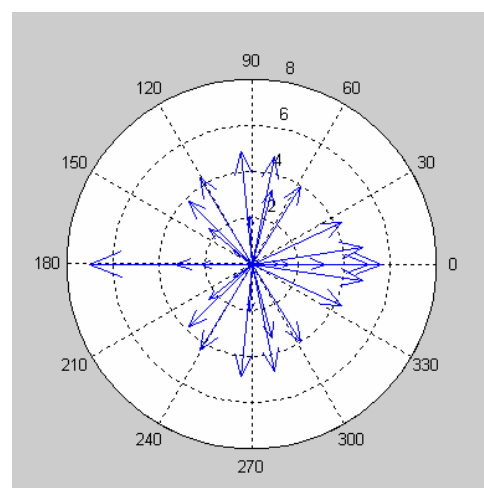


图 5-25　罗盘图

5.2.8　羽列图

羽列图显示源于水平轴上等距点的向量。

用 feather 函数绘羽列图（速率向量图），

其调用格式如下。

- feather(U,V)　　显示由 U 和 V 指定的向量，其中 U 包含作为相对坐标的 *x* 元素，V 包含作为相对坐标的 *y* 元素。
- feather(Z)　　显示由 Z 中的复数元素指定的向量。它等价于 feather(real(Z),imag(Z))。
- feather(…,LineSpec)　　使用 LineSpec 参数指定的线型、标注和颜色绘羽列图。

【例 5-23】　创建一个羽列图，显示 theta 的方向。

```
theta = (-180:10:180)*pi/180;
r = 3*ones(size(theta));
[u,v] = pol2cart(theta,r);
feather(u,v);
```

绘制结果如图 5-26 所示。

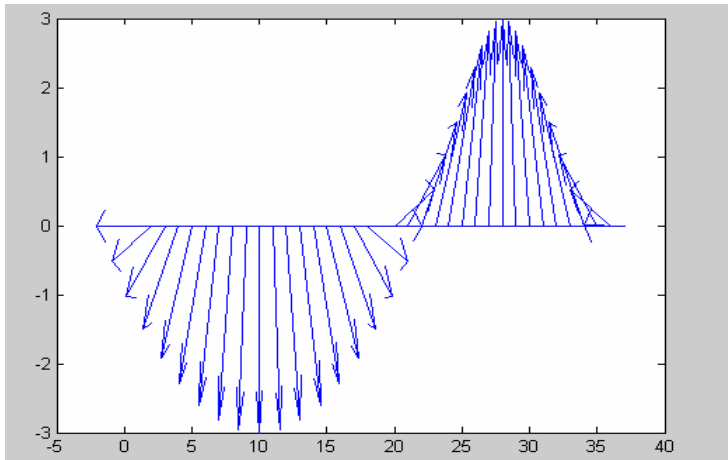


图 5-26　羽列图

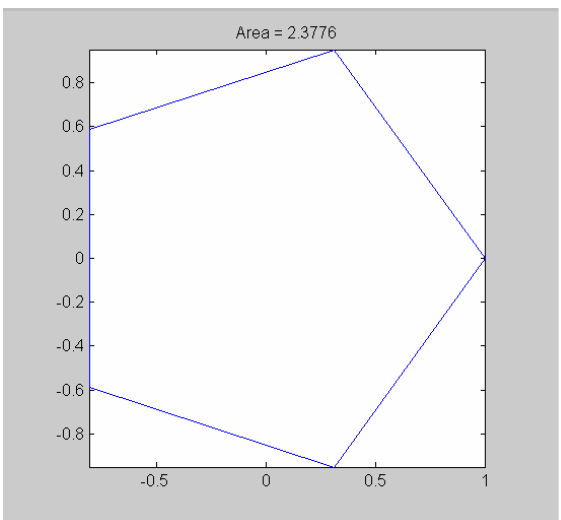


图 5-27　多边形面积图

5.2.9　多边形面积图

用 polyarea 函数计算多边形的面积，其调用格式如下。

- A = polyarea(X,Y)　　返回由 X 向量和 Y 向量指定的多边形的面积。
- A = polyarea(X,Y,dim)　　沿标量 dim 指定的维进行操作。

【例 5-24】　L = linspace(0,2.*pi,6); xv = cos(L)';yv = sin(L)';

```
xv = [xv ; xv(1)]; yv = [yv ; yv(1)];
A = polyarea(xv,yv);
plot(xv,yv); title(['Area = ' num2str(A)]);
axis image
```

绘制结果如图 5-27 所示。

5.3 特殊图形绘制

5.3.1 对数坐标图

在很多工程问题中，通过对数据进行对数转换可以更清晰地看出数据的某些特征。在对数坐标系中描绘数据点的曲线，可以直观地表现对数转换。对数转换有双对数坐标转换和单轴对数坐标转换两种。在 MATLAB 中，用 loglog 函数可以实现双对数坐标转换，用 semilogx 和 semilogy 函数可以实现单轴对数坐标转换。

用 loglog 函数绘制对数-对数比例图，其调用格式如下。

- loglog(Y)　　若 Y 的列值均为实数，则根据 Y 的列值和它们的对应编号绘图。若 Y 的列值为复数，则 loglog(Y)和 loglog(real(Y),imag(Y))等价，即根据 Y 各元素的实部和虚部数据绘图。
- loglog(X1,Y1,…)　　根据 Xn 和 Yn 匹配数据绘图。若 Xn 和 Yn 中只有一个为矩阵，则 loglog 函数绘制向量变量与矩阵行或列的配套数据的图，它取决于向量的行或列的维数是否与矩阵配套。
- loglog(X1,Y1,LineSpec,…)　　绘制所有由 Xn, Yn 和 LineSpec 等定义的线条。其中，LineSpec 确定线型、标记和图中直线的颜色。
- loglog(…,'PropertyName',PropertyValue,…)　　给 loglog 函数创建的所有直线对象设置属性值。
- h = loglog(…)　　返回句柄列向量到直线图形对象，一个句柄对应一条直线。

注意：当绘制一条以上的直线时，如果没有指定颜色，则 loglog 函数自动按当前轴指定的顺序循环使用颜色和线型。

【例 5-25】　表 5-1 中为某抽水井的水位降深资料，现根据该资料绘 lgs-lgt 曲线。

表 5-1　某抽水井的水位降深资料

累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)
0	0	13	0.222	36	0.242	670	0.370	1930	0.450
0.25	0.100	14	0.225	38	0.242	720	0.380	1990	0.450
0.5	0.120	15	0.227	40	0.245	790	0.390	2050	0.455
0.75	0.135	16	0.226	45	0.245	850	0.400	2110	0.455
1	0.160	17	0.226	50	0.250	910	0.395	2170	0.465
1.25	0.170	18	0.226	55	0.255	970	0.400	2230	0.465
1.5	0.180	19	0.226	60	0.258	1030	0.400	2290	0.460
2	0.185	20	0.228	75	0.265	1090	0.405	2350	0.465
3	0.190	21	0.230	90	0.280	1150	0.395	2410	0.470
3.5	0.200	22	0.230	100	0.285	1210	0.400	2530	0.470
4	0.205	23	0.232	130	0.295	1330	0.400	2590	0.475
4.5	0.215	24	0.232	160	0.305	1390	0.410	2650	0.480
5	0.220	25	0.235	190	0.315	1450	0.415	2710	0.475
6	0.215	26	0.235	250	0.325	1510	0.425	2770	0.480
7	0.217	27	0.235	310	0.345	1570	0.430	2830	0.495

续表

累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)
8	0.218	28	0.235	370	0.350	1630	0.435	2930	0.505
9	0.218	29	0.235	430	0.355	1690	0.435		
10	0.218	30	0.238	490	0.360	1750	0.440		
11	0.220	32	0.238	550	0.360	1810	0.450		
12	0.220	34	0.238	610	0.370	1870	0.445		

下面进行编程。

```
a=[0 0.25 0.5 0.75 1 1.5 2 3 4 4.5 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 32 34 36 38 40 45 50 55 60 75 90 100 130 160 190 250 310 370 430 490 550 610 670 720 790 850 910 970 1030 1090 1150 1210 1330 1390 1450 1510 1570 1630 1690 1750 1810 1870 1930 1990 2050 2110 2170 2230 2290 2350 2410 2530 2590 2650 2710 2770 2830 2930];
b=[0 0.100 0.120 0.135 0.160 0.170 0.180 0.185 0.190 0.200 0.205 0.215 0.220 0.215 0.217 0.218 0.218 0.218 0.220 0.220 0.222 0.225 0.227 0.226 0.226 0.226 0.226 0.228 0.230 0.230 0.232 0.232 0.235 0.235 0.235 0.235 0.235 0.238 0.238 0.238 0.242 0.242 0.245 0.245 0.250 0.255 0.258 0.265 0.280 0.285 0.295 0.305 0.315 0.325 0.345 0.350 0.355 0.360 0.360 0.370 0.370 0.380 0.390 0.400 0.395 0.400 0.400 0.405 0.395 0.400 0.400 0.410 0.415 0.425 0.430 0.435 0.435 0.440 0.450 0.445 0.450 0.450 0.455 0.465 0.460 0.465 0.470 0.470 0.475 0.480 0.475 0.480 0.495 0.505];
loglog(a,b);
grid on
```

绘制结果如图 5-28 **所示。**

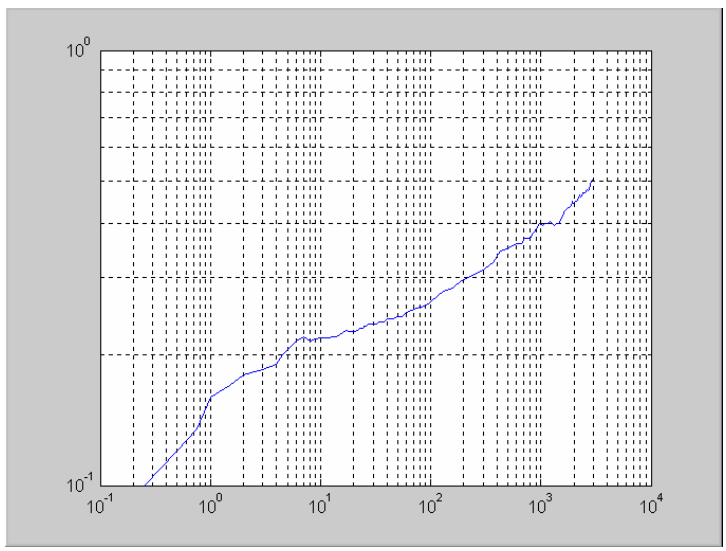


图 5-28　**双对数坐标图示例**

5.3.2　半对数坐标图

用 semilogx **和** semilogy **函数分别对** *x* **轴和** *y* **轴绘制半对数坐标数据图，其调用格式如下。**

- semilogx(Y)　　令 *x* 轴取以 10 为底的对数比例，*y* 轴取线性比例。如果 Y 的值为实数，则根据 Y 的列值和它们对应的编号绘图。如果 Y 的值为复数，则 semilogx(Y)函数等价于 semilogx(real(Y), imag(Y))。
- semilogx(X1,Y1,…)　　根据所有的 Xn 和 Yn 配对数据绘图。如果 Xn 和 Yn 中只有一个为矩阵，则 semilogx 函数绘制向量变量与矩阵的行或列的数据图。取行还是取列决定于向量的行还是向量的列的维数与矩阵相匹配。
- semilogx(X1,Y1,LineSpec,…)　　绘制所有由 Xn, Yn, LineSpec 等定义的直线。LineSpec 确定线型、标记和线条的颜色。
- semilogx(…,'PropertyName',PropertyValue,…)　　为所有由 semilogx 函数创建的直线图形对象设置属性值。
- semilogy(…)　　用以 10 为底的对数比例定义 *y* 轴，*x* 轴取线性比例，在该坐标系中绘制数据图。
- h = semilogx(…)和 h = semilogy(…)　　返回直线图形对象的句柄向量，一条直线对应一个句柄。

注意：当绘制一条以上的直线时，如果没有指定颜色和线型，则 semilogx 和 semilogy 函数自动按 ColorOrder 和 LineStyleOrder 的当前属性值的大小次序循环使用颜色和线型。可以将 Xn，Yn 匹配对与 Xn，Yn 和 LineSpec 三者混合使用。例如：

```
semilogx(X1,Y1,X2,Y2,LineSpec,X3,Y3)
```

【例 5-26】　某抽水井的水位降深资料如表 5-2 所示，要求绘出 S-lg(t)曲线。

表 5-2　某抽水井的水位降深资料

累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)	累计时间 (min)	水位降深 (m)
1	1.30	8	2.30	60	3.20	400	3.69
2	1.63	9	2.36	70	3.25	500	3.70
3	1.80	10	2.41	80	3.30	600	3.72
4	1.98	20	2.72	90	3.34	700	3.73
5	2.10	30	2.95	100	3.37	800	3.73
6	2.18	40	3.05	200	3.55	900	3.73
7	2.25	50	3.15	300	3.63	1000	3.73

在命令窗口输入下面的命令行：

```
t=[1 2 3 4 5 6 7 8 9 10 20 30 40 50 60 70 80 90 100 200 300 400 500 600 700 800 900 1000];
s=[1.30 1.63 1.80 1.98 2.10 2.18 2.25 2.30 2.36 2.41 2.72 2.95 3.05 3.15 3.20 3.25 3.30 3.34 3.37 3.55
3.63 3.69 3.70 3.72 3.73 3.73 3.73 3.73];
semilogx(t,s);
grid on
```

绘制结果如图 5-29 所示。

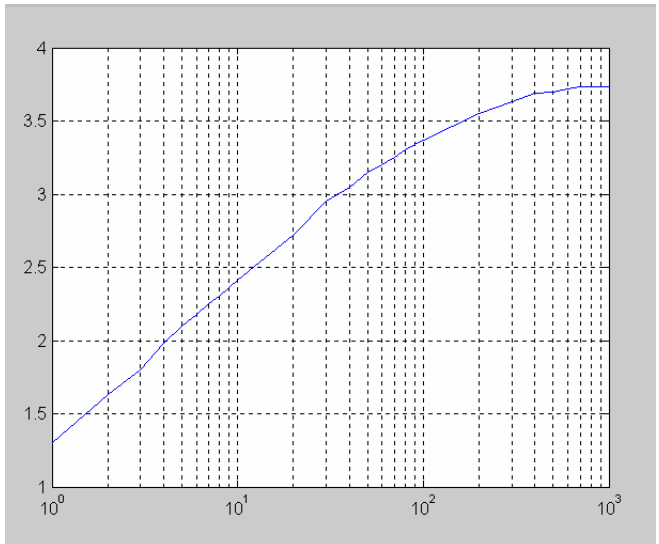


图 5-29　半对数坐标图示例

5.3.3　多轴线形图

在同一个图中绘制坐标度量单位不同的图形，可以使图形表达更加简练。在有些情况下，多轴图有利于数据对比。

利用 plotyy 函数可以绘制双轴图，其调用格式如下。

- plotyy(X1,Y1,X2,Y2)　　用标注在图形左侧的 *y* 轴单位绘 X1 和 Y1 的图形，用标注在图形右侧的 *y* 轴单位绘 X2 和 Y2 的图形。
- plotyy(X1,Y1,X2,Y2,'function')　　用字符串“function”指定的函数绘制每个图形。“function”可以是 plot, semilogx, semilogy, loglog, stem 或所有接受下面语法的 MATLAB 函数：

  h = function(x,y)
- plotyy(X1,Y1,X2,Y2,'function1','function2')　　对于图形左侧的坐标轴用 function1(X1,Y1)绘数据图，对于图形右侧的坐标轴用 function2(X2,Y2)绘数据图。
- [AX,H1,H2] = plotyy(…)　　返回 AX 中创建的两个坐标轴的句柄以及 H1 和 H2 中每个图形绘图对象的句柄。AX(1)为左侧轴，AX(2)为右侧轴。

【例 5-27】　本例用 plot 函数绘两个数学函数。

```
x = 0:0.01:20;
y1 = 200*exp(-0.05*x).*sin(x);
y2 = 0.8*exp(-0.5*x).*sin(10*x);
[AX,H1,H2] = plotyy(x,y1,x,y2,'plot');
```

绘制结果如图 5-30 所示。

可以使用 plotyy 函数返回的句柄标注坐标轴，并设置线型。用坐标轴句柄，可以指定图形左侧和右侧坐标轴的 Ylabel 属性：

```
set(get(AX(1),'Ylabel'),'String','Left Y-axis')
set(get(AX(2),'Ylabel'),'String','Right Y-axis')
```

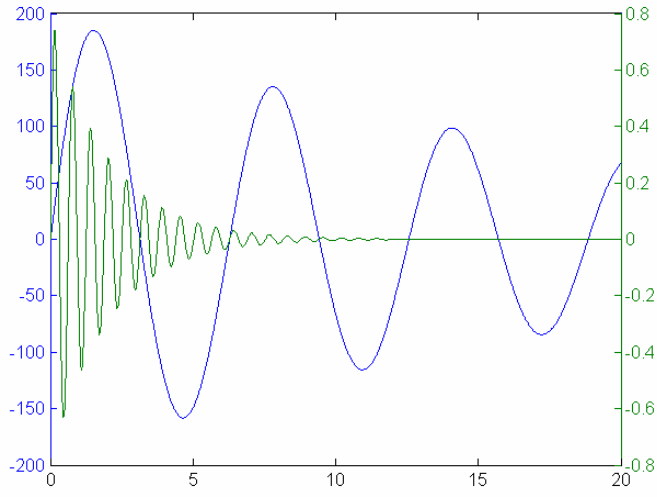


图 5-30　双轴曲线图

使用 xlabel 和 title 命令标注 *x* 轴和添加标题：

```
xlabel('Zero to 20 \musec.')
title('Labeling plotyy')
```

使用直线句柄设置对应于左侧和右侧纵轴的图形的 LineStyle 属性：

```
set(H1,'LineStyle','--')
set(H2,'LineStyle',':')
```

绘制结果如图 5-31 所示。

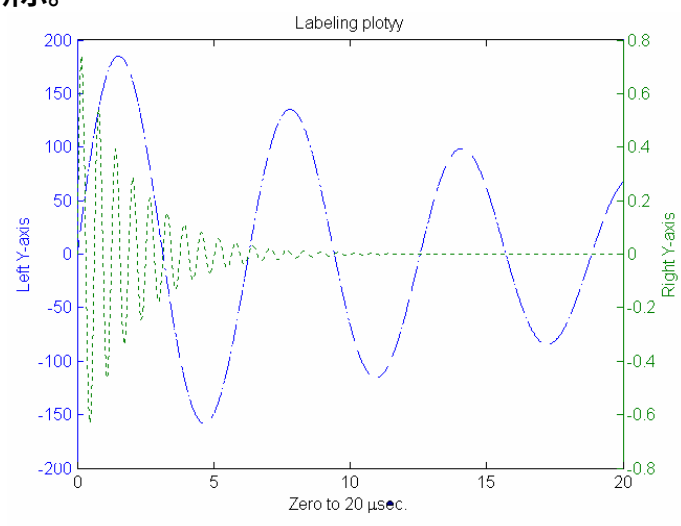


图 5-31　改变线型以后的双轴曲线图

5.3.4　极坐标图

用 polar 函数绘极坐标图，其调用格式如下。

- polar(theta,rho)　　**根据角度 theta 和半径 rho 创建极坐标图。**
- polar(theta,rho,LineSpec)　　LineSpec **指定极坐标图中直线的线型、标记和颜色。**

【例 5-28】　下面程序绘制一个简单的极坐标图。

```
t = 0:.01:2*pi;
polar(t,sin(2*t).*cos(2*t),'--r')
```

绘制结果如图 5-32 所示。

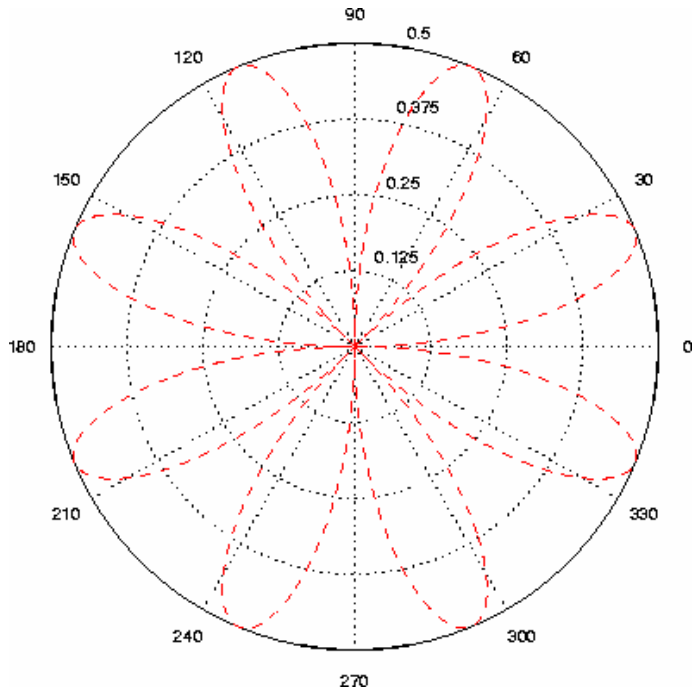


图 5-32　极坐标图

5.3.5　柱形图

用 cylinder 函数生成柱形图，其调用格式如下。

- cylinder　　生成 *x*, *y* 和 *z* 坐标系的单位圆柱图。可以用 surf 或 mesh 函数画图，或不提供输出变量直接画图。
- [X,Y,Z] = cylinder　　返回半径为 1 的圆柱的 *x*, *y* 和 *z* 坐标。
- [X,Y,Z] = cylinder(r)　　返回用 r 定义周长曲线的圆柱的三维坐标。cylinder 函数将 r 中的每个元素作为半径。
- [X,Y,Z] = cylinder(r,n)　　返回用 r 定义周长曲线的圆柱的三维坐标。在它周围有 n 个间隔点。
- cylinder(…)　　无输出变量，用 surf 函数绘圆柱图。

【例 5-29】　用随机涂色表面绘制圆柱图。

```
cylinder
axis square
h = findobj('Type','surface');
set(h,'CData',rand(size(get(h,'CData'))))
```

绘制结果如图 5-33 所示。

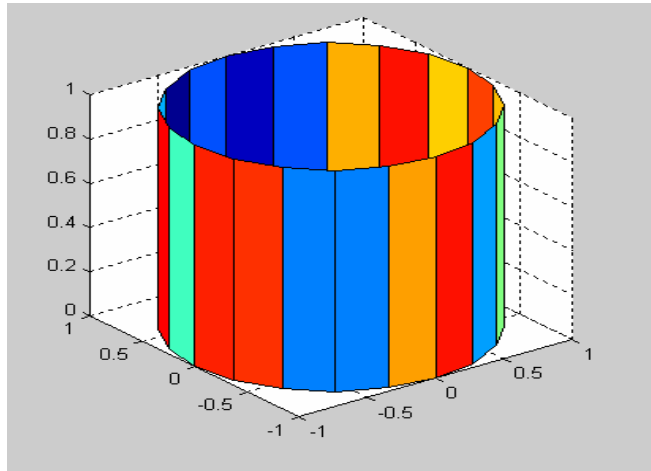


图 5-33　圆柱图之一

可以沿纵轴方向改变圆柱的半径，例如：

```
t = 0:pi/10:2*pi;
[X,Y,Z] = cylinder(2+sin(t));
surf(X,Y,Z)
axis square
```

结果生成图 5-34。

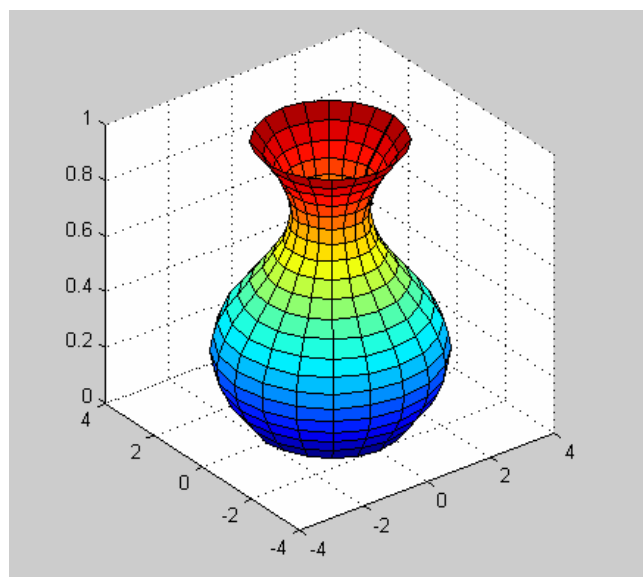


图 5-34　圆柱图之二

5.4　图形格式控制

5.4.1　添加标题

1．使用 Insert 菜单中的 Title 选项添加标题

在图形窗口中的菜单条中单击 Insert 菜单，然后选择 Title 选项。MATLAB 将在坐标轴的

顶端打开一个文本输入窗口，在其中可以输入标题名称。添加标题以后的图形效果如图 5-35 所示。

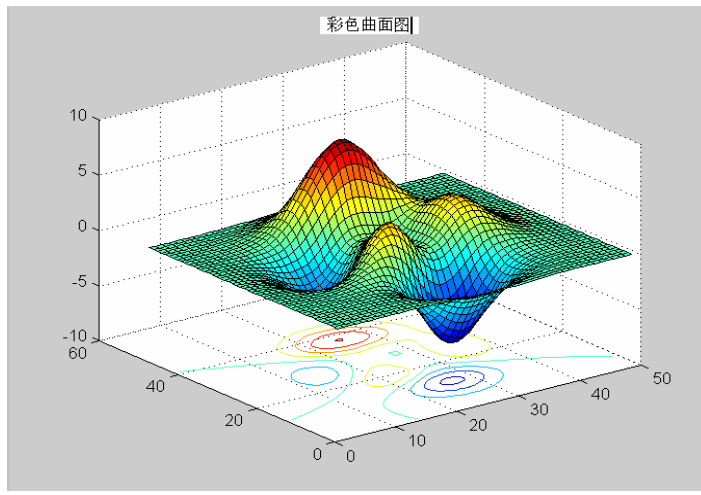


图 5-35　添加图形标题

输完文本以后，在图形背景域中单击任意一处，关闭标题周围的文本输入框。如果单击图中的另一对象，如坐标轴或直线，将关闭标题文本输入框，但同时自动选择单击的对象。

2．使用属性编辑器添加标题

使用属性编辑器可在图中添加标题。首先双击图中的坐标轴，打开属性编辑器，弹出如图 5-36 所示对话框。也可以通过用鼠标右键单击坐标轴，然后在打开的菜单中选择属性来打开属性编辑器。

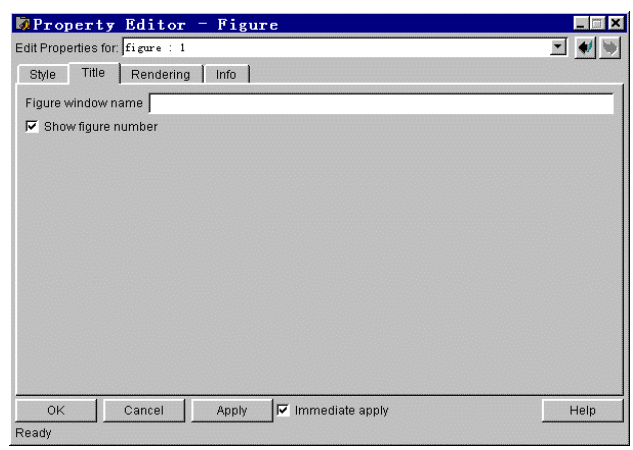


图 5-36　“Property Editor-Figure”对话框

选择“Title”选项卡，并在“Figure window name”文本框中键入标题，单击 Apply 按钮，就可以为当前图形添加标题了。

创建的标题为文本对象，所以，可以改变字体、粗细、位置和许多其他的格式。

3．使用 title 函数添加标题

使用 title 函数，可以在 MATLAB 命令行或 M 文件中给图形添加标题。该函数使你可以

在给图形添加标题时指定标题的属性值。

例如，下面的代码给当前坐标轴添加标题，并将 FontWeight 属性的值设置为 bold。

```
title('Lotka-Volterra Predator-Prey Population Model',…
    'FontWeight','bold')
```

用 set 函数，可以从 MATLAB 命令行或 M 文件编辑标题。

5.4.2 图例

1．添加图例

单击“ Insert ”菜单，并在打开的菜单条中选择“ Legend ”选项，将图例添加到图中。MATLAB 创建一个图例，将它放在图形的右上角，如图 5-37 所示。

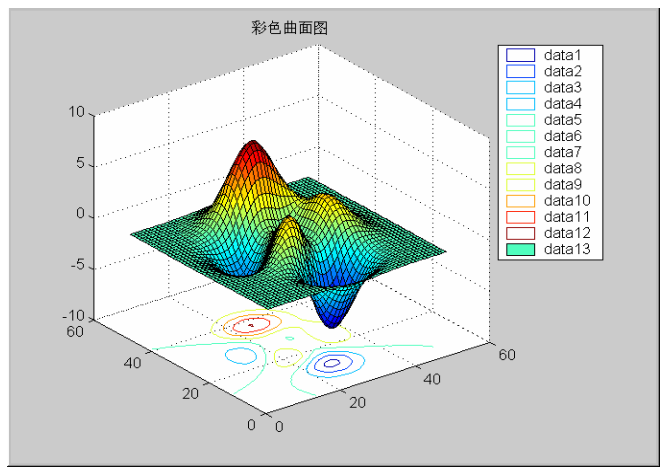


图 5-37　添加图例

也可以使用 legend 函数在 MATLAB 命令行或 M 文件中添加图例到图形。使用 legend 函数创建图例时，必须指定文本标签。

例如，下面的代码将一个图例添加到当前坐标轴中：

```
legend('Y1 Predator','Y2 Prey')
```

legend 函数需要指定有关图例的其他参数，如它的位置等。

2．图例的位置

根据图形编辑模式是否可用，可有两种方法在图中改变图例的位置。

如果图形编辑模式不可用，则将鼠标放到图例上方，按住鼠标左键不要放，MATLAB 将改变图标，指示可能的移动方向。将图例进行拖拉操作，可以将它放到图中任意位置。然后放开鼠标。

如果图形编辑模式可用，则在图例上用鼠标右键单击。该操作将选择图例并显示图例的上、下文菜单。从该菜单中选择“ Unlock Axes Position ”选项。

对于包含一条或多条直线对象的单轴对象，图例可以作为补充，代表图中图形的实例。一个或多个文本对象则代表图中每个数据集的标签。

3．用图形编辑模式编辑图例

激活图形编辑模式以后,在“ Edit Properties for ”下拉式列表框中选择“ legend: 'legend' ”选项，打开图 5-38 所示的“ Property Editor-Legend ”对话框，在其中进行设置，可以编辑组

成图例的每一个对象。

如果双击图例轴，属性编辑器将为轴对象显示属性面板集合。改变轴的属性值，然后单击“Apply”按钮。

如果双击图例中的文本标签，MATLAB 将打开一个包围所有文本标签的文本编辑框。用户可以在该编辑框中编辑图例中任意一个文本标签。

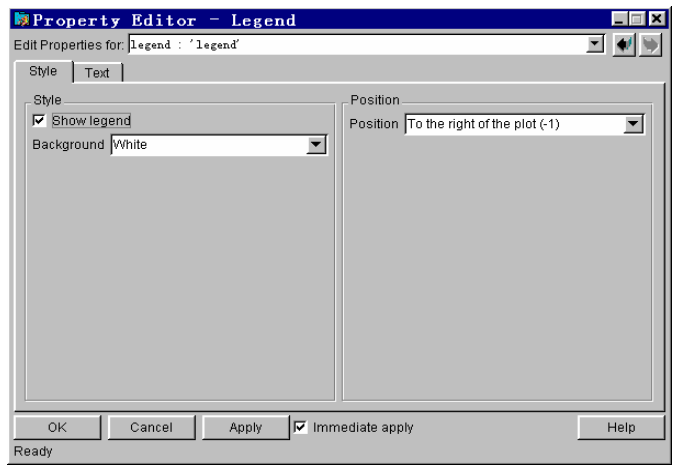


图 5-38 “Property Editor-Legend”对话框

4．当图形编辑模式不可用时编辑图例

当图形编辑模式不可用时，仍然可以编辑图例中的文本标签。双击图例中的文本标签，MATLAB 将打开一个包围选择文本标签的文本编辑框。所有其他文本标签将暂时隐藏。一次只能编辑一个文本标签。改变文本标签，然后在文本框以外的图内区域单击任意一处。MATLAB 自动改变图例框的大小来适应长标签或多行标签。

5．改变图例的大小

打开图形编辑模式，用鼠标右键单击图例轴，并从弹出式菜单中选择“Unlock Axes Position”选项，将图标移回到图例轴处。MATLAB 通过改变图标来指示可能的移动方向。

6．删除图例

如果已经激活了图形编辑模式，则可以通过在上面单击并在“Edit”菜单中选择“Cut”选项来删除图例。还可以通过鼠标右键单击并在弹出式菜单中选择“Cut”选项来删除图例。

如果图形编辑模式没有激活，则可以通过在“Insert”菜单中选择“Legend”选项来删除图例。

5.4.3 坐标轴标签

在 MATLAB 中，轴标签是文本字符串，分别与 *x* 轴、*y* 轴或 *z* 轴对齐。在图中添加轴标签可以帮助解释每个轴所代表的意义。

在图中添加轴标签，可以使用下面技巧中的任意一种：

- 在“Insert”菜单中使用“Label”选项；
- 使用属性编辑器；
- 使用标签命令。

1．在“Insert”菜单中使用“Label”选项

单击“Insert”菜单并选择希望标注的对应轴的“Label”选项。MATLAB 将在坐标轴附近打开输入轴标签的文本框，在该文本框中输入文本，作为坐标轴的标签，如图 5-39 所示。单击图形背景中的任意一处，关闭包围标签的文本入口框。

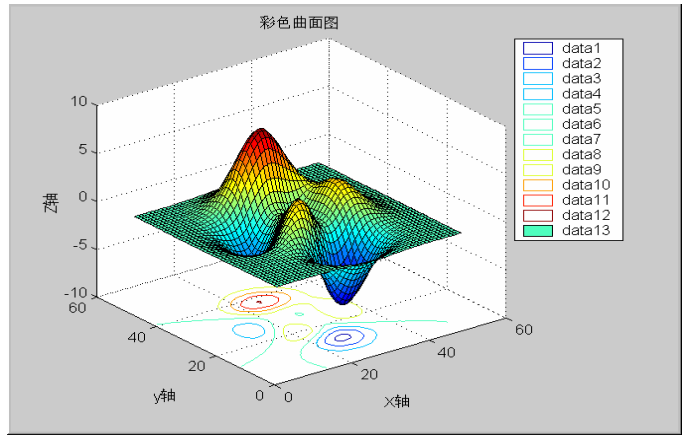


图 5-39　添加轴标签

2．使用属性编辑器添加轴标签

通过双击图形轴或单击轴然后从弹出式菜单中选择“Properties”选项，启动属性编辑器，弹出如图 5-40 所示对话框。属性编辑器将显示一系列的属性面板。选择标签面板，在对应文本框中输入标签文本，然后单击“Apply”按钮。

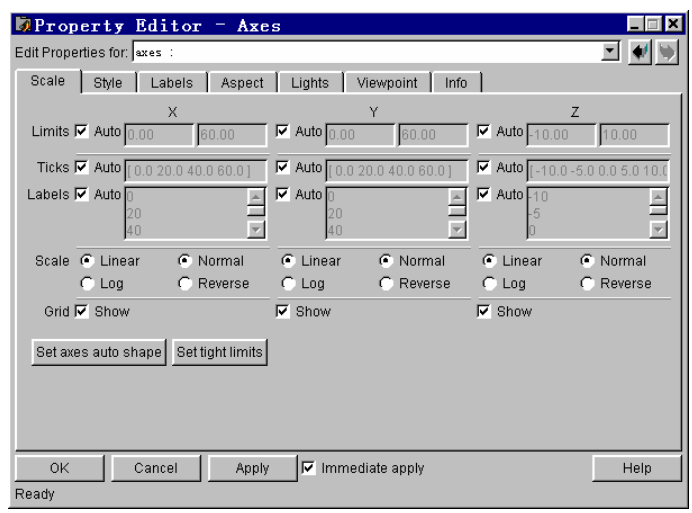


图 5-40　“Property Editor-Axes”对话框

3．使用 Label 命令

可以用 xlabel，ylabel 和 zlabel 命令添加 *x* 轴、*y* 轴和 *z* 轴标签。例如，下面的命令行标注坐标轴并添加标题。

```
xlabel('t = 0 to 2\pi','FontSize',16)
ylabel('sin(t)','FontSize',16)
```

```
title('\it{Value of the Sine from Zero to Two Pi}','FontSize',16)
```

标注命令自动将文本字符串放在合适的位置上。

5.4.4 文本的添加

可以在 MATLAB 图形中的任意位置添加自由格式的文本脚注，帮助解释数据内容或提醒别人注意数据集中的指定点。如果激活图形编辑模式，可以通过在图中或背景的某个位置用鼠标单击并输入文本来创建文本脚注。还可以从命令行中添加文本脚注。

使用图形编辑模式或 gtext 命令可以很容易地在图中任意位置放置文本脚注。希望在数据集指定点放置文本脚注时，使用 text 命令。

在图中添加文本脚注，首先单击“ Insert ”菜单，并选择文本选项或单击图形窗口工具条中的文本按钮。

注意：使用插入文本以后，图中的图形编辑模式被激活。在图中希望添加文本脚注的地方放置图标，并单击之。MATLAB 在图中该点处打开一个文本编辑框，然后输入文本。在图形背景中单击任意一处，关闭文本入口框。如果单击图中的另一对象，如图例轴或直线，将关闭标题文本入口框，但自动选择单击的对象。

也可以用 text 或 gtext 命令创建文本脚注。使用 text 函数创建文本脚注，必须指定文本及其在图中的位置，使用 *x*-*y* 坐标系。

【例 5-30】　下面的代码在指定点处创建文本脚注。

```
str1(1) = {'Many Predators;'};
str1(2) = {'Prey Population'};
str1(3) = {'Will Decline'};
text(7,220,str1)
str2(1) = {'Few Predators;'};
str2(2) = {'Prey Population'};
str2(3) = {'Will Increase'};
text(5.5,125,str2)
```

本例还演示如何创建带单元数组的多行脚注。

下面的代码在图上的 3 个数据点上添加脚注。

```
text(3*pi/4,sin(3*pi/4),…
    '\leftarrowsin(t) = .707',…
    'FontSize',16)
text(pi,sin(pi),'\leftarrowsin(t) = 0',…
    'FontSize',16)
text(5*pi/4,sin(5*pi/4),'sin(t) = -.707\rightarrow',…
    'HorizontalAlignment','right',…
    'FontSize',16)
```

可以使用文本对象在指定位置强制性地标注坐标轴。MATLAB 将文本放在坐标轴的数据单位倍数处。例如，假设绘制函数 $f(t)=0.25e^{-0.005t}$ 的图形。

```
t = 0:900;
plot(t,0.25*exp(-0.005*t))
t = 0:900;
```

```
plot(t,0.25*exp(-0.005*t))
```

如图 5-41 **所示。**

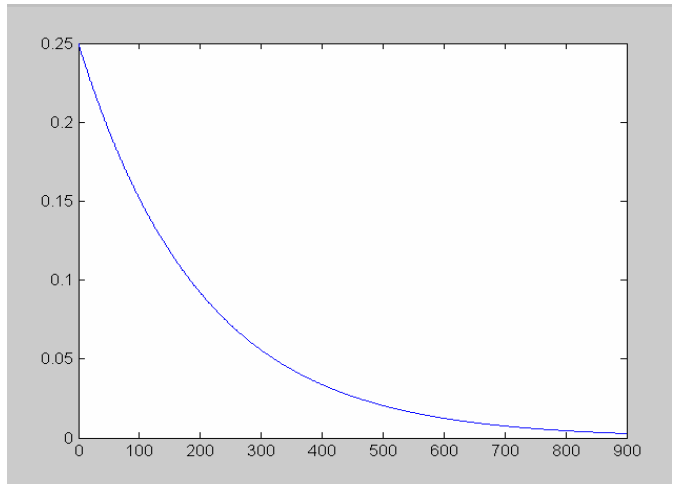


图 5-41　**原曲线图**

```
text(300,.25*exp(-0.005*300),…
'\bullet\leftarrow\fontname{times}0.25{\ite}^{-0.005{\itt}}
at {\itt} = 300',…
'FontSize',14)
```

在 *t*=300 **的值处标注点，用绘图函数计算文本坐标，如图** 5-42 **所示。该表达定义文本的位置属性为** *x* = 300**。**

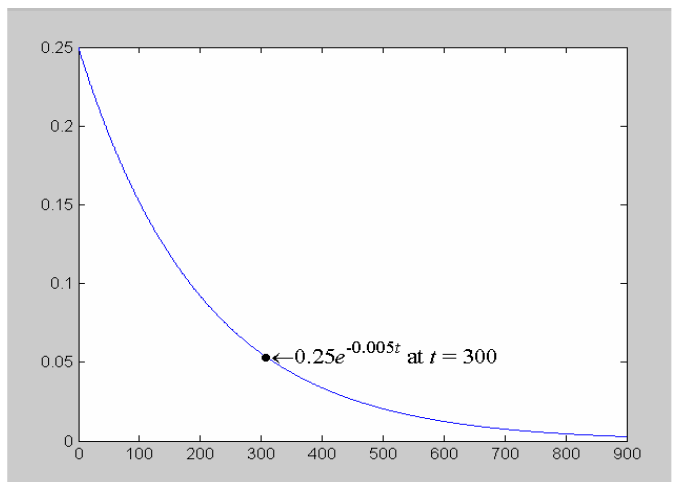


图 5-42　**在图中添加文本注释**

HorizontalAlignment **属性和** VerticalAlignment **属性控制相对于指定的** *x*，*y* **和** *z* **坐标的文本字符的对齐方式。默认的对齐方式为：**

```
HorizontalAlignment = left
VerticalAlignment = middle
```

MATLAB **并不将文本字符串精确地放在指定的位置。例如，前面显示了含有标注了文本**

的点的图形。缩小图形可以看见实际的文本位置。

【例 5-31】 下面是一个文本对齐的例子。假设现在希望用靠近点的文本来标注图中的最小值和最大值，并显示真实值。本例使用绘图数据确定图上文本和数值的位置。从 peaks 矩阵中取出一列进行绘图。

```
Z = peaks;
h = plot(Z(:,33));
```

绘制结果如图 5-43 所示。

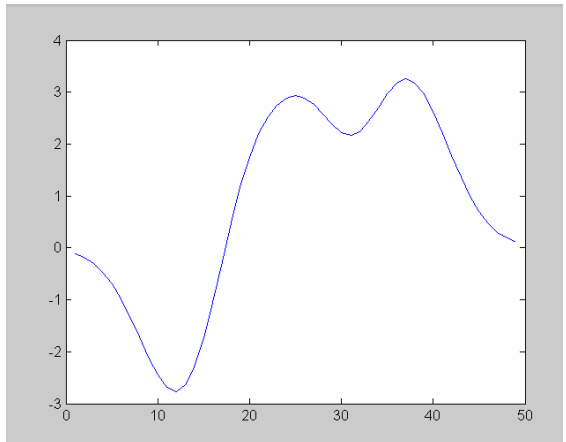


图 5-43 曲线图

第一步是找到最小值和最大值的指数，确定放置文本的坐标系，然后描述字符串。

```
x = get(h,'XData');
y = get(h,'YData');
imin = find(min(y) == y);
imax = find(max(y) == y);
text(x(imin),y(imin),[ 'Minimum = ',num2str(y(imin))],…
    'VerticalAlignment','middle',…
    'HorizontalAlignment','left',…
    'FontSize',14)
text(x(imax),y(imax),[ 'Maximum =    ',num2str(y(imax))],…
    'VerticalAlignment','bottom',…
    'HorizontalAlignment','right',…
    'FontSize',14)
```

标注结果如图 5-44 所示。

可以编辑图中的任何文本标签或脚注。首先启动图形编辑模式，然后双击字符串，或单击字符串后从弹出式菜单中选择“ String ”选项，文本周围出现一个编辑框，对文本进行修改。在文本编辑框外单击任意一处，结束文本编辑。

也可以在文本字符串中包含标记和希腊字母，如下例所示。

【例 5-32】

```
title('{\itAe}^{-\alpha\itt}sin\beta{\itt} \alpha<<\beta')
xlabel('Time \musec.')
```

ylabel('Amplitude')

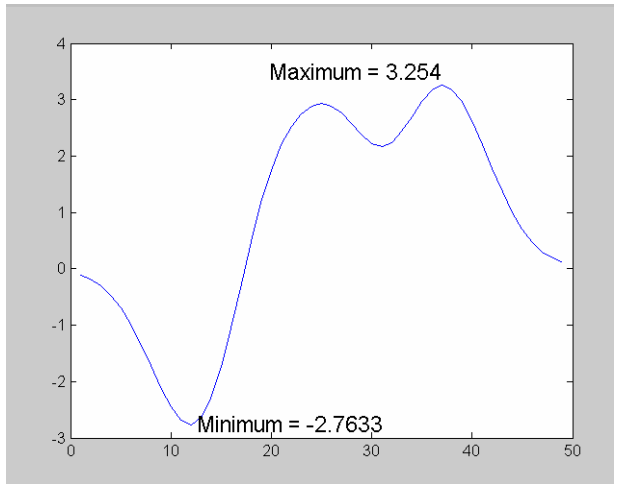


图 5-44　标注最大值和最小值

效果如图 5-45 所示。注意，所有文本字符序列前面都要添加符号"\"**。**

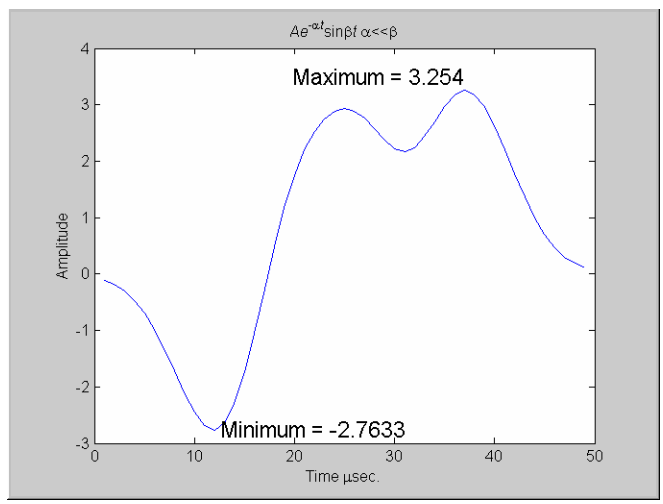


图 5-45　在文本字符串中包含标记和希腊字母

MATLAB **支持使用单元数组的多行文本。下面举一个用多行文本进行标注的例子。**

【例 5-33】

```
str1(1) = {'Center each line in the Uicontrol'};
str1(2) = {'Also check out the textwrap function'};
str2(1) = {'Each cell is a quoted string'};
str2(2) = {'You can specify how the string is aligned'};
str2(3) = {'You can use LaTeX symbols like \pi \chi \Xi'};
str2(4) = {'\bfOr use bold \rm\itor italic font\rm'};
str2(5) = {'\fontname{courier}Or even change fonts'};
plot(0:6,sin(0:6))
uicontrol('Style','text','Position',[80 80 250 65],…
```

'String',str1);

text(5.75,sin(2.5),str2,'HorizontalAlignment','right')

标注效果如图 5-46 **所示。**

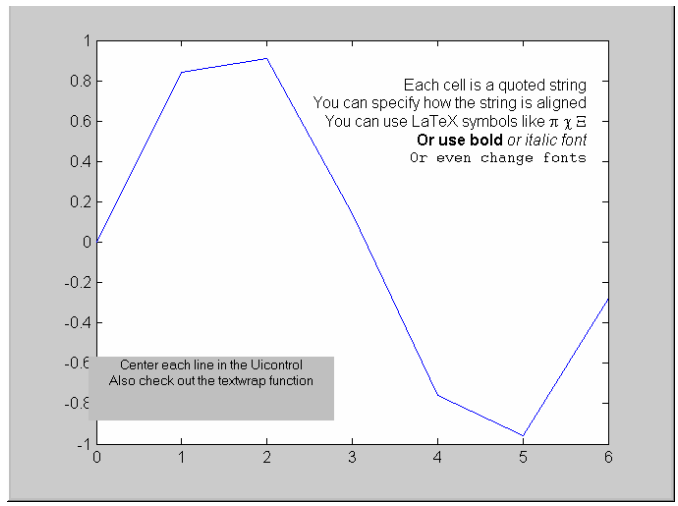


图 5-46　**添加多行文本**

5.4.5　基本数据统计量的添加

利用 MATLAB **数据统计工具，可以计算图中绘图数据与中心趋势和变异性有关的基本统计量，并在图中添加统计量标注。**

例如，下面的图形包含一个含有 *x* **值均值、***y* **值标准差的图形。使用下面的命令绘** census **数据（美国历史人群数据）的图形。**

load census

plot(cdate,pop,'+')

生成如图 5-47 **所示的图形。**

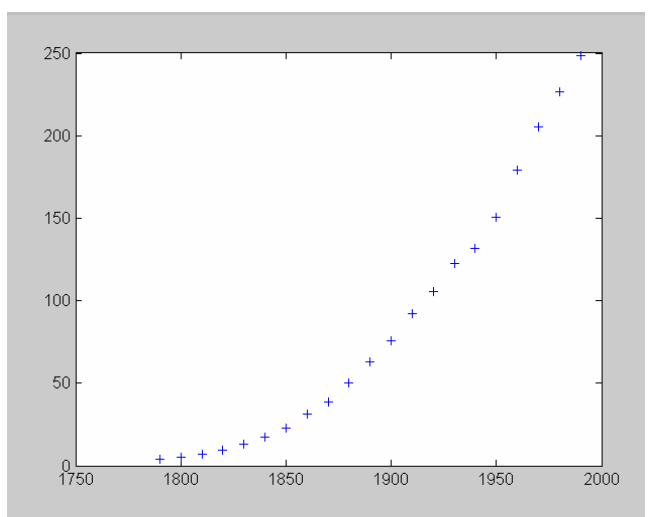


图 5-47　**数据曲线图**

从图形窗口中的“Tools”菜单中选择“Data Statistics”选项，打开“Data Statistics”对话框，如图 5-48 所示。

Data Statistics 工具在图中计算 x 数据和 y 数据的基本统计量，并在对话框中显示结果。

单击值 1890 和 78.6 后面的核选框，选择图中希望绘图的统计量。数据统计量工具在图中添加 x 均值和 y 标准差，如图 5-49 所示。

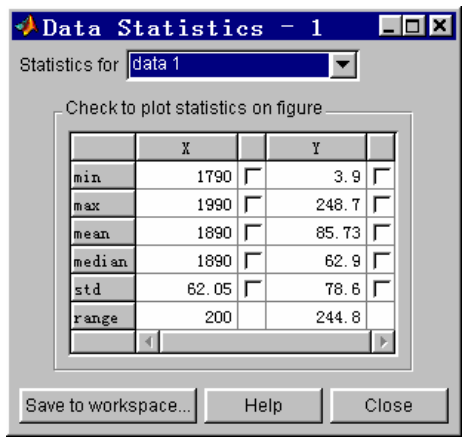


图 5-48　Data Statistics-1 对话框

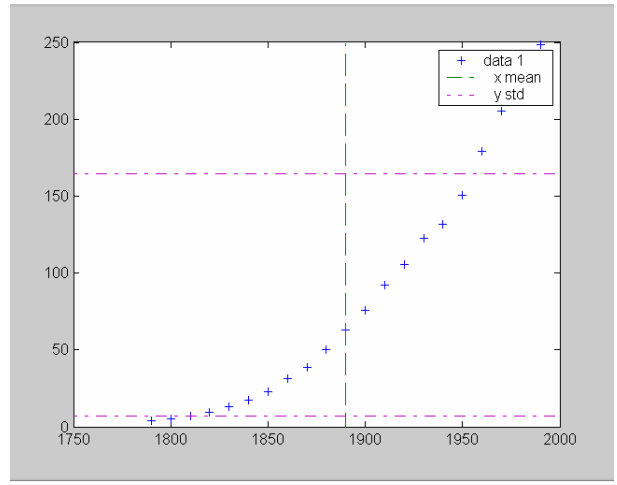


图 5-49　添加 x 均值和 y 标准差

5.5　图形属性控制

5.5.1　图形的缩放

利用 zoom 函数可以对二维图形进行放大或缩小。

用 zoom 函数放大或缩小二维图形的调用格式如下。

- zoom on　　启动交互缩放。当激活交互缩放以后，在坐标系内或坐标系外用鼠标单击可以对图形进行缩小或放大。单击鼠标左键，缩小图形，单击鼠标右键，放大图形。拖拉坐标轴将绘出一条橡皮筋线，放开鼠标以后，坐标轴的大小缩放到橡皮筋的位置。双击坐标轴返回到坐标轴的初始设置。
- zoom off　　取消交互缩放模式。
- zoom out　　将图形返回到初始缩放设置。
- zoom reset　　将当前缩放设置保存为初始缩放设置。以后调用 zoom out 或双击时返回到该缩放水平。
- zoom　　切换交互缩放状态。
- zoom xon 和 zoom yon　　分别设置 x 轴和 y 轴的缩放。
- zoom(factor)　　通过指定缩放因子来进行缩放，不影响交互缩放模式。如果值大于 1，则缩小该倍数，当值大于 0 但小于 1 时，放大 1/factor 倍。
- zoom(fig, option)　　上面选项中的任意一项可以用本语法进行指定。

5.5.2 网格显示控制

利用 grid 函数控制网格的显示。

用 grid 函数绘二维和三维图的网格线，其调用格式如下。

- grid　该函数确定是否是坐标轴的网格线。
- grid on　在当前坐标轴上添加网格线。
- grid off　从当前坐标轴上删除网格线。
- grid　切换网格可视状态。
- grid(axes_handle,…)　使用 axes_handle 指定的坐标轴代替当前坐标轴。

5.5.3 图形的叠加

利用 hold 函数可以控制图形的叠加，其调用格式如下。

- hold　该函数决定是否在当前图形中添加新的图形对象，或替换图中的对象。
- hold on　返回当前图和一定的坐标轴属性，使得接下来的图形命令可以添加到已存在的图形中。
- hold off　在绘新图形之前将坐标轴属性设置为默认属性。hold off 是默认的。
- hold　在添加图形和替换图形之间转换保留状态。

5.5.4 图形的颜色

用 colormap 函数设置和获取当前彩色图，其调用格式如下。

- colormap　是一个 $m\times3$ 的矩阵，其元素为 0 和 1 之间的实数。每一行为一 RGB 向量，定义一种颜色。第 k 行定义第 k 种颜色，其中，map(k,:) = [r(k) g(k) b(k)]) 指定红色、绿色和蓝色的密度。
- colormap(map)　将 colormap 设置为矩阵图形。如果 map 中的任何值位于区间[0，1]之外，则 MATLAB 返回下面的出错信息：

  error:　Colormap must have values in [0,1]。
- colormap('default')　设置当前彩色图为默认的彩色图。
- cmap = colormap;　提取当前彩色图值，返回值位于区间[0，1]内。

MATLAB 支持下面一些色图函数：

- autumn　从红色向橘黄色、黄色平稳过渡。
- bone　为灰阶色图，具有较高的蓝色组分。
- colorcube　试图提供更多灰阶、纯红、纯绿和纯蓝时，该函数可以包含 RGB 颜色空间中尽可能多的规则间隔的颜色。
- cool　由青色和洋红阴影组成的颜色。它在青色和洋红之间平滑过渡。
- copper　在黑色和亮铜色之间平滑过渡。
- flag　由红色、白色、蓝色和黑色组成。
- gray　返回线性灰阶色图。
- hot　在黑色、红色、橘红色、黄色和白色之间平滑过渡。
- hsv　变化色度-饱和度颜色模型中的色度组分。颜色从红色开始，然后为黄色、绿色、青色、蓝色、洋红，最后返回到红色。该色图特别适用于显示周期性函数。hsv(m)

与 hsv2rgb([h ones(m,2)])相同，其中 h 为线性坡度，h = (0:m-1)'/m。

- jet　在蓝色、青色、黄色、橘红色、红色之间过渡。
- lines　生成由坐标轴 ColorOrder 属性指定的色图和灰色阴影。
- pink　包含品红阴影。
- prism　重复红色、橘红色、黄色、绿色、蓝色和紫色等 6 种颜色。
- spring　由洋红和黄色阴影组成。
- summer　由绿色和黄色阴影组成。
- white　白色色图。
- winter　由蓝色和绿色阴影组成。

用 Shading 函数设置颜色阴影属性，其调用格式如下。

- shading　函数控制表面和填充图形对象的阴影。
- shading flat　每个网格线段和平面的颜色是一定的，它由线段末端或平面顶点的颜色值确定。
- shading faceted　在 shading flat 方式的基础上在图中添加黑色网格线。这是默认方式。
- shading interp　通过在直线或平面上对色图指数或真彩色进行插值来改变每个直线段和表面的颜色。

【例 5-34】　比较上面 3 种阴影设置方式在球体上的设置效果，如图 5-50 所示。

```
subplot(3,1,1)
sphere(16)
axis square
shading flat
title('Flat Shading')
subplot(3,1,2)
sphere(16)
axis square
shading faceted
title('Faceted Shading')
subplot(3,1,3)
sphere(16)
axis square
shading interp
title('Interpolated Shading')
```

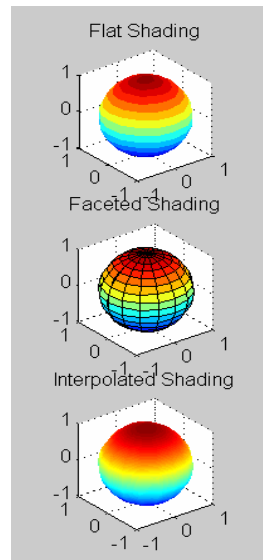


图 5-50　阴影设置

用 colorbar 函数显示颜色比例色条，其调用格式如下。

- colorbar　函数显示当前图中的当前色图并改变当前轴的大小以适应色条。
- colorbar　更新最新创建的色条或当当前轴没有色条时，colorbar 函数添加一个新的垂直色条。
- colorbar('vert')　添加一个垂直色条到当前轴。
- colorbar('horiz')　添加一个水平色条到当前轴。
- colorbar(h)　使用 h 创建色条。
- h = colorbar(…)　返回一个句柄到色条，它是一个轴图形对象。

- colorbar(…,'peer',axes_handle)　　创建一个与轴 axes_handle 相关的色条。

【例 5-35】　下面程序是在坐标轴旁边显示一个色条。

```
surf(peaks(30))
colormap cool
colorbar
```

效果如图 5-51 所示。

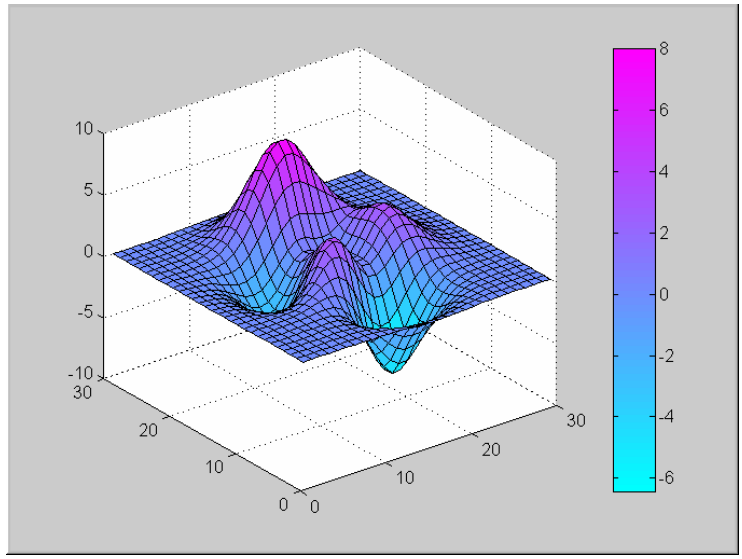


图 5-51　添加色条

5.6　坐标轴属性控制

5.6.1　标签属性

MATLAB 提供了很多标注和控制坐标轴外观的属性。下面的表面图显示了一些标注和属性控制的实例。

用指定的字符创建坐标轴。创建本坐标轴，需要为属性指定值。

```
h = axes('Color',[.9 .9 .9],…
        'GridLineStyle','--',…
        'ZTickLabels','-1|Z = 0 Plane|+1',…
        'FontName','times',…
        'FontAngle','italic',…
        'FontSize',14,…
        'XColor',[0 0 .7],…
        'YColor',[0 0 .7],…
        'ZColor',[0 0 .7]);
```

单个的坐标轴标签是文本对象，其句柄通常隐藏在命令行中（它们的 HandleVisibility 属性设置为 callback）。可以使用 xlabel，ylabel，zlabel 和 title 函数创建坐标轴标签。这些函数对当前坐标轴的影响较小。如果标注当前轴以外的其他坐标轴，则必须从对应的坐标轴属性

获得文本对象句柄。例如：

```
get(axes_handle,'XLabel')
```

返回文本对象句柄，作为 *x* 轴的标签。

下面的命令行为上面的坐标轴定义 *x* 轴和 *y* 轴的标签。

```
set(get(axes_handle,'XLabel'),'String','Values of X')
set(get(axes_handle,'YLabel'),'String','Values of Y')
set(get(axes_handle,'Title'),'String','\fontname{times}\itZ =
f(x,y)')
```

因为标签为文本，所以必须为字符串属性指定一个值，在初始情况下，它设置为空字符串（即没有标签）。

MATLAB 通过覆盖许多其他的文本属性来控制这些标签的位置和方位。可以设置 Color，FontAngle，FontName，FontSize，FontWeight 和 String 属性。

5.6.2 坐标轴的位置

坐标轴的 Position 属性控制图中坐标轴的大小和位置。默认的坐标轴与默认图形具有相同的方向比并填充除了边界以外的大部分图形。但是，可以像任何矩形一样定义坐标轴位置，并将它们放在图中任何想放的地方。

MATLAB 将坐标轴的 Position 属性定义为一个向量。

```
[left bottom width height]
```

left 和 bottom 定义图中的一个点，确定坐标轴矩形的左下角。width 和 height 指示坐标轴矩形的维数。

5.6.3 单个坐标轴的控制

创建图形时，MATLAB 自动确定坐标轴的范围、标记放置和标记标签。也可以通过设置合适的属性值来人工指定这些值。

给模式控制的属性指定值时，MATLAB 将模式设置为 manual，使用户可以进行自动覆盖。

可控制的坐标轴属性如表 5-3 所列。

表 5-3 坐标轴属性

属 性	目 标
XLim,YLim,ZLim	设置坐标轴范围
XLimMode YLimMode ZLimMode	指定坐标轴范围是由 MATLAB 自动确定还是人工确定
XTick YTick ZTic	沿坐标轴设置标记的位置
XtickMode YTickMode ZTickMode	指定标记位置是由 MATLAB 自动确定还是由人工指定

续表

属　性	目　标
XTickLabel YTickLabel ZTickLabel	指定坐标轴标记的标签
XtickLabelMode YTickLabelMode ZTickLabelMode	指定坐标轴标记的标签是由 MATLAB 自动确定还是由人工指定
XDir,YDir,ZDir	设置坐标值增加的方向

5.7 图形窗口控制

5.7.1 图形窗口的创建

用 figure 函数创建一个图形对象，其调用格式如下。

- figure　　用默认属性创建一个新的图形对象。
- figure('PropertyName',PropertyValue,…)　　用指定的属性值创建一个新的图形对象。
- figure(h)　　如果 h 为已经存在图形的句柄，则 figure(h)将使 h 对应的图形可见，并将该图形显示在所有其他图形的上方。当前图形是图形输出对象。如果 h 不是已经存在的图形的句柄，但它是一个整数，则 figure(h)创建一个图形，并指定其句柄为 h。
- h = figure(…)　　返回图形对象的句柄。

注意：为了创建一个图形对象，MATLAB 创建一个新的窗口，其特点是由默认的图形属性和指定的变量属性控制的。

可以将属性指定为属性名/属性值、结构数组和单元数组的形式。

gcf 命令返回当前图形的句柄并可以作为一个变量应用于 set 命令和 get 命令。

下面的程序创建一个图形窗口，其大小为屏幕窗口的四分之一并放在屏幕的左上角，使用 root 对象的 ScreenSize 属性确定其大小。ScreenSize 为一个四元素向量。

```
[left, bottom, width,height]:
scrsz = get(0,'ScreenSize');
figure('Position',[1 scrsz(4)/2 scrsz(3)/2 scrsz(4)/2])
```

5.7.2 图形的刷新和清除

用 refresh 函数重绘当前图形，其调用格式如下。

- refresh　　该函数删除并重绘当前图形。
- refresh(h)　　重绘由 h 确定的图形。

用 clf 函数清除当前图形窗口，其调用格式如下。

- clf　　从当前图形窗口中删除所有的句柄没有隐藏的图形对象（即它们的 HandleVisibility 属性设置为 on）。
- clf reset　　不论 HandleVisibility 属性如何设置，从当前图形窗口中删除所有的图形对象，并重新设置除了 Position, Units, PaperPosition 和 PaperUnits 以外的属性。

5.7.3 关闭图形窗口

用 close 函数删除指定的图形，其调用格式如下。

- close　　删除当前的图形（与 close(gcf)等价）。
- close(h)　　删除 h 指定的图形。如果 h 为一个向量或矩阵，close 删除所有由 h 指定的图形。
- close name　　删除指定名称的图形。
- close all　　删除所有句柄没有隐藏的图形。
- close all hidden　　删除所有隐藏句柄的图形。
- status = close(…)　　如果指定的窗口已经删除，则返回 1，否则返回 0。

第 6 章　科学计算可视化

6.1　概述

科学计算，特别是一些大型计算如数值模拟等涉及到大量的数据处理，如何使研究对象的几何形体可视、对象在内外因作用下的演变行为和机制可视以及最终结果可视，是科学计算工作者长期以来一直希望解决的问题。随着计算机科学技术的发展，这些问题已经在一定程度上得到了解决。采用一些先进的算法，甚至可以用具有高度真实感的实体模型来模拟科学计算的过程和结果。发展到现在，科学计算可视化已经是一门新的交叉学科了。

MATLAB 提供了比较完备的科学计算可视化函数。利用这些函数，可以绘制二维/三维的矢量图、等值线图、剖面图以及流线图、流锥图、流管图、流带图和卷曲图等表现流动特征的图形；利用快照技术，可以实现动画。而且，在 MATLAB 中还可以将各种图形组合起来使用，具有更强的表现效果。

6.2　等值线图

等值线图通过将空间一定范围内数值相等的点依次连线来达到展示数据分布的效果。这种图在地质、气象、力学等领域有着广泛的应用，特点是简便、直观，而且，二维等值线图还能表现三维信息。比如地形图就是等值线图的一种具体应用。利用等值线的疏密程度可以判断地形坡度的陡缓，利用等值线的标注可以了解某个位置的高程（海拔高度）。在 MATLAB 中，可以作二维等值线图，也可以作三维等值线图。

6.2.1　二维等值线图

用 contour 函数生成二维等值线图，其调用格式如下。

- contour(Z)　　绘矩阵 Z 的等值线图，其中，Z 可解释为 *x*-*y* 平面的高度。Z 必须至少是 2×2 的矩阵。等值线的水平数和等值线的水平值根据 Z 的最小值和最大值自动选取。*x* 轴和 *y* 轴的范围分别为[1:n]和[1:m],其中[m,n] = size(Z)。
- contour(Z,n)　　根据 Z 矩阵的数据绘有 n 个水平数的等值线图。
- contour(Z,v)　　根据向量 v 指定的数据值绘矩阵 Z 的等值线图。等值线的水平值等于 length(v)。绘水平数为 i 的的单条等值线时，使用 contour(Z,[i i])。
- contour(X,Y,Z)，contour(X,Y,Z,n)和 contour(X,Y,Z,v)　　绘 Z 的等值线图。X 和 Y 指定 *x* 轴和 *y* 轴的范围。当 X 和 Y 为矩阵时，它们必须与 Z 具有相同的大小。
- contour(…,LineSpec)　　用参数 LineSpec 指定的线型和颜色绘等值线。contour 函数忽略标记。
- [C,h] = contour(…)　　返回等值线矩阵 C 和图形对象的句柄向量。

【例 6-1】　利用 peaks 数据，在命令行键入：

```
contour(peaks(40),20);
```

得等值线图如图 6-1 所示。其中等值线的水平数为 20。

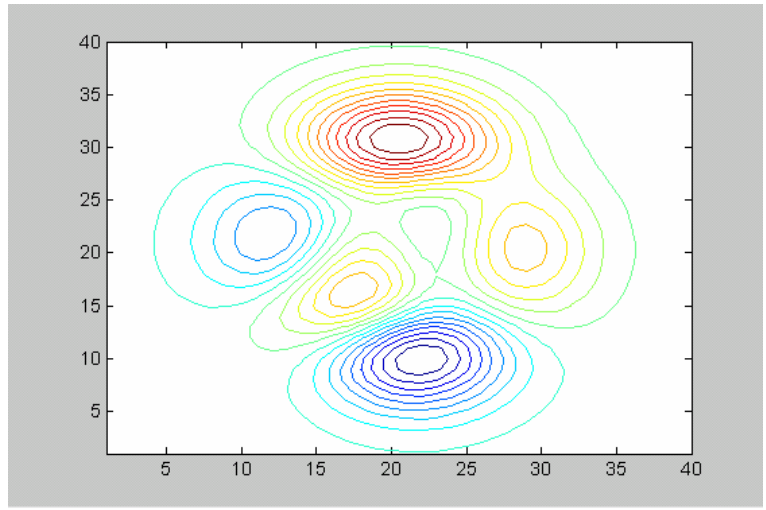


图 6-1　等值线图

对于上面同样的函数和范围，设置等值线的水平数为 30，重新设置颜色，生成的加密后的等值线图如图 6-2 所示。

```
contour(peaks(40),30);
Colormap(cool)
```

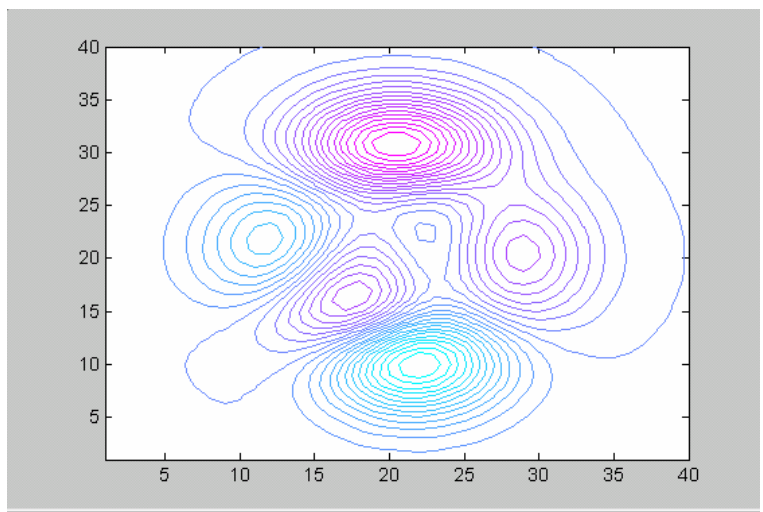


图 6-2　加密后的等值线图

使用 interp2 函数和 contour 函数创建平滑化的等值线图如图 6-3 所示。

```
P = Peaks(40);
contour(interp2(P,4));
```

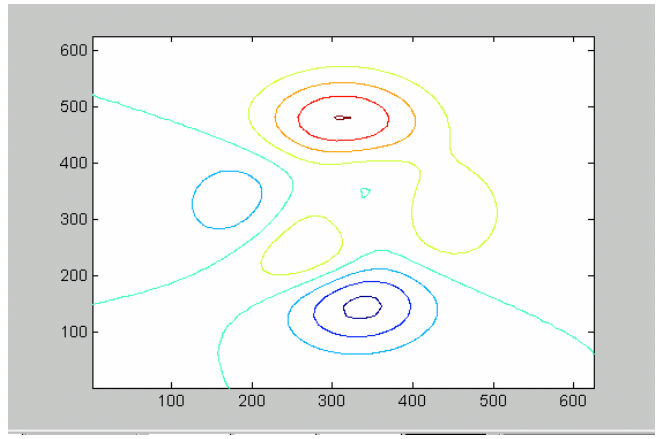


图 6-3　平滑化后的等值线图

6.2.2　等值线的标注

用 clabel 函数在等值线图中进行标注，其调用格式如下。

- clabel(C,h)　　旋转标签并将它们插到等值线中。该函数只插入那些在等值线图中合适的标签，它决定于等值线图的大小。
- clabel(C,h,v)　　v 向量给定等值线的水平值，创建标签，然后旋转标签并将它们插到等值线中去。
- clabel(C,h,'manual')　　将等值线标签放到鼠标选定的位置。单击鼠标左键，在最靠近图标中心位置下方的位置上进行标注。当图标处于图形窗口中时，单击回车键终止标注。旋转标签并插入到等值线图中。
- clabel(C)　　根据等值线结构参数 C（contour 函数的输出）的值把标签添加到当前等值线图中。该函数标注所有显示的等值线并随机选择标注位置。
- clabel(C,v)　　只标注向量 v 中给定的等值线的水平值。
- clabel(C,'manual')　　在鼠标选定的位置上放置等值线标签。

【例 6-2】　下面程序创建、绘制并标注简单等值线图，效果如图 6-4 所示。

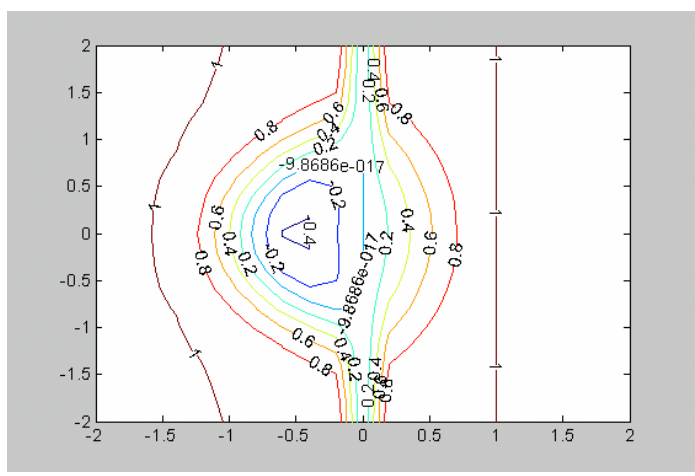


图 6-4　等值线图的标注

```
[x,y] = meshgrid(-2:.2:2);
z = x.^exp(-x.^2-y.^2);
[C,h] = contour(x,y,z);
clabel(C,h);
```

6.2.3 等值线填充

用 contourf 函数填充二维等值线图，其调用格式如下。

- contourf(Z)　　绘矩阵 Z 的等值线图，其中 Z 为平面的高度。Z 必须至少为 2×2 的矩阵。等值线的水平数和等值线的水平值自动选择确定。
- contourf(Z,n)　　绘一有 n 个等值水平数的矩阵 Z 的等值线图。
- contourf(Z,v)　　绘向量 v 指定的水平值的矩阵 Z 的等值线图。
- contourf(X,Y,Z)，contourf(X,Y,Z,n)和 contourf(X,Y,Z,v)　　用 X 和 Y 确定 *x* 轴和 *y* 轴的范围，生成 Z 的等值线图。当 X 和 Y 为矩阵时，它们必须与 Z 的大小相同。
- [C,h,CF] = contourf(…)　　返回等值线矩阵 C、句柄向量 h 和填充区域的等值线矩阵 CF。

【例 6-3】　下面程序创建 peaks 函数的填充等值线图，效果如图 6-5 所示。

```
Contourf(peaks(40),20);
colormap cool
```

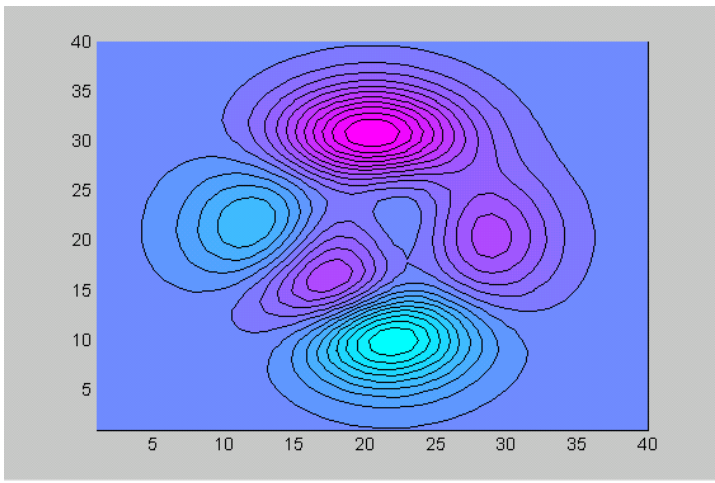


图 6-5　填充的等值线图

6.2.4 三维等值线图

用 contour3 函数绘制三维等值线图，其调用格式如下。

- contour3(Z)　　绘矩阵 Z 的三维等值线图。Z 可以看作是 *x*-*y* 平面以上的高程，它必须至少是一个 2×2 的矩阵。等值线的水平数和等值线的水平值自动选择。*x* 轴和 *y* 轴的范围分别为[1:n]和[1:m]，其中[m,n] =size(Z)。
- contour3(Z,n)　　绘 Z 矩阵的具有 n 个水平数的三维等值线图。
- contour3(Z,v)　　绘矩阵 Z 的三维等值线图，等值线位于向量 v 指定的值处。等值线的水平数等于 length(v)。使用 contour(Z,[i i])，可以只绘水平数为 i 的等值线。

- contour3(X,Y,Z), contour3(X,Y,Z,n)和 contour3(X,Y,Z,v)　　使用 X 和 Y 定义 *x* 轴和 *y* 轴的界限。如果 X 为一矩阵，则 X(1,:)定义 *x* 轴；如果 Y 为一矩阵，则 Y(:,1) 定义 *y* 轴。当 X 和 Y 均为矩阵时，它们必须与 Z 具有相同的大小，此时它们将指定一个表面。
- contour3(…,LineSpec)　　用参数 LineSpec 指定的线型和颜色绘等值线。
- [C,h] = contour3(…)　　返回等值线矩阵 C 和包含图形对象句柄的列向量。除非指定 LineSpec 属性（此时 contour3 函数创建直线图形对象），contour3 函数将创建阴影图形对象。

注意：如果没有指定 LineSpec 属性，colormap 和 caxis 将控制颜色。如果 X 或 Y 的间隔不确定，则 contour3 函数用确定间隔的等值线网格计算等值线，然后将数据转换到 X 或 Y。

【例 6-4】　下面的程序用来创建一个函数的三维等值线图并在上面叠加一个表面图，以增强函数的可视性，效果如图 6-6 所示。

```
[X,Y,Z]=peaks(40);
contour3(X,Y,Z,30)
surface(X,Y,Z,'EdgeColor',[.8 .8 .8],'FaceColor','none')
grid off
view(-15,25)
```

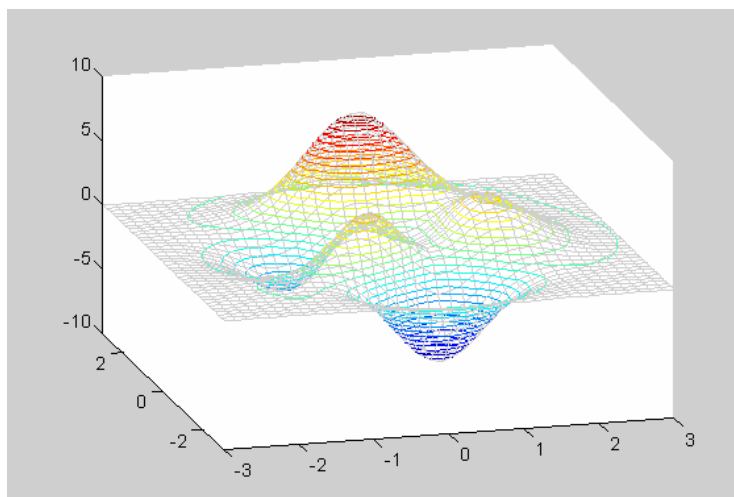


图 6-6　三维等值线图

6.3　向量图

向量图用箭头显示图中各点处的向量大小和方向。其中，箭头指示的方向为向量的方向，箭头的长短表示向量的大小。

6.3.1　二维向量图

用 quiver 函数绘制向量或速率图，其调用格式如下。

- quiver(U,V)　　在由 *x* = 1:*n* 和 *y* = 1:*m* 确定的坐标系中绘 U 和 V 指定的向量图，其中 [m,n] = size(U) = size(V)。

- quiver(X,Y,U,V)　　对每个 X 和 Y 配对数据绘向量图。X 和 Y 为向量，length(X) =n 且 length(Y) = m，其中[m,n] = size(U) = size(V)。 向量 X 对应于 U 和 V 的列，向量 Y 对应于 U 和 V 的行。
- quiver(…,scale)　　自动对向量设置显示比例。如果 scale=2，则将原向量的长度乘以 2 以后显示。若 scale=0，则不自动设置显示比例。
- quiver(…,LineSpec)　　用参数 LineSpec 指定线型、标注和颜色。quiver 函数在向量原点处绘标记。
- quiver(…,LineSpec,'filled')　　填充由 LineSpec 指定的标记。
- h = quiver(…)　　返回直线句柄向量。

注意：如果 X 和 Y 为向量，该函数为：

```
[X,Y] = meshgrid(x,y)
quiver(X,Y,U,V)
```

【例 6-5】　下面程序绘制函数的梯度场。

```
Z = peaks(40);
[DX,DY] = gradient(Z,.2,.2);
contour(X,Y,Z)
hold on
quiver(X,Y,DX,DY)
colormap hsv
grid off
hold off
```

绘制结果如图 6-7 所示。

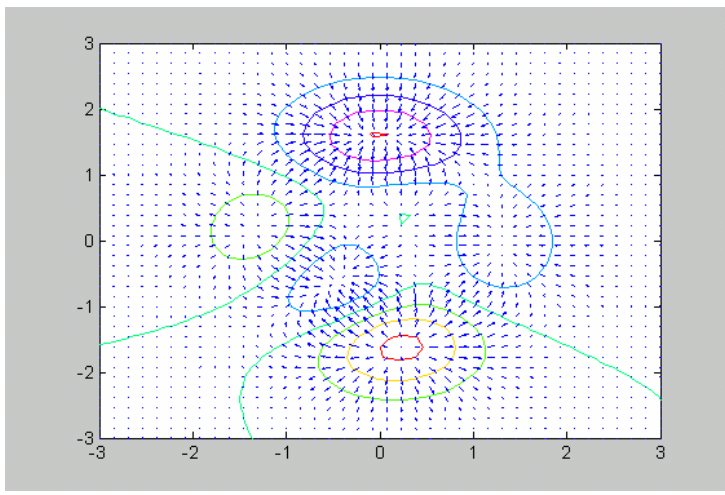


图 6-7　向量图

6.3.2　三维向量图

用 quiver3 函数绘制三维向量图，其调用格式如下。

- quiver3(Z,U,V,W)　　在 Z 矩阵指定的等距离表面点上绘制向量图。quiver3 函数自动根据它们之间的距离确定显示比例，以防止显示溢出。

- quiver3(X,Y,Z,U,V,W)　　指定点（*x*,*y*,*z*）处的元素（*u*,*v*,*w*），绘向量图。矩阵 X，Y，Z，U，V，W 必须大小相等，且包含对应位置和向量元素。
- quiver3(…,scale)　　自动根据绘图数据之间的距离确定显示比例，然后用比例值乘以原数据。
- quiver3(…,LineSpec)　　用任意有效的 LineSpec 参数指定线型和颜色。
- quiver3(…,LineSpec,'filled')　　填充 LineSpec 参数指定的标记。
- h = quiver3(…)　　返回一个直线句柄向量。

【例 6-6】　下面程序绘制函数的表面法向向量。

```
[X,Y] = meshgrid(-2:0.25:2,-1:0.2:1);
Z = X.* exp(-X.^2 - Y.^2);
[U,V,W] = surfnorm(X,Y,Z);
quiver3(X,Y,Z,U,V,W,0.5);
hold on
surf(X,Y,Z);
colormap hsv
view(-35,45)
axis ([-2 2 -1 1 -.6 .6])
hold off
```

绘制结果如图 6-8 所示。

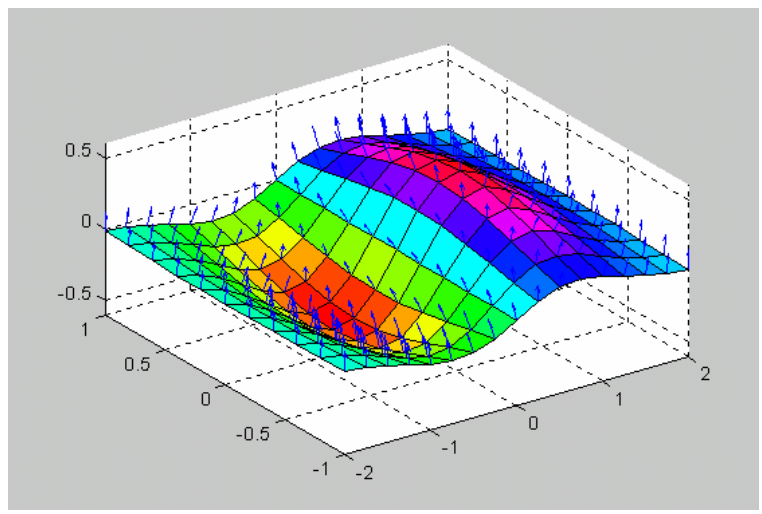


图 6-8　三维向量图

6.4　剖面图

剖面图显示穿过三维体积内的一个切面，这个切面可以是二维横剖面、纵剖面、任意方向的剖面，也可以是一个曲面，甚至是一个球形表面。剖面图在工程中有着广泛的应用。比如，在地质上，如果某个剖面在地质构造、工程选址布线等方面有着重要意义，往往要把这个剖面单独绘出来进行研究。多个连续且有一定间隔的二维剖面也能反映三维总体的信息。

6.4.1 slice 函数

用 slice 函数绘制剖面图，其调用格式如下。

- slice(V,sx,sy,sz)　　绘体积 V 中向量 sx, sy 和 sz 指定的点上沿 *x*, *y* 和 *z* 方向的剖面。V 是一个 *m*×*n*×*p* 的体积数组，包含默认位置 *x* = 1:*n*, *y* = 1:*m*, *z* =1:*p* 处的数据点。向量 sx, sy 和 sz 中的每个元素定义剖面在 *x*, *y* 和 *z* 方向上的分量。
- slice(X,Y,Z,V,sx,sy,sz)　　绘体积 V 的剖面图。X, Y 和 Z 为三维数组，指定 V 的坐标。X, Y 和 Z 必须位于各向同性的正交空间中。
- slice(V,XI,YI,ZI)　　为由 XI, YI 和 ZI 定义的剖面在体积 V 中绘数据剖面图。XI, YI 和 ZI 为矩阵，定义一个表面，体积在表面点上计算。XI, YI 和 ZI 必须大小相同。
- slice(X,Y,Z,V,XI,YI,ZI)　　沿数组 XI, YI 和 ZI 定义的表面绘穿过体积 V 的剖面。
- slice(…,'method')　　指定内插方法。'method'可以是'linear', 'cubic'或'nearest'。
  'linear'——指定为线性插值（默认选项）
  'cubic'——指定为三次插值
  'nearest'——指定为最小距离插值
- h = slice(…)　　返回表面图形对象的句柄向量。

【例 6-7】　在区间 $-2\le x\le 2$, $-2\le y\le 2$, $-2\le z\le 2$ 上绘下面函数的图形：

$$v = x\cdot e^{(-x^2-y^2-z^2)}$$

```
[x,y,z] = meshgrid(-2:.2:2,-2:.25:2,-2:.16:2);
v = x.*exp(-x.^2-y.^2-z.^2);
xslice = [-1.2,.8,2]; yslice = 2; zslice = [-2,0];
slice(x,y,z,v,xslice,yslice,zslice)
colormap hsv
```

绘制结果如图 6-9 所示。

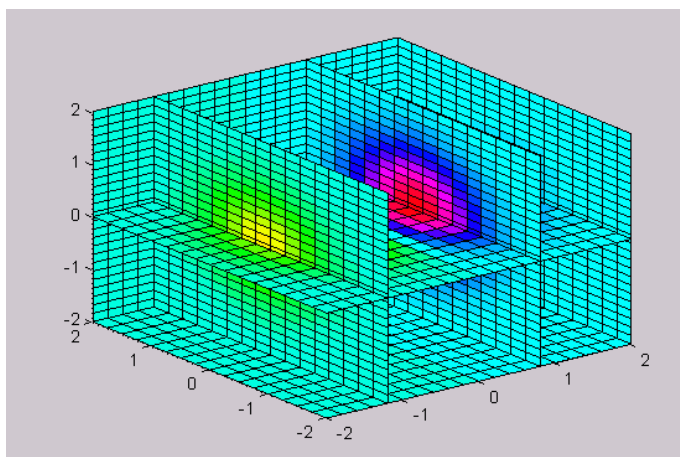


图 6-9　剖面图

也可以沿给定方位的面板进行切片，它需要进行以下步骤。

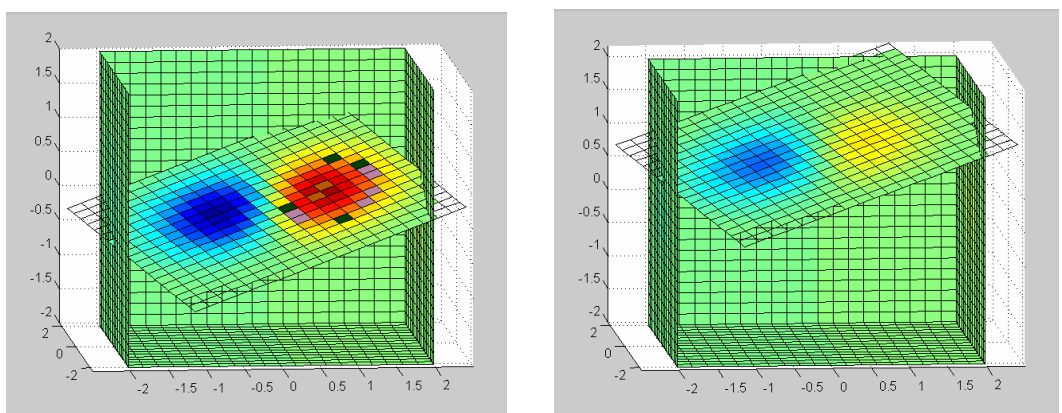
- 在体积域内创建一个切片表面（用 surf 和 linspace 函数）。
- 旋转，确定该表面相对于坐标轴的方位（用 rotate 函数）。

- **获得表面 3 个方向上的数据** Xdata，Ydata **和** Zdata（**用** get **函数**）。
- **使用上面的数据绘体积内的切片面板。**

下面的例子演示了一个面板是如何沿 *z* **轴方向穿过体积内部的。**

```
for i = -2:.5:2
    hsp = surf(linspace(-2,2,20),linspace(-2,2,20),zeros(20)+i);
    rotate(hsp,[1,-1,1],30)
    xd = get(hsp,'XData');
    yd = get(hsp,'YData');
    zd = get(hsp,'ZData');
    delete(hsp)
    slice(x,y,z,v,[ -2,2],2,  -2) % Draw some volume boundaries
    hold on
    slice(x,y,z,v,xd,yd,zd)
    hold off
    axis tight
    view(-5,10)
    drawnow
end
```

图 6-10(a)**和**(b)**演示了活动面板穿越体积过程中位于体积下方和上方的两种情况。**



(a)　　(b)

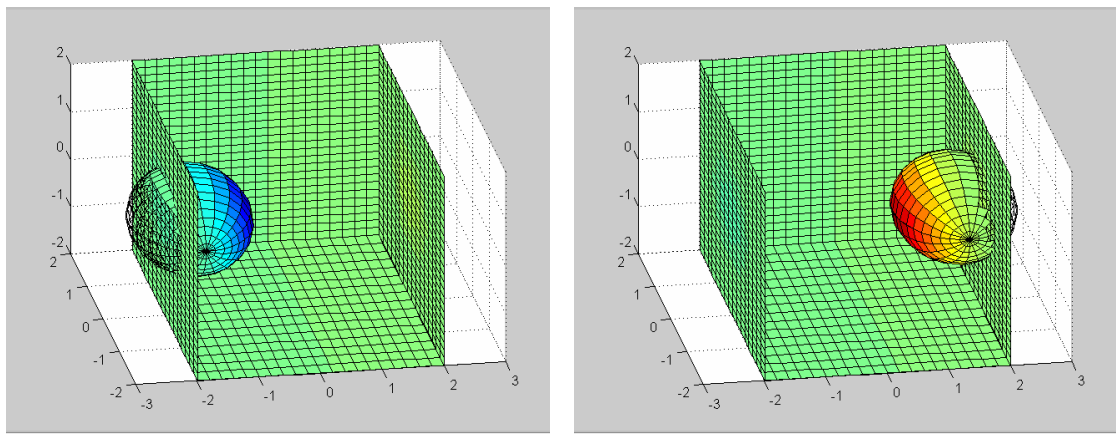
图 6-10　**切片沿** *z* **轴方向穿越体积**

也可以用任意形状的表面对体积进行切片。下面的一个例子演示了一个球形切片表面穿过体积的情形。

```
[xsp,ysp,zsp] = sphere;
slice(x,y,z,v,[-2,2],2,  -2)
for i = -3:.2:3
    hsp = surface(xsp+i,ysp,zsp);
    rotate(hsp,[1 0 0],90)
    xd = get(hsp,'XData');
    yd = get(hsp,'YData');
```

```
    zd = get(hsp,'ZData');
    delete(hsp)
    hold on
    hslicer = slice(x,y,z,v,xd,yd,zd);
    axis tight
    xlim([-3,3])
    view(-10,35)
    drawnow
    delete(hslicer)
    hold off
end
```

图 6-11(a)**和**(b)**分别为球形切片穿越体积时位于体积左侧和右侧时的情形。**



(a)　　(b)

图 6-11　**球形切片穿越体积**

6.4.2　切片等值线图

切片等值线图是绘三维体积图的剖面图并在其上叠加对应位置的等值线图。与切片流线图的功能相似，但它体现的是与流线方向垂直的等势线，用 contourslice **函数来实现。**

用 contourslice **函数在体积切片面板中绘等值线图，其调用格式如下。**

- contourslice(X,Y,Z,V,Sx,Sy,Sz)　　**在与** *x* **轴、** *y* **轴和** *z* **轴平行的面板中，在向量** Sx, Sy **和** Sz **所表示的点上绘等值线。数组** X，Y **和** Z **定义体积** V **的坐标。每条等值线的颜色由体积** V **确定，它必须是一个** $m\times n\times p$ **的体积数组。**
- contourslice(X,Y,Z,V,Xi,Yi,Zi)　　**沿数组** Xi,Yi,Zi **定义的表面绘穿过体积** V **的等值线。**
- contourslice(V,Sx,Sy,Sz)**和** contourslice(V,Xi,Yi,Zi) (**忽略** X, Y **和** Z **变量**)　　**假设**[X,Y,Z] = meshgrid(1:n,1:m,1:p)**，其中** [m,n,p]= size(v)**。**
- contourslice(⋯,n)　　**在每个面板中绘** n **条等值线，覆盖自动设置的值。**
- contourslice(⋯,cvals)　　**在每个面板中由向量** cvals **指定的位置上绘** length(cval)**条等值线。**
- contourslice(⋯,[cv cv])　　**在水平** cv **上计算每个面板中的单条等值线。**

- contourslice(…,'method') 指定将要使用的内插方法。内插方法可以是 linear, cubic 或 nearest。nearest 是默认方法。
- h = contourslice(…) 返回阴影对象的句柄向量。

【例 6-8】 本例使用 flow 数据集绘制切片等值线图。

- 对于 Sx，指定一个 length=9 的向量；对于 Sy，指定一个空向量，对于 Sz，指定一个值为 0 的标量。这将在 *y*-*z* 面板中沿 *x* 方向创建 9 条等值线。在 *x*-*y* 面板中 *z*=0 处创建一条等值线。
- 使用 linspace 函数定义一个数值从-8 到 2 的具有 10 个元素的线性间隔的向量，它指定每个间隔所绘的等值线条数。
- 定义视图和投影类型(camva，camproj 和 campos)。
- 设置图像(gcf)和坐标轴 axes(gca)的属性。

```
[x y z v] = flow;
h = contourslice(x,y,z,v,[1:9],[],[0],linspace(-8,2,10));
axis([0,10,-2,2,-2,2]); daspect([1,1,1])
camva(24); camproj perspective;
campos([-3,-15,5])
set(gcf,'Color',[.5,.5,.5],'Renderer','zbuffer')
set(gca,'Color','black','XColor','white', …
    'YColor','white','ZColor','white')
box on
```

绘制结果如图 6-12 所示。

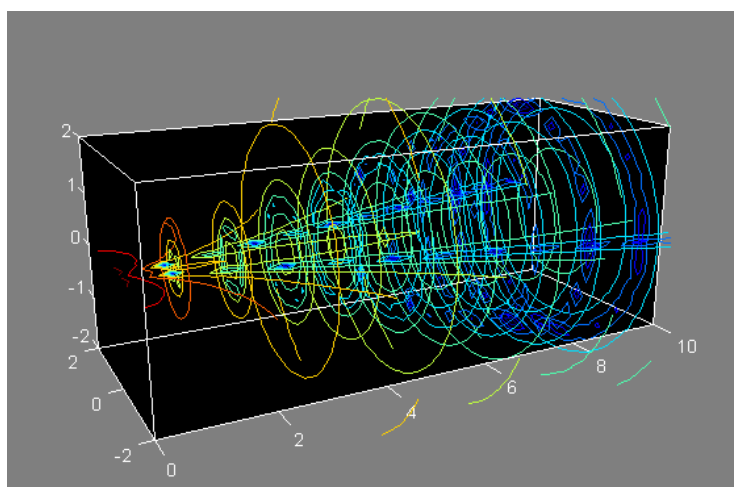


图 6-12 切片等值线图

6.4.3 切片流线图

切片流线图是在三维图的图切剖面上叠加流线图。利用该图可以显示三维体积内部某个特定剖面上的流动特征。用 streamslice 函数实现。

用 streamslice 函数在切片面板中绘制流线图，其调用格式如下。

- streamslice(X,Y,Z,U,V,W,startx,starty,startz) 用箭头绘向量数据 U，V，W 的间隔一

定的流线图。数组 X, Y, Z 定义 U，V，W 的坐标。U，V，W 必须是 $m \times n \times p$ 的体积数组。

需要注意的是，不能假定流动方向与切片面板平行。例如，在常数 z 处的流动切片中，当为面板计算流线时，忽略 z 方向上的向量场分量 W。

流动切片对于确定流线、流动锥体和流带的起点很有用。

- streamslice(U,V,W,startx,starty,startz)　　假设 X, Y 和 Z 由下面的语句确定：

  [X,Y,Z] = meshgrid(1:n,1:m,1:p)

其中，[m,n,p] = size(U)。

- streamslice(X,Y,U,V)　　根据向量体积数据 U 和 V 绘流线。数组 X 和 Y 定义 U 和 V 的坐标。
- streamslice(U,V)　　假设 X, Y 和 Z 由下面的语句定义：

  [X,Y,Z] = meshgrid(1:n,1:m,1:p)

其中，[m,n,p] = size(U)。

- streamslice(…,density)　　对流线的自动间隔进行修改。density 参数必须大于 0，其默认值为 1；该值越大，在每个面板上创建的流线越多。
- streamslice(…,'arrowsmode')　　确定方向箭头是否存在。arrowmode 可以是：

  arrows ——在流线上绘方向箭头（默认选项）

  noarrows ——不绘方向箭头
- streamslice(…,'method')　　指定内插方法，可以是：

  linear ——线性插值（默认设置）

  cubic ——三次内插

  nearest ——最小距离内插
- h = streamslice(…)　　返回所创建的直线对象的句柄向量。
- [vertices arrowvertices] = streamslice(…)　　返回流线和箭头的两个单元数组。可以将这些值传递给任何一条流线。

【例 6-9】　本例利用 wind 数据创建一个 z=5 处的流动切片图，效果如图 6-13 所示。

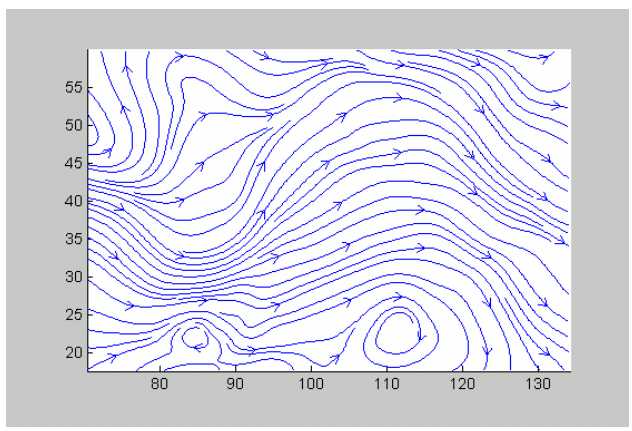


图 6-13　切片流线图之一

```
load wind
daspect([1 1 1])
```

```
streamslice(x,y,z,u,v,w,[],[],[5])
axis tight
```

本例使用 streamslice **函数计算流线和方向箭头的顶点数据。**streamline **函数将利用该数据绘直线和箭头，效果如图** 6-14 **所示。**

```
load wind
daspect([1 1 1])
[verts averts] = streamslice(u,v,w,10,10,10);
streamline([verts averts])
spd = sqrt(u.^2 + v.^2 + w.^2);
hold on;
slice(spd,10,10,10);
colormap(hot)
shading interp
view(30,50); axis(volumebounds(spd));
camlight; material([.5 1 0])
```

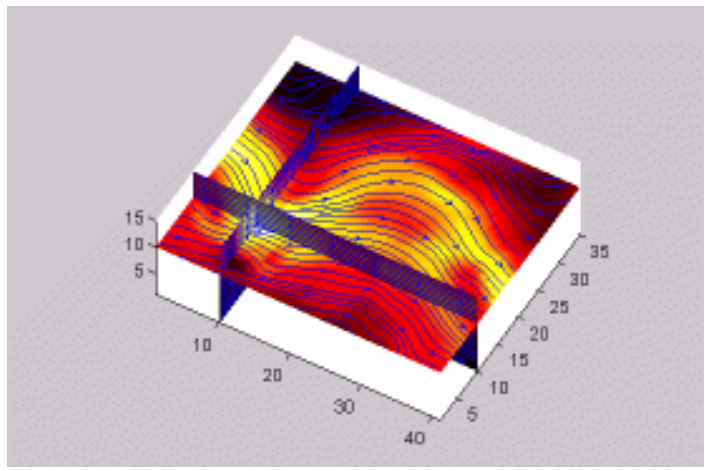


图 6-14　**切片流线图之二**

本例在表面图上添加等值线，然后使用 streamslice **函数绘直线，表示表面的梯度。使用** interp2 **函数查找表面上直线的点，效果如图** 6-15 **所示。**

```
z = peaks;
surf(z)
shading interp
hold on
[c ch] = contour3(z,20); set(ch,'edgecolor','b')
[u v] = gradient(z);
h = streamslice(-u,-v);
set(h,'color','k')
for i=1:length(h);
    zi = interp2(z,get(h(i),'xdata'),get(h(i),'ydata'));
        set(h(i),'zdata',zi);
```

```
end
view(30,50); axis tight
```

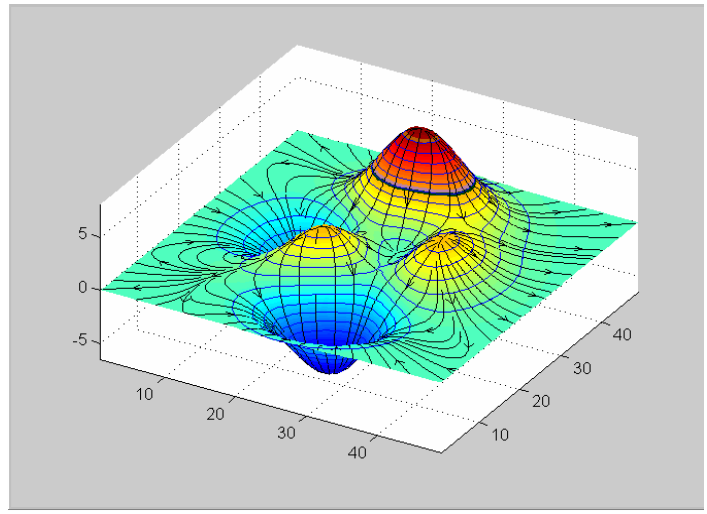


图 6-15　切片流线图之三

6.5　流线图

流线图常用于表现流动物体的流动特征。图 6-16 是一个典型的地下水流线图，图中灰色半封闭多边形表示一个三面隔水的基坑，基坑中分布有数排井点，井点连续抽水，使水位下降，然后才能比较方便地进行基坑开挖。井点处水位下降以后，周围的地下水向井点处流动，流线图则清晰地把地下水的流动路径表现出来。

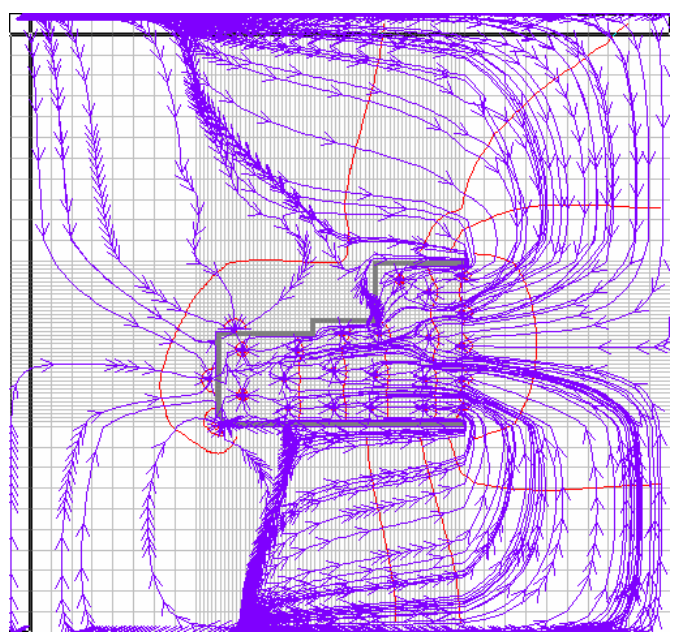


图 6-16　一个典型的流线图（叠加了等值线）

MATLAB 对流线图有更生动的表现。在 MATLAB 中，不仅可以绘制常规的流线图，还可以绘制流锥图、流沙图、流管图、流带图和卷曲图等图形，用多种形式表现物体的流动特征。流带图、流锥图、流管图等更是采用了先进的颜色填充算法和光照技术，可以用具有高度真实感的实体图形来进行表现。

6.5.1 常规的流线图

用 streamline 函数根据二维和三维向量数据绘制常规的流线图，其调用格式如下。

- h = streamline(X,Y,Z,U,V,W,startx,starty,startz)　　绘三维向量数据 U，V，W 的流线图。数组 X，Y，Z 定义 U，V，W 的坐标。startx,starty,startz 定义流线的起点位置。输出变量 h 包含直线句柄向量，每条流线对应一个句柄。
- h = streamline(U,V,W,startx,starty,startz)　　假设数组 X, Y 和 Z 由[X,Y,Z] = meshgrid(1:N,1:M,1:P)定义，其中[M,N,P] = size(U)。
- h = streamline(XYZ)　　假设 XYZ 为预先计算好的顶点数组的单元数组。
- h = streamline(X,Y,U,V,startx,starty)　　绘二维向量数据 U，V 的流线图。数组 X，Y 定义 U，V 的坐标。startx 和 starty 定义流线的起点位置。输出变量 h 包含一个直线句柄向量，一条流线对应一个句柄。
- h = streamline(U,V,startx,starty)　　假设数组 X 和 Y 由[X,Y] = meshgrid(1:N,1:M)定义，其中，[M,N] = size(U)。
- h = streamline(XY)　　假设 XY 为预先计算好的顶点数组的单元数组。
- streamline(…,options)　　指定创建流线时所用的选项。options 可以定义为一个只有一个元素（步长）的向量或有两个元素（步长和流线顶点的最大个数）的向量：

  [stepsize]

  或

  [stepsize, max_number_vertices]

  如果没有指定值，MATLAB 使用默认值：

  stepsize = 0.1 (单元的十分之一)

  最大顶点个数 = 1000

【例 6-10】　本例利用 MATLAB 内部所带数据集 wind 绘制流线图。装载 wind 数据集，在 MATLAB 主空间中创建变量 x, y, z, u, v 和 w。

流线面板指示，空气的流动方向是从西到东，起点位置在 $x = 80$ 处(它靠近 x 坐标的最小值)。meshgrid 函数生成流线的起点位置。

```
load wind
[sx,sy,sz] = meshgrid(80,20:10:50,0:5:15);
h = streamline(x,y,z,u,v,w,sx,sy,sz);
set(h,'Color','red')
view(3)
```

绘制结果如图 6-17 所示。

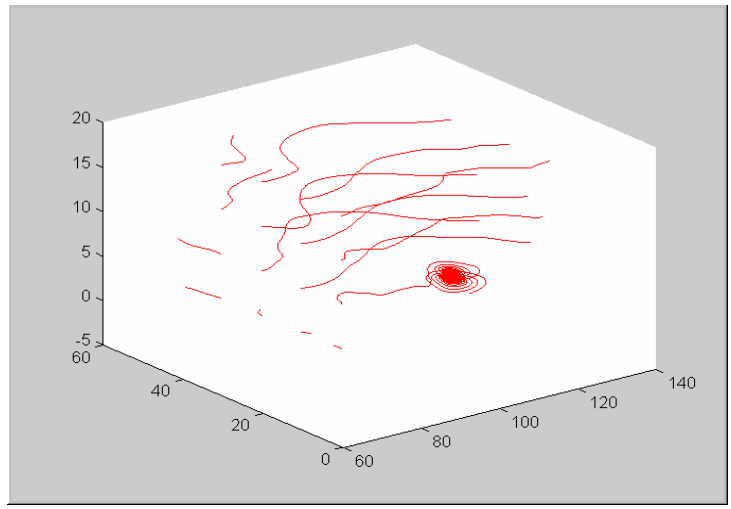


图 6-17　流线图

6.5.2　流锥图

用 coneplot 函数在三维向量场中用圆锥体绘制速率向量图，其调用格式如下。

- coneplot(X,Y,Z,U,V,W,Cx,Cy,Cz)　　用圆锥体绘速率向量图。圆锥体指定速率向量的方向，其长度与速率向量的大小成比例。X, Y, Z 定义向量场的坐标。U, V, W 定义向量场。这些数组必须具有相同的大小、单调性和三维特征。Cx, Cy, Cz 定义向量场中圆锥体的位置。
- coneplot(U,V,W,Cx,Cy,Cz) (忽略 X, Y 和 Z 变量)　　假设[X,Y,Z] = meshgrid(1:n, 1:m,1:p)，其中，[m,n,p]= size(U)。
- coneplot(…,s)　　MATLAB 自动确定圆锥体的显示比例 s，使它们能拟合图形的大小。如果没有指定 s 的大小，MATLAB 将它设为 1。令 s=0，则不自动设置圆锥体的比例。
- coneplot(…,color)　　在向量场中进行 color 数组插值，并根据插值的值来确定圆锥体的颜色。color 的大小必须与 U, V, W 数组的大小相同。该选项只与圆锥体一起使用。
- coneplot(…,'quiver')　　用箭头代替圆锥体绘图。
- coneplot(…,'method')　　指定插值方法。可选方法包括：线性插值、二次插值、三次插值和最小距离插值等。默认时选用线性插值。
- coneplot(X,Y,Z,U,V,W,'nointerp')　　不将圆锥体的位置插值到三维区域中。圆锥体在由 X，Y 和 Z 确定的位置绘制，其方位由 U, V, W 确定。数组 X, Y, Z, U, V 和 W 必须大小相同。
- h = coneplot(…)　　返回绘制圆锥体的阴影对象的句柄。可以使用 set 命令改变圆锥体的属性。

注意：coneplot 函数自动设置图中圆锥体的显示比例，并使它们与速率向量的大小成比例。调用 coneplot 函数之前，可以用 daspect 命令设置数据方向比。例如：

```
daspect([1,1,1])
```

【例 6-11】　本例演示向量空间数据的速率向量圆锥体图，该数据代表方形空间区域中气流的流动。最后生成的图形采用了一些方法使得数据显示的效果更好，包括：

- 圆锥体图形表示气流速率的大小和方向。

- 数据边界处的分片面板显示了三维区域内部圆锥体的可视化承接。
- 一定方向的光照提供了圆锥体方位的视图序列。
- 通过选择视点、投影类型和放大来最好地重现数据的内容信息。

1．装载数据

本例用到 winds 数据集，其中包括 6 个三维数组：u, v 和 w 指定每个点上向量元素，x, y 和 z 指定坐标位置。为了指定圆锥体的放置位置，可以首先确定数据的范围。

```
load wind
xmin = min(x(:));
xmax = max(x(:));
ymin = min(y(:));
ymax = max(y(:));
zmin = min(z(:));
```

2．创建圆锥体图

确定在数据空间的什么位置绘圆锥图。下面的例子分 8 步选择 x 和 y 的完整范围，在 z 中分 4 步选择 3～15 的范围。

调用 coneplot 函数之前，使用 daspect 设置坐标轴的数据方向比率，使得 MATLAB 可以确定圆锥体的合适大小。

绘圆锥体，设置比例因子为 5，使得圆锥体比默认大小更大。

设置每个圆锥体的颜色(表面色,边缘色)。

```
daspect([2,2,1])
xrange = linspace(xmin,xmax,8);
yrange = linspace(ymin,ymax,8);
zrange = 3:4:15;
[cx cy cz] = meshgrid(xrange,yrange,zrange);
hcones = coneplot(x,y,z,u,v,w,cx,cy,cz,5);
set(hcones,'FaceColor','red','EdgeColor','none')
```

绘制结果如图 6-18 所示。

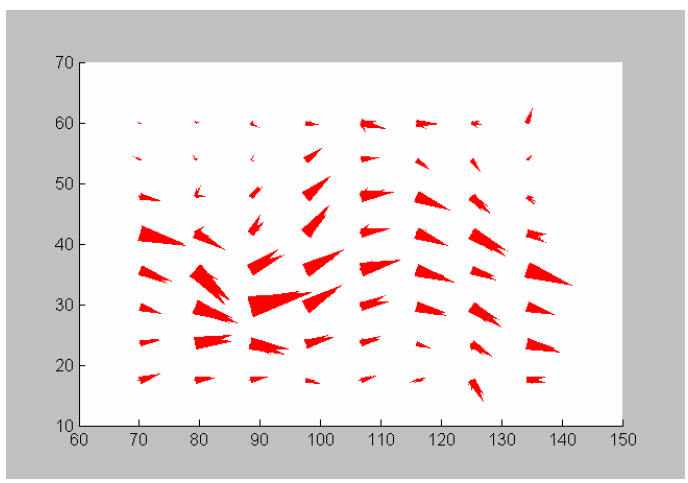


图 6-18　圆锥体图

3．添加分片面板

计算向量场的大小（它代表风的速度），为 slice **命令生成标量数据。沿** *x* **轴在** xmin **和** xmax **处，沿** *y* **轴在** ymin **和** ymax **处创建分片面板。指定内插面颜色，分片的着色指示风速，不绘边缘，效果如图** 6-19 **所示。**

```
hold on
wind_speed = sqrt(u.^2 + v.^2 + w.^2);
hsurfaces = slice(x,y,z,wind_speed,[xmin,xmax],ymax,zmin);
set(hsurfaces,'FaceColor','interp','EdgeColor','none')
hold off
```

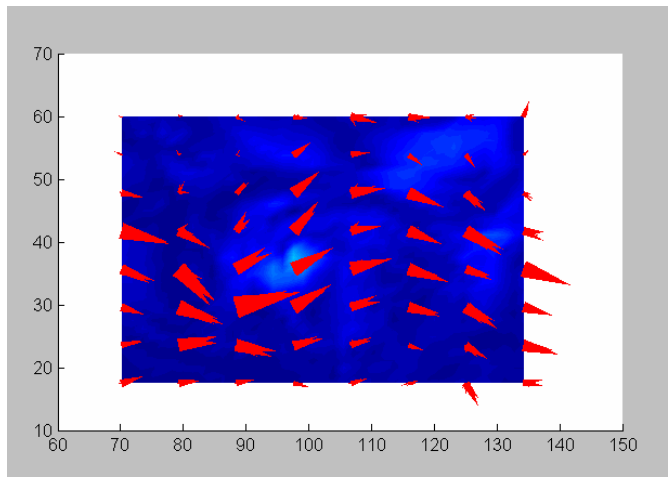


图 6-19　**添加分片面板**

4．定义视图

使用 axis **命令设置坐标轴的范围与数据的范围相等。设置视图方向为** azimuth = 30，elevation = 40。**使图尽量大，效果如图** 6-20 **所示。**

```
axis tight; view(30,40); axis off
camproj perspective; camzoom(1.5)
```

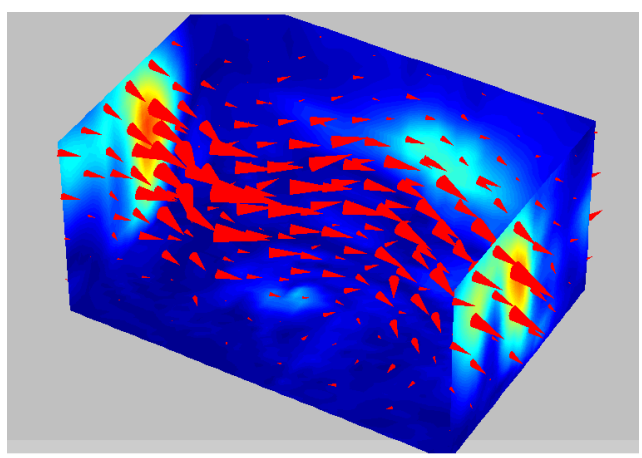


图 6-20　**定义视图**

5．添加光线

光源会影响分片面板（表面）和圆锥体（阴影）。但可以单独设置它们的光照特性。

在相机右侧给光，给圆锥体和分片面板一个平滑、三维的表面。对于每个分片面板，增加 AmbientStrength 属性的值，改进深蓝色的可视性，效果如图 6-21 所示。

增加 DiffuseStrength 属性的值，加亮显示这些圆锥体，并显示特殊的反射光影。

```
camlight right; lighting phong
set(hsurfaces,'AmbientStrength',.6)
set(hcones,'DiffuseStrength',.8)
```

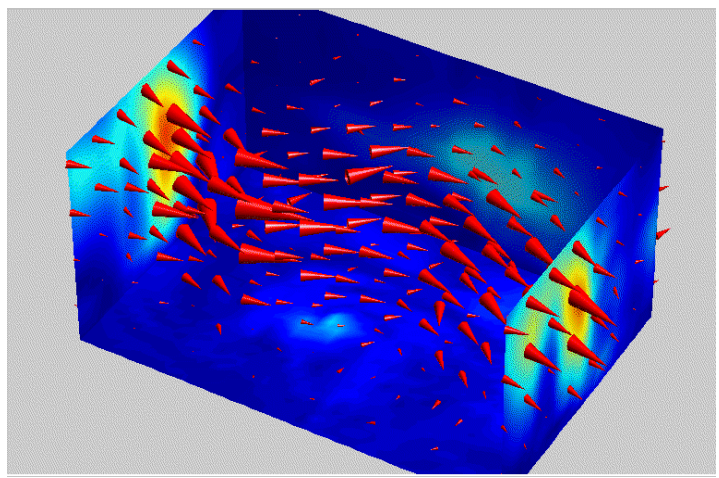


图 6-21　添加光线

6.5.3　流沙图

MATLAB 可以用流动的点来表示物体的流动，称为流沙图。它用 streamparticles 函数来实现。

用 streamparticles 函数生成和显示流沙图，其调用格式如下。

- streamparticles(vertices)　　绘向量场的流沙图。流动的颗粒通常用标记代表，可以显示流线的位置和速率。vertices 为一个二维或三维向量的单元数组。
- streamparticles(vertices,n)　　用 n 确定需要绘多少个流动微粒。ParticleAlignment 属性控制如何解释 n。
  - ➢ 如果 ParticleAlignment 设置为 off（默认设置），并且 n 大于 1，则在流线上等间距绘近似于 n 个微粒。
  - ➢ 如果 n 小于或等于 1，n 解释为原始流动顶点的一部分。比如，当 n=0.2，则近似有 20% 的顶点被使用。n 确定所绘顶点个数的上限。注意，微粒的实际个数可能会偏离 n。
  - ➢ 如果 ParticleAlignment 设置为 on，则 n 确定微粒最多的流线上的微粒个数。并将其他流线上微粒的间隔数设置为此值。默认时，n = 1。
- streamparticles(…,'PropertyName',PropertyValue,…)　　用给定的属性和指定值控制流动

  微粒。任何没有指定的属性按默认值赋值。MATLAB 忽略表 6-1 所示的属性名。

表 6-1　流动微粒属性表

流动微粒的属性	描　　述
Animate – 流动微粒的运动[非负整数]	流动微粒的快照次数。默认值为 0，不进行快照。设置为 Inf 时，将一直进行快照，直到按下 ctrl-c 位置
FrameRate – 每秒快照景框[非负整数]	本属性为快照指定每秒的景框数。设置为 Inf 时，将尽可能快地绘快照图（默认设置）。注意，快照的速度受到计算机速度的限制。此时，FrameRate 属性的值可能不必达到
ParticleAlignment – 将微粒与流线对齐[onl{off}]	设置此属性为 on，在每个流线的起点绘微粒。本属性控制 streamparticles 函数如何解释变量 n（流动微粒数）

流动微粒为直线对象，除了流动微粒属性以外，可以指定任何直线对象属性，如 Marker 和 EraseMode 等。streamparticles 函数被调用时设置表 6-2 所示的直线属性。

表 6-2　直线属性表

直线属性	Streamparticles 函数设定的值
EraseMode	xor
LineStyle	none
Marker	o
MarkerEdgeColor	none
MarkerFaceColor	red

可以通过指定一个属性名和属性值作为变量来覆盖这些属性值中的任意一个。例如，下面的语句用 RGB 值将 MarkerFaceColor 设置为中灰：

```
streamparticles(vertices,'MarkerFaceColor',[.5 .5 .5])
```

- streamparticles(line_handle,…)　　使用由 line_handle 确认的线性对象绘流沙图。
- h = streamparticles(…)　　返回创建的直线对象的句柄向量。

【例 6-12】　本例将流线和流沙快照组合起来，interpstreamspeed 函数控制快照的速度。将坐标轴的 DrawMode 属性设置为 fast，将使快照速度更快，效果如图 6-22 所示。

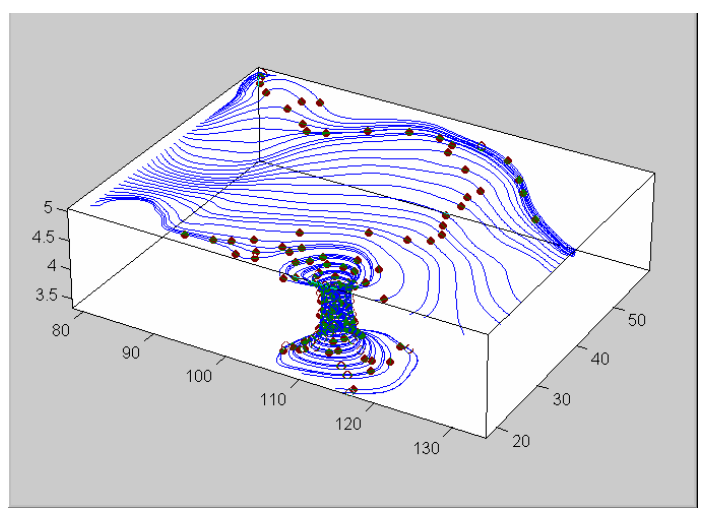


图 6-22　流沙图

```
load wind
[sx sy sz] = meshgrid(80,20:1:55,5);
verts = stream3(x,y,z,u,v,w,sx,sy,sz);
sl = streamline(verts);
iverts = interpstreamspeed(x,y,z,u,v,w,verts,.025);
axis tight; view(30,30); daspect([1 1 .125])
camproj perspective; camva(8)
set(gca,'DrawMode','fast')
box on
streamparticles(iverts,35,'animate',10,'ParticleAlignment','on')
```

下面的图片为快照的一张静态视图。本例使用 z=5 处面板的流线对沿这些直线的流动进行快照，如图 6-23 所示。

```
load wind
daspect([1 1 1]); view(2)
[verts averts] = streamslice(x,y,z,u,v,w,[],[],[5]);
sl = streamline([verts averts]);
axis tight off;
set(sl,'Visible','off')
iverts = interpstreamspeed(x,y,z,u,v,w,verts,.05);
set(gca,'DrawMode','fast','Position',[0 0 1 1],'ZLim',[4.9 5.1])
set(gcf,'Color','black')
streamparticles(iverts, 200,…
    'Animate',100,'FrameRate',40,…
'MarkerSize',10,'MarkerFaceColor','yellow')
```

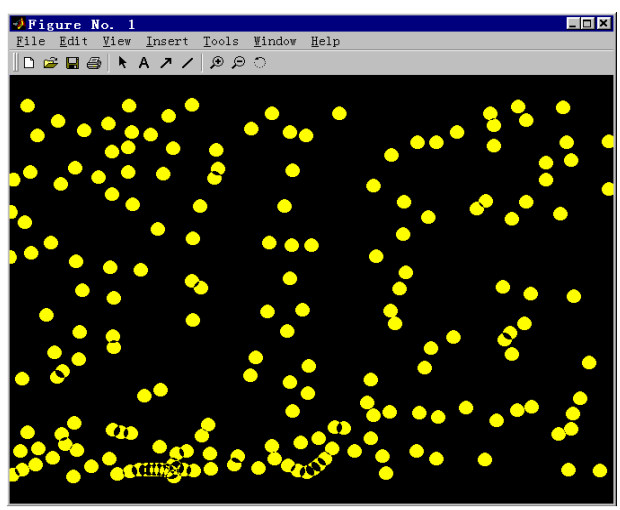


图 6-23　流沙图之二

6.5.4　流带图

流带图用飘动的条带来表现物体的流动。

用 streamribbon 函数创建一个三维流带图。调用格式如下。

- streamribbon(X,Y,Z,U,V,W,startx,starty,startz)　　根据向量体积数据 U，V 和 W 绘流带图。数组 X，Y 和 Z 定义 U，V 和 W 的坐标。startx，starty 和 startz 定义条带中心处流带的起点位置。条带的扭曲与向量场的卷曲成比例。条带的宽度自动进行计算。一般地讲，应该在调用 streamribbon 函数之前设置 DataAspectRatio 的属性。
- streamribbon(U,V,W,startx,starty,startz)　　假设 X，Y 和 Z 由下面的表达式定义：

  [X,Y,Z] = meshgrid(1:n,1:m,1:p)

其中，[m,n,p] = size(U)。

- streamribbon(vertices,X,Y,Z,cav,speed)　　假定预先计算的流线顶点、卷曲角速率和流速。vertices 为流动直线顶点的单元数组（就像 stream3 所创建的）。X，Y，Z，cav 和 speed 为三维数组。
- streamribbon(vertices,cav,speed)　　假定 X，Y 和 Z 由下面的语句确定：

  [X,Y,Z] = meshgrid(1:n,1:m,1:p)

其中，[m,n,p] = size(cav)。

- streamribbon(vertices,twistangle)　　使用向量单元数组 twistangle 确定条带的扭曲，vertices 每个元素的大小必须和 twistangle 的相同。
- streamribbon(…,width)　　设置条带的宽度为 width。
- h = streamribbon(…)　　返回句柄向量到表面对象。

【例 6-13】　本例使用 wind 数据集来绘制流带图。输入变量包括坐标、向量场组分和流带的起点位置，绘制效果如图 6-24 所示。

```
load wind
[sx sy sz] = meshgrid(80,20:10:50,0:5:15);
daspect([1 1 1])
streamribbon(x,y,z,u,v,w,sx,sy,sz);
%-----定义视图和光照。
axis tight
shading interp;
view(3);
camlight; lighting gouraud
```

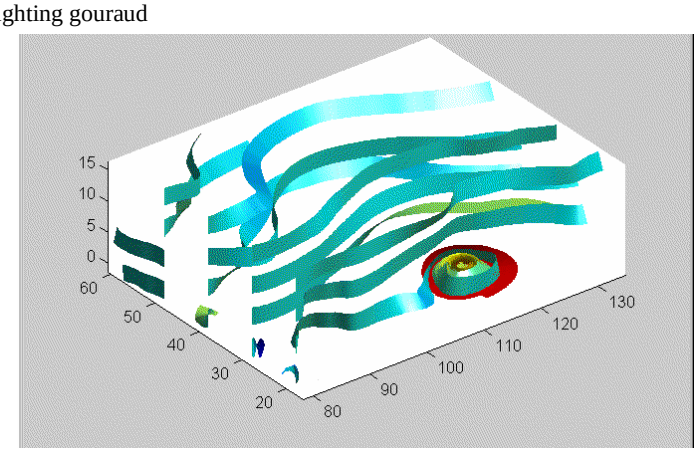


图 6-24　流带图之一

本例使用预先计算的顶点数据（stream3）、卷曲平均速率(curl)和流动速度，绘制结果如图 6-25 所示。

```
load wind
[sx sy sz] = meshgrid(80,20:10:50,0:5:15);
daspect([1 1 1])
verts = stream3(x,y,z,u,v,w,sx,sy,sz);
cav = curl(x,y,z,u,v,w);
spd = sqrt(u.^2 + v.^2 + w.^2).*.1;
streamribbon(verts,x,y,z,cav,spd);
axis tight
shading interp
view(3)
camlight; lighting gouraud
```

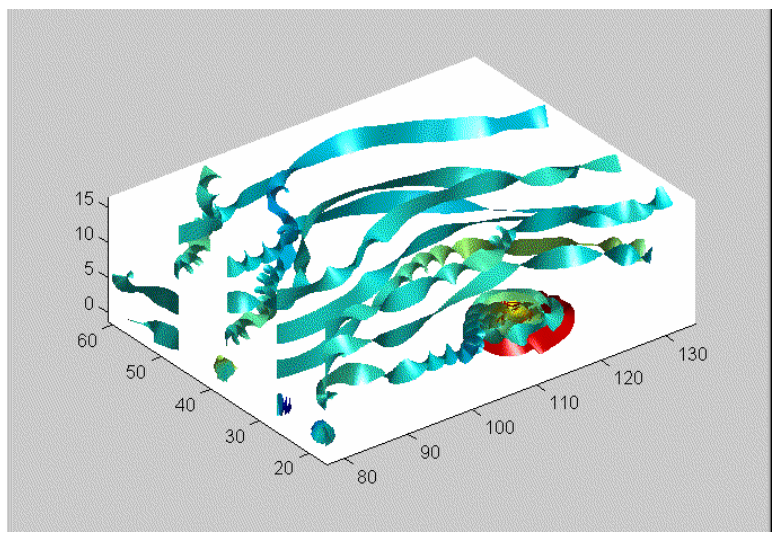


图 6-25　流带图之二

本例指定流带的扭曲角度，效果如图 6-26 所示。

```
t = 0:.15:15;
verts = {[cos(t)' sin(t)' (t/3)']};
twistangle = {cos(t)'};
daspect([1 1 1])
streamribbon(verts,twistangle);
axis tight
shading interp;
view(3);
camlight; lighting gouraud
```

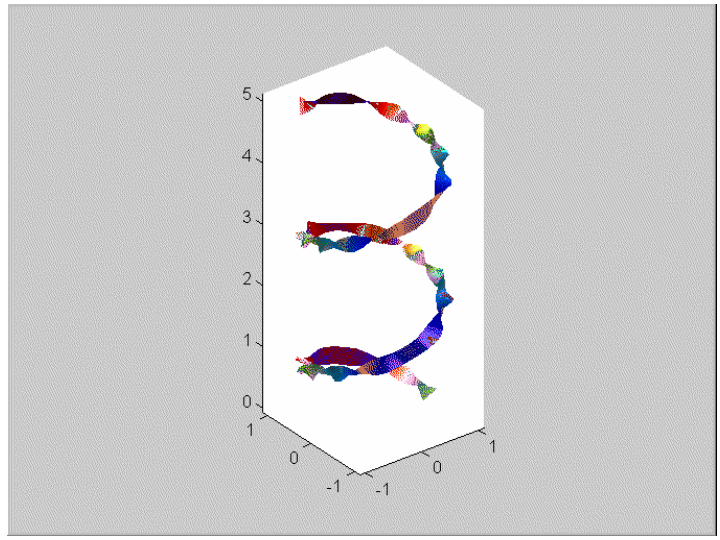


图 6-26　流带图之三

本例将圆锥图和流带图组合到一张图中，效果如图 6-27 所示。

```
%-----定义三维数组 x, y, z, u, v, w
xmin = -7; xmax = 7;
ymin = -7; ymax = 7;
zmin = -7; zmax = 7;
x = linspace(xmin,xmax,30);
y = linspace(ymin,ymax,20);
z = linspace(zmin,zmax,20);
[x y z] = meshgrid(x,y,z);
u = y; v = -x; w = 0*x+1;
daspect([1 1 1]);
[cx cy cz] = meshgrid(linspace(xmin,xmax,30),…
      linspace(ymin,ymax,30),[-3 4]);
h = coneplot(x,y,z,u,v,w,cx,cy,cz,'quiver');
set(h,'color','k');
%-----绘两个流带集合
[sx sy sz] = meshgrid([-1 0 1],[-1 0 1],-6);
streamribbon(x,y,z,u,v,w,sx,sy,sz);
[sx sy sz] = meshgrid([1:6],[0],  -6);
streamribbon(x,y,z,u,v,w,sx,sy,sz);
%-----定义视图和光照
shading interp
view(-30,10) ; axis off tight
camproj perspective; camva(66); camlookat;
camdolly(0,0,.5,'fixtarget')
camlight
```

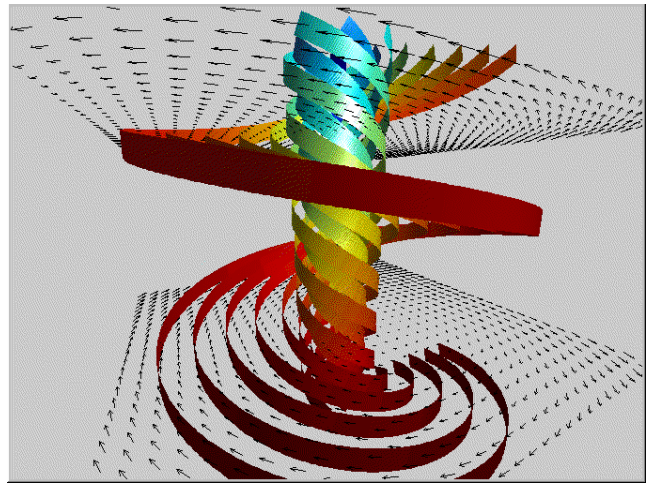


图 6-27　流带图之四

6.5.5　流管图

流管图是用不同粗细和颜色的空间管道表现物体在空间的流动特征。用 streamtube 函数绘制流管图。

用 streamtube 函数创建一个三维流管图，其调用格式如下。

- streamtube(X,Y,Z,U,V,W,startx,starty,startz)　　根据向量体积数据 U，V，W 绘流管图。数组 X，Y，Z 定义 U，V，W 的坐标。startx, starty 和 startz 定义流管中心处流线的起点。
- streamtube(U,V,W,startx,starty,startz)　　假设 X, Y 和 Z 由下面的语句确定：

  [X,Y,Z] = meshgrid(1:n,1:m,1:p)

其中，[m,n,p] = size(U)。

- streamtube(vertices,X,Y,Z,divergence)　　假定预先计算的流线顶点和差异。vertices 为流线顶点的单元数组。X，Y，Z 和 divergence 为三维数组。
- streamtube(vertices,divergence)　　假定 X，Y 和 Z 由下面的语句确定：

  [X,Y,Z] = meshgrid(1:n,1:m,1:p)

其中，[m,n,p] = size(divergence)。

- streamtube(vertices,width)　　在向量单元数组 width 中指定圆管的宽度。vertices 中每一个对应元素的大小必须与 width 相等。width 也可以是一个标量，为所有流管的宽度指定大小。
- streamtube(vertices)　　自动选择宽度。
- streamtube(…,[scale n])　　按 scale 参数确定管体的宽度。默认时 scale = 1。n 为沿圆管周边分布的点数。默认时 *n* = 20。
- h = streamtube(…z)　　返回一个句柄向量到表面对象。

【例 6-14】　本例利用 wind 数据集生成流管图。输入变量包括坐标、向量场分量和流管的起点位置，绘制效果如图 6-28 所示。

```
load wind
[sx sy sz] = meshgrid(80,20:10:50,0:5:15);
```

```
daspect([1 1 1])
streamtube(x,y,z,u,v,w,sx,sy,sz);
%-----定义视图和光照。
view(3)
axis tight
shading interp;
camlight; lighting gouraud
```

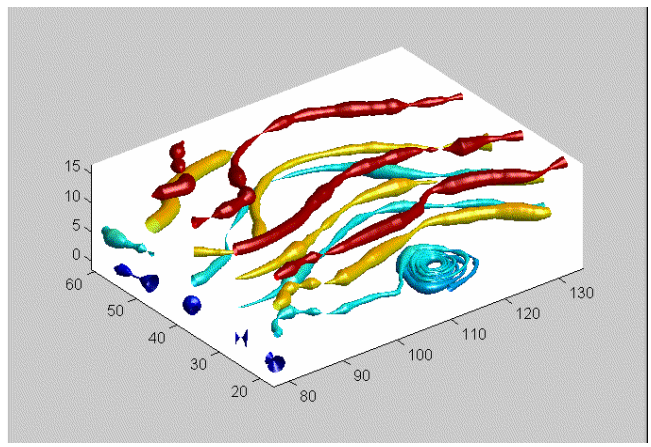


图 6-28　**流管图之一**

本例使用预先计算的顶点数据和差异值绘图，绘制效果如图 6-29 **所示。**

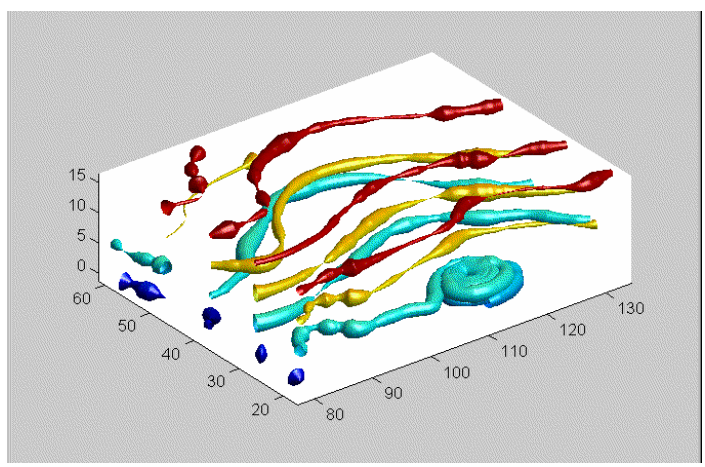


图 6-29　**流管图之二**

```
load wind
[sx sy sz] = meshgrid(80,20:10:50,0:5:15);
daspect([1 1 1])
verts = stream3(x,y,z,u,v,w,sx,sy,sz);
div = divergence(x,y,z,u,v,w);
streamtube(verts,x,y,z,-div);
%-----定义视图和光照。
```

```
view(3)
axis tight
shading interp
camlight; lighting gouraud
```

6.5.6 卷曲图

卷曲图通过计算向量场的卷曲和角速率来体现物体的流动特征。用 curl 函数绘图。

用 curl 函数计算向量场的卷曲和角速率的调用格式如下。

- [curlx,curly,curlz,cav] = curl(X,Y,Z,U,V,W)　　计算垂直于三维向量 U，V，W 所代表的流场的卷曲和角速率。数组 X，Y，Z 定义 U，V，W 的坐标。
- [curlx,curly,curlz,cav] = curl(U,V,W)　　假设 X, Y 和 Z 由下面的语句确定：

  [X Y Z] = meshgrid(1:n,1:m,1:p)

其中，[m,n,p] = size(U)。

- [curlz,cav]= curl(X,Y,U,V)　　计算卷曲的 z 分量和垂直于二维向量场 U，V 的 z 方向的角速率。数组 X，Y 定义 U，V 的坐标。
- [curlz,cav]= curl(U,V)　　假设 X 和 Y 由下面的语句确定：

  [X Y] = meshgrid(1:n,1:m)

其中，[m,n] = size(U)。

- [curlx,curly,curlz] = curl(…), curlx,curly] = curl(…)　　只返回卷曲参数。
- cav = curl(…)　　只返回卷曲角速率。

【例 6-15】　本例使用彩色切片面板在向量场中指定的位置上显示卷曲角速率，结果如图 6-30 所示。

```
load wind
cav = curl(x,y,z,u,v,w);
slice(x,y,z,cav,[90 134],[59],[0]);
shading interp
daspect([1 1 1]); axis tight
colormap hot(16)
camlight
```

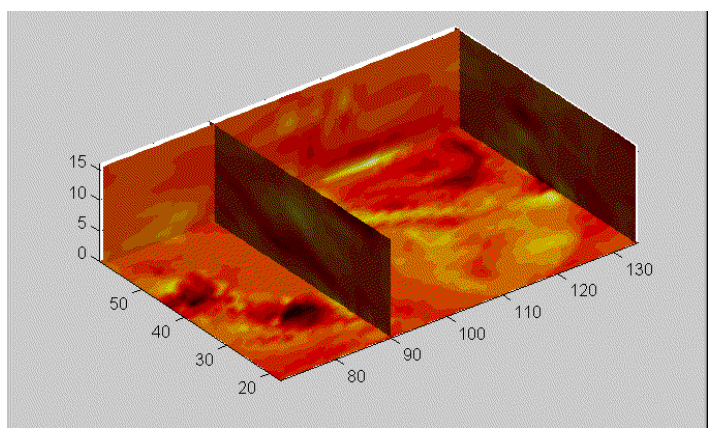


图 6-30　卷曲图之一

下面的例子是在体积的一个面板中查看卷曲角速率，并在相同的面板中绘制速率向量图。绘制结果如图 6-31 所示。

```
load wind
k = 4;
x = x(:,:,k); y = y(:,:,k); u = u(:,:,k); v = v(:,:,k);
cav = curl(x,y,u,v);
pcolor(x,y,cav); shading interp
hold on;
quiver(x,y,u,v,'y')
hold off
colormap copper
```

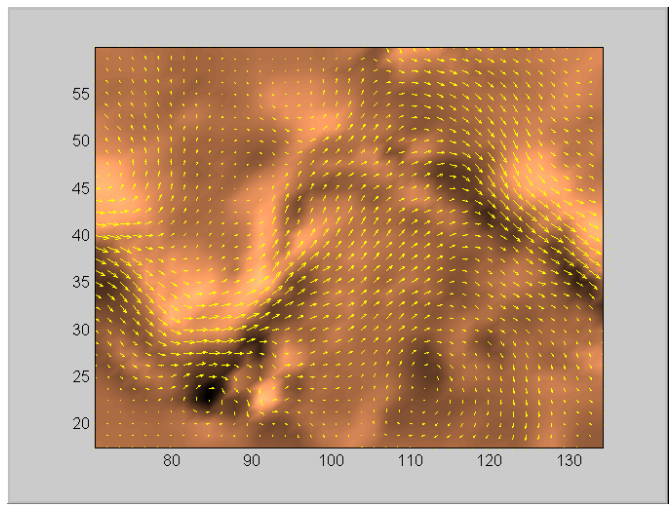


图 6-31　卷曲图之二

6.6　三维网格图

网格图常用于表现二维平面或三维实体。作为前处理，网格图常常用于建立二维、三维有限元计算对象的几何模型。其中的网格可以是三角形网格，也可以是四边形网格，还可以是其他多边形网格。MATLAB 中提供了四边形和三角形两种三维网格图形的生成函数，可以用它们绘图。

6.6.1　四边形网格图

用 mesh，meshc 和 meshz 函数绘四边形网格图，其调用格式如下。

- mesh(X,Y,Z)　　绘网格，用 Z 确定颜色。颜色与表面高度成比例。X 和 Y 为向量，length(X) = n，length(Y) = m,其中[m,n] = size(Z)。本例中，X(j),Y(i),Z(i,j)为网格线的横断面；X 和 Y 对应于 Z 的列和行。若 X 和 Y 为矩阵，则 X(i,j),Y(i,j),Z(i,j)为网格线的横断面。
- mesh(Z)　　用 X = 1:n 和 Y = 1:m 绘一个网格，其中，[m,n] = size(Z)。高度 Z 为矩形网格上的单值函数。颜色与表面高度成比例。

- mesh(…,C)　　用矩阵 C 确定的颜色画网格。若 X，Y 和 Z 为矩阵，则它们必须与 C 具有相同的大小。
- meshc(…)　　在网格下方画一个等值线图。
- meshz(…)　　在网格周围画一个 curtain 图(例如，画参考平面)。
- h = mesh(…)，h = meshc(…)和 h = meshz(…)　　返回表面图对象的句柄。

【例 6-16】　下面程序生成 peaks 函数的网格图与等值线图的组合图，如图 6-32 所示。

```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshc(X,Y,Z);
axis([-3 3 -3 3 -10 5])
```

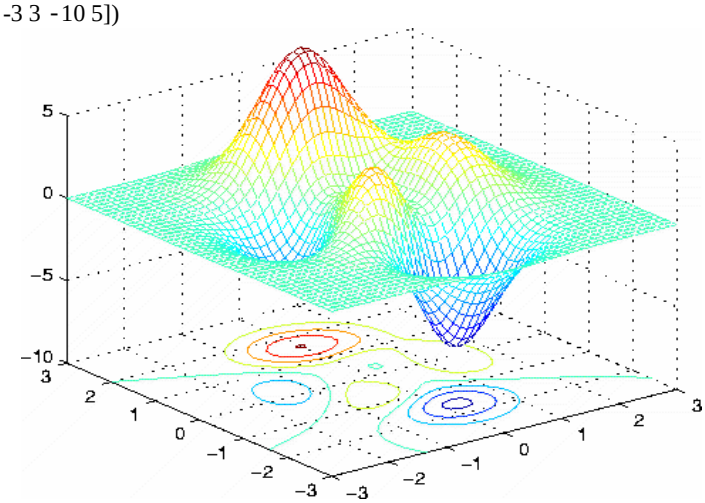


图 6-32　网格图和等值线图的组合图

下面程序创建 peaks 函数的窗帘图，如图 6-33 所示。

```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshz(X,Y,Z)
```

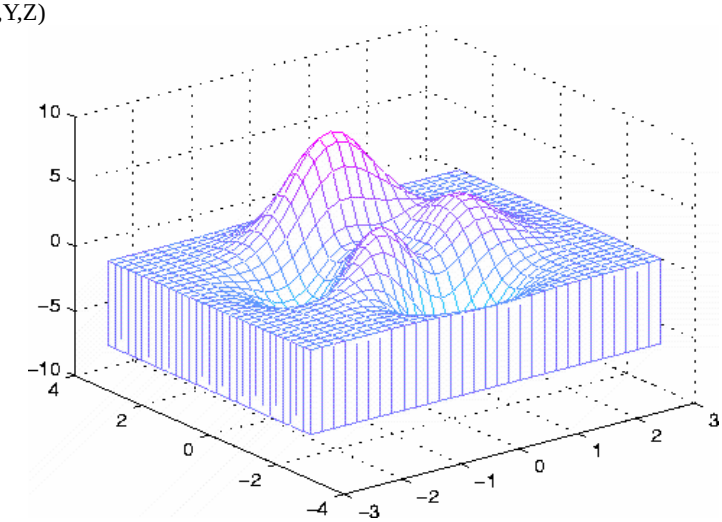


图 6-33　窗帘图

6.6.2 三角形网格图

用 trimesh 函数生成三角形网格图，其调用格式如下。

- trimesh(Tri,X,Y,Z)　　显示由 *m*×3 的矩阵 Tri 定义的三角形网格。Tri 的每一行通过给向量或矩阵 X，Y 和 Z 赋指数来定义单个三角形。
- trimesh(Tri,X,Y,Z,C)　　用同样的方式作为 surf 函数定义 C 来指定颜色。
- trimesh(…'PropertyName',PropertyValue…)　　为函数创建的阴影图形对象指定附加阴影属性名和属性值。
- h = trimesh(…)　　返回阴影图形对象的句柄。

【例 6-17】　下面程序创建顶点向量和表面矩阵，然后创建一个三角形网格图，绘制结果如图 6-34 所示。

```
x = rand(1,50);
y = rand(1,50);
z = peaks(6*x-3,6*x-3);
tri = delaunay(x,y);
        trimesh(tri,x,y,z)
```

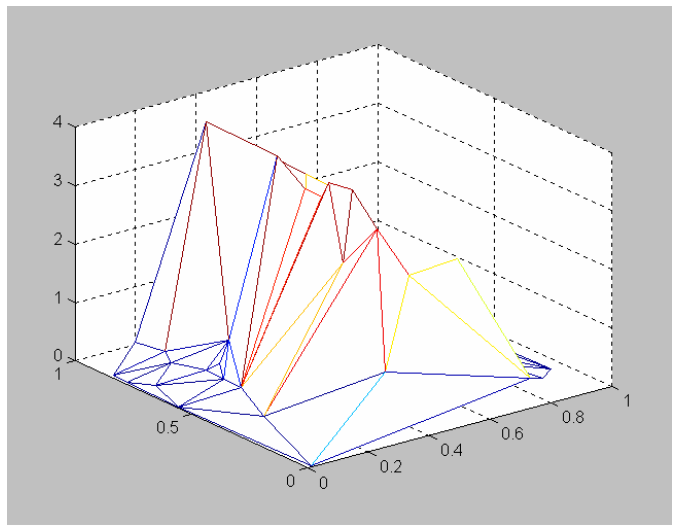


图 6-34　三角形网格图

6.7 三维表面图

将三维网格图中的三角形单元或四边形单元用颜色进行填充，生成三维表面图，对应的有三角形表面图和四边形表面图。

6.7.1 四边形表面图

用 surf 函数绘制四边形表面图，其调用格式如下。

- surf(Z)　　用 X = 1:n 和 Y = 1:m，其中，[m,n] = size(Z)，根据 Z 矩阵中的 z 元素创建一个三维阴影表面图。高度 Z 为一单值函数，定义在一个矩形网格上。Z 指定颜色数

据和表面高度，颜色与表面高度成比例。

- surf(X,Y,Z)　　用 Z 作为颜色数据和表面高度数据创建阴影表面图。X 和 Y 为向量或矩阵。若 X 和 Y 为向量，则 length(X) = n ，length(Y) = m，其中，[m,n] = size(Z)。
- surf(X,Y,Z,C)　　用 C 定义的颜色创建阴影表面图。
- surf(…,'PropertyName',PropertyValue)　　同时指定表面属性。
- surfc(…)　　在表面图下方绘一个等值线图。
- h = surf(…) 和 h = surfc(…)　　返回表面图形对象的句柄。

【例 6-18】　下面程序生成 peaks 函数的表面图和等值线图，效果如图 6-35 所示。

```
[X,Y,Z] = peaks(30);
surfc(X,Y,Z)
colormap hsv
axis([-3 3 -3 3 -10 5])
```

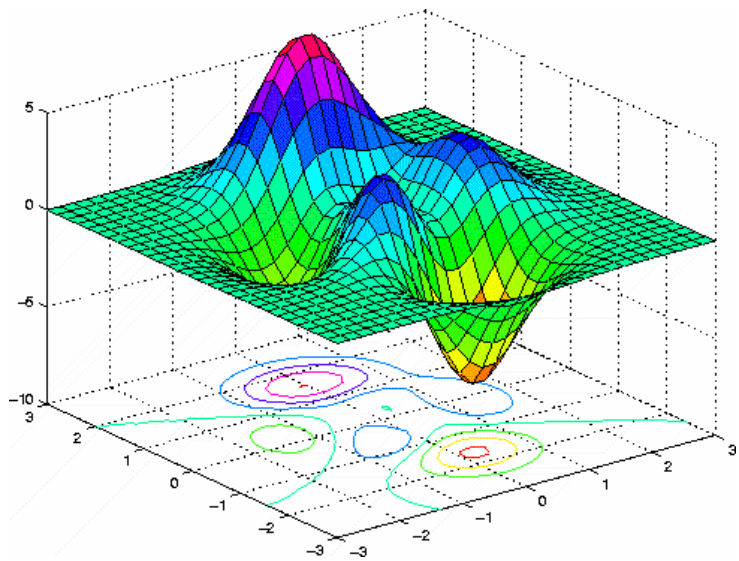


图 6-35　四边形表面图

6.7.2　三角形表面图

用 trisurf 函数生成三角形表面图，其调用格式如下。

- trisurf(Tri,X,Y,Z)　　显示 $m\times3$ 的矩阵 Tri 定义的三角形网格，并作为表面。Tri 的每一行通过给向量或矩阵 X，Y 和 Z 赋指数来定义单个三角形。
- trisurf(Tri,X,Y,Z,C)　　用同样的方式作为 surf 函数定义 C 来指定颜色。
- trisurf(…'PropertyName',PropertyValue…)　　为函数创建的阴影图形对象指定附加阴影属性名和属性值。
- h = trisurf(…)　　返回一个阴影句柄。

【例 6-19】　下面程序创建顶点向量和表面矩阵，然后创建一个三角形表面图。

```
x = rand(1,50);
y = rand(1,50);
z = peaks(6*x-3,6*x-3);
```

```
tri = delaunay(x,y);
trisurf(tri,x,y,z)
```

生成的图形如图 6-36 **所示。**

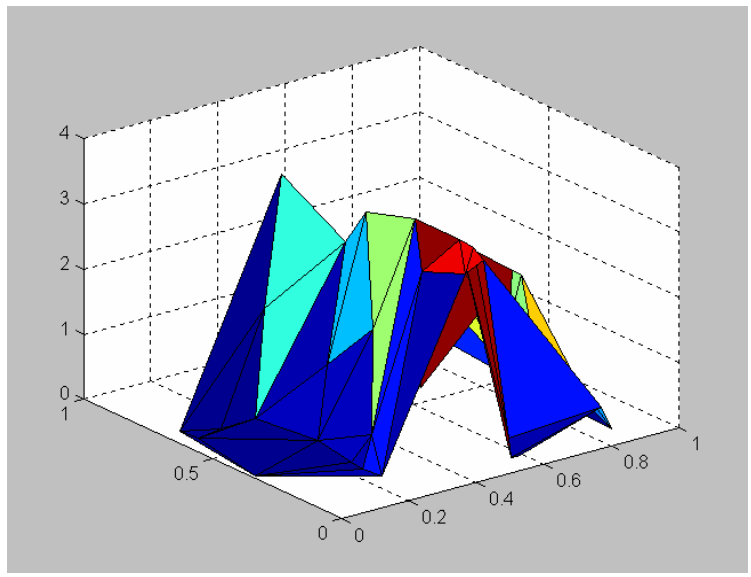


图 6-36　**三角形表面图**

6.8　三维曲面图

三维曲面图在三维表面图的基础上生成，它对三维表面图中的三角形单元或四边形单元进行了平滑处理，使其成为曲面片，因而更接近实体外观。

使用 surfl **函数，用基于色图的光照绘制表面图，其调用格式如下。**

- surfl(Z)**和** surfl(X,Y,Z)　　**用默认的光源方向设置和默认的阴影模型光线系数创建三维阴影表面图。**X，Y **和** Z **为向量或矩阵，定义表面的** *x*，*y* **和** *z* **分量。**
- surfl(…,'light')　　**用光线对象生成一个彩色的、有明暗调子的表面。**
- surfl(…,s)　　**指定光源的方向。**s **是一个二元素或三元素向量，指定表面到光源的方向** s = [sx sy sz]**或** s = [azimuth elevation]**。默认设置为从当前视角逆时针旋转** 45 **度。**
- surfl(X,Y,Z,s,k)　　**指定反射常数。**k **为四元素向量，定义光线、漫反射、特殊反射和特殊照射系数等。**k = [ka kd ks shine]**，默认值为**[.55,.6,.4,10]**。**
- h = surfl(…)　　**返回表面图形对象的句柄。**

【例 6-20**】　下面程序用基于色图的光照生成** peaks **函数的表面图，如图** 6-37 **所示。**

```
[x,y] = meshgrid(-3:1/8:3);
z = peaks(x,y);
surfl(x,y,z);
shading interp
colormap(gray);
axis([-3  3  -3  3  -8  8])
```

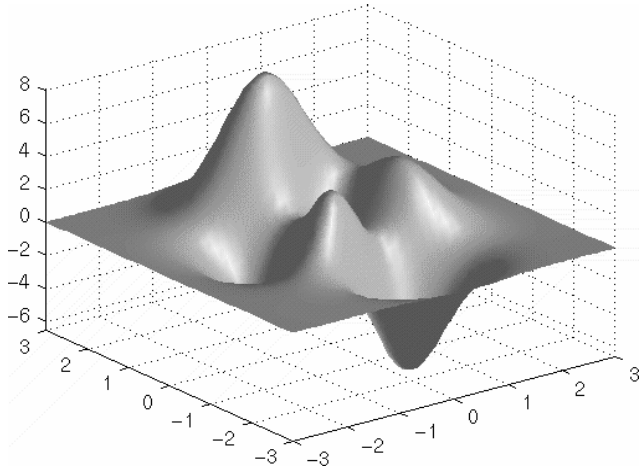
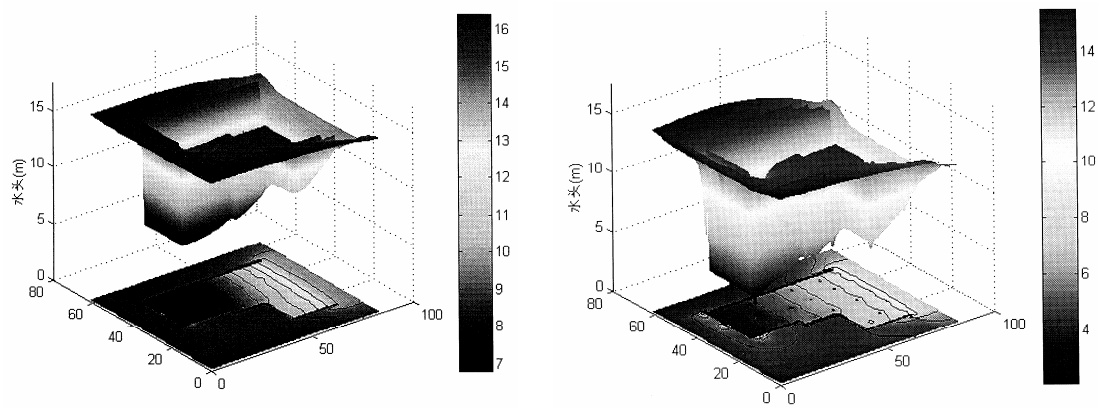


图 6-37　灰度阴影图

6.9　云图

云图用渐变色来表现某个量在三维空间上的变化情况。该图的特点是形象、直观而且华丽，给人印象深刻。

图 6-38 中图(a)，(b)演示了基坑降水过程中不同时段的降深情况。各图中基坑及其周边一定范围内的地下水位用不同颜色来表现，十分清晰。从图(a)到图(b)，计算范围内的地下水位不断下降，甚至可以看见井点处的降落漏斗。



(a)　　(b)

图 6-38　云图示例

在 MATLAB 中绘制云图很简单，即在绘制三维曲面图以后，用 shading 函数设置曲面表面的颜色就行了。下例所示，用球面坐标绘制一个椭球，然后用 shading 函数着过渡色。

```
ellipsoid(0,0,0,10,60,150)
axis square
shading interp
```

绘制结果如图 6-39 所示。

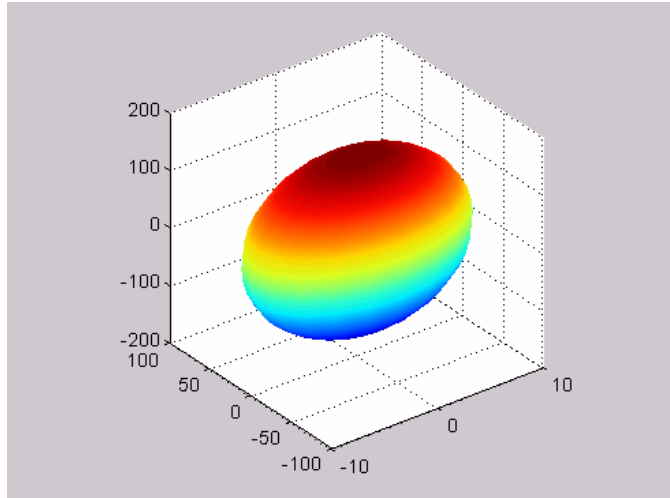


图 6-39　一个椭球的云图

6.10　视图控制

本节介绍怎样定义不同的视图参数，以获得希望得到的视图效果。一般地讲，视图控制应用于三维图形或模型，通过调整二维图形的方向比可以达到指定的比例或使图形拟合成一定的形状。

MATLAB 视图控制包括以下两个内容：

（1）设置比例　设置方向比和相对坐标轴比例，控制显示对象的形状。

（2）放置视点　讨论怎样用 aximuth 和 elevation 指定看图的视点。

创建图形时，MATLAB 会自动设置视图。它实际选择什么样的视图决定于用户创建二维图还是三维图。

MATLAB 让用户可以控制坐标轴中图形显示的方向；可以指定视点、视图目标、方向和图形窗口中视图显示的程度，这些视图特点由一系列图形属性控制；可以直接指定这些属性值或使用 view 命令选择一个视图方向，并利用 MATLAB 的自动属性选择功能定义一个合适的视图。

view 命令通过定义与坐标轴原点相对应的 azimuth 和 elevation 参数指定视点。azimuth 参数是 *x*-*y* 面板上的极角，正的角度表示从视点沿反时针方向旋转。elevation 参数为高于或低于 *x*-*y* 面板的角度。

MATLAB 根据图形是二维图还是三维图来自动选择视点。

对于二维图，默认值 azimuth = 0°，elevation = 90°。

对于三维图，默认值 azimuth = -37.5°，elevation = 30°。

例如，下面这些命令行创建一个三维表面图并按默认的三维视图设置显示。创建结果如图 6-40 所示。

```
[X,Y] = meshgrid([-2:.25:2]);
Z = X.*exp(-X.^2 -Y.^2);
surf(X,Y,Z)
```

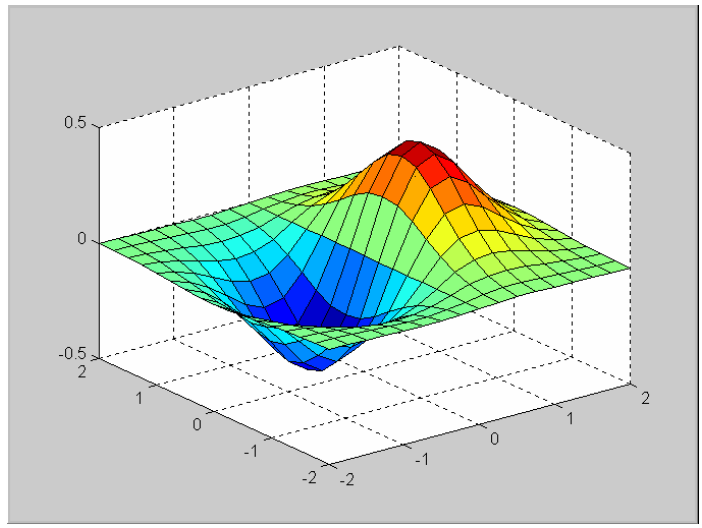


图 6-40　视图设置之一

命令行

```
view([180 0])
```

表示在 z=0 **的高度上沿** y **的负方向看图，如图** 6-41 **所示。**

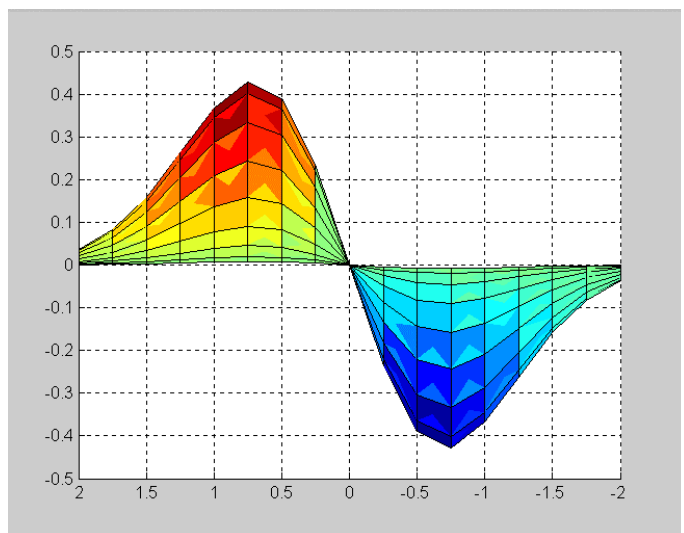


图 6-41　视图设置之二

可以使用负的高度将视点移到坐标轴原点以下的位置。例如：

```
view([-37.5 -30])
```

生成的图形如图 6-42 **所示。**

用 azimuth **和** elevation **参数指定视点，有一定的范围限制。它不允许用户指定视点的实际位置，其方向和** z **轴始终指向上方。它不允许缩小和放大图形或进行强制性旋转和转换。**

MATLAB **中的相机图形工具提供了对简单调节的更强大控制。下面的内容讨论怎样使用相机属性控制视图。**

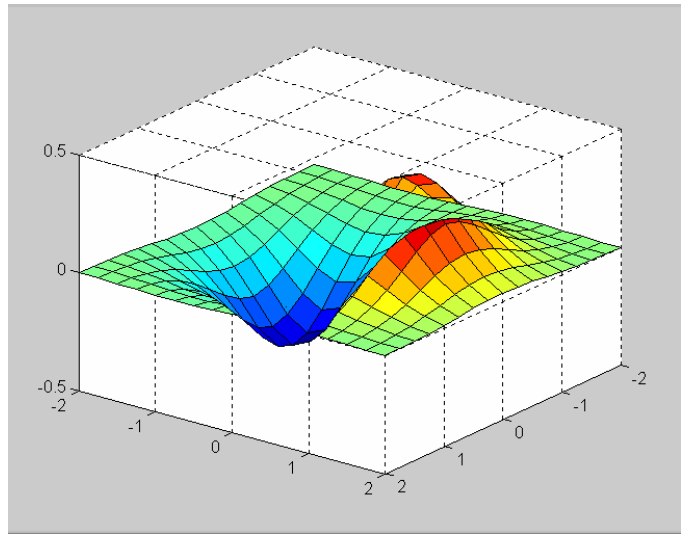


图 6-42　视图设置之三

6.11　光照控制

对于图景，进行光照投影是增加其真实性的有效技巧，它通过模拟自然光下物体的明暗调子来达到该目的。MATLAB 通过调用 light 对象来定义图形。

用 lighting 函数选择光照算法，其调用格式如下。

- lighting　　选择算法，计算当前坐标系中光线照在对象所有表面和阴影对象上的效果。
- lighting flat　　选择平面光照。
- lighting gouraund　　选择 gouraud 光照。
- lighting phong　　选择 phong 光照。
- lighting none　　不设置光照。

注意：surf, mesh, pcolor, fill, fill3, surface 和 patch 函数创建被光源影响的图形对象。lighting 命令为图形的表面和阴影设置合适的 FaceLighting 和 EdgeLighting 属性。

用 camlight 函数在摄影坐标系中创建和删除光线对象，其调用格式如下。

- camlight('headlight')　　在相机位置创建一条光线。
- camlight('right')　　创建源于相机右方和上方的光线。
- camlight('left')　　创建源于相机左方和下方的光线。
- camlight 与 camlight('right')效果相同。
- camlight(az,el)　　创建相对于相机位置指定 azimuth (az)和 elevation (el)的光线。摄影目标为旋转中心，az 和 el 用度表示。

  camlight(…,'style') style 变量可以有以下两个值：

  ➢ local (默认选项) —— 为点光源，光线向周围的所有方向照射。

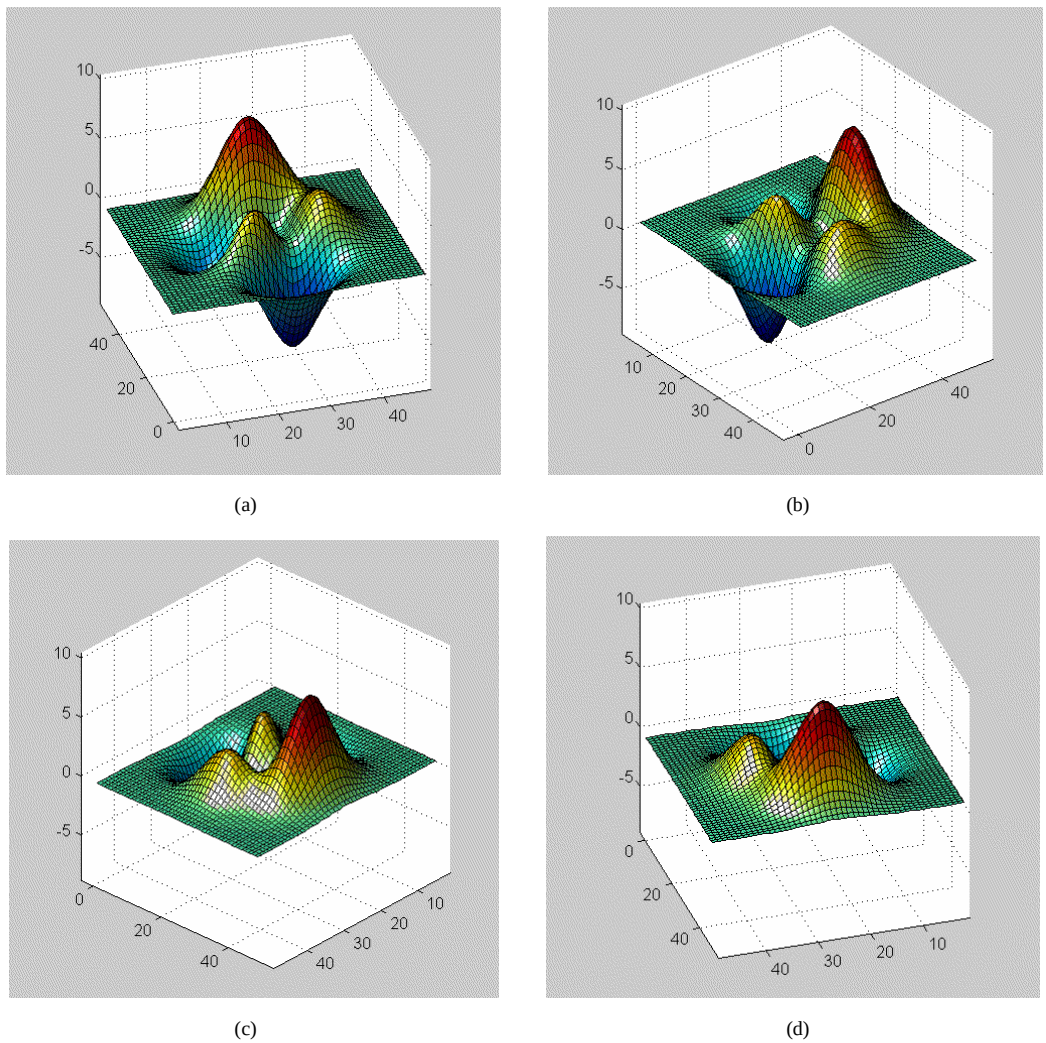
  ➢ infinite —— 光线为平行线。
- camlight(light_handle,…)　　使用 light_handle 指定的光线。
- light_handle = camlight(…)　　返回光线的句柄。

【例 6-21】　下面的程序创建一条射向相机左侧的光线，然后每次移动相机位置都重新

设置光线的位置。

```
surf(peaks)
axis vis3d
h = camlight('left');
for i = 1:20;
    camorbit(10,0)
    camlight(h,'left')
    drawnow;
end
```

生成的图形如图 6-43 **中图**(a)**至**(d)**所示。**



(a) (b)

(c) (d)

图 6-43　**移动相机位置后的效果图**

下面这些内容描述如何使用对象的不同透明属性，以改进视图。

- Making Objects Transparent —— **提供指定对象透明度属性的概览。**
- Specifying a Single Transparency Value —— **描述怎样指定透明属性值，它将用于图形**

对象的所有表面。

- Mapping Data to Transparency —— 显示怎样将透明度作为数据可视化的另一维。
- Selecting an Alphamap —— 描述不同图形的特点并演示其生成效果。

下面是几个具体的应用实例，读者朋友可以从中领略视图控制和光照控制的妙处。

1．沿图景移动相机

景物的移动效果是通过在三维空间中移动相机来得到的。

下面的例子利用穿越效果查看等表面的内部。该等表面是用 wind 速率向量场定义的，它代表北美洲上空的气流。

用到的技巧包括：

- 用等表面和圆锥图演示三维区域内部的气流流动。
- 利用光照演示三维区域内部的等表面和锥体。
- 用流线定义相机穿越三维区域内部的途径。
- 相机位置、目标和光照等的坐标。

（1）体积数据图示　第一步是画一个等表面并用锥体图表示空气的流动。

在绘锥体图之前将数据方向比（daspect）设置为[1,1,1]。

```
load wind
wind_speed = sqrt(u.^2 + v.^2 + w.^2);
hpatch = patch(isosurface(x,y,z,wind_speed,35));
isonormals(x,y,z,wind_speed,hpatch)
set(hpatch,'FaceColor','red','EdgeColor','none');
[f vt] = reducepatch(isosurface(x,y,z,wind_speed,45),0.05);
daspect([1,1,1]);
hcone = coneplot(x,y,z,u,v,w,vt(:,1),vt(:,2),vt(:,3),2);
set(hcone,'FaceColor','blue','EdgeColor','none');
```

生成的图形如图 6-44 所示。

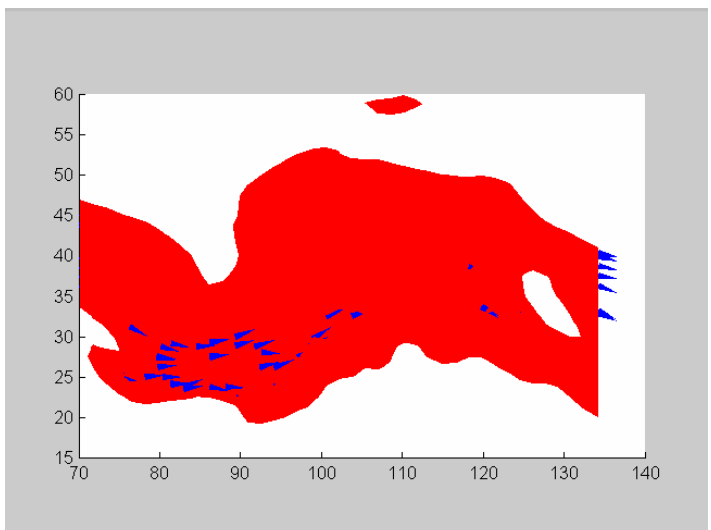


图 6-44　等表面叠加锥体图

（2）设置视图　需要定义视图参数，以保证需要图形正确显示。

选择透视投影，提供相机穿越等表面内部时的深度探察（camproj）。设置相机视角为固定值，MATLAB 自动将整个图景进行压缩，就像缩小了一样（camva）。

```
camproj perspective
camva(25)
```

（3）指定光源　将光源的位置放在相机处，并修改等表面和圆锥体的反射特性，改进图景的真实性。

在相机位置创建一个光源，提供一个随相机在等表面内部移动的“头灯”。

设置等表面的反射特性，用高反射物质（将 SpecularStrength 和 DiffuseStrength 参数设置为 1）使得等表面内部有一些黑色的阴影(将 AmbientStrength 参数设置为 0.1)。

将圆锥体的 SpecularStrength 属性设置为 1，使它们高反射。

```
hlight = camlight('headlight');
set(hpatch,'AmbientStrength',.1,…
        'SpecularStrength',1,…
        'DiffuseStrength',1);
set(hcone,'SpecularStrength',1);
set(gcf,'Color','k')
```

生成的图形如图 6-45 所示。

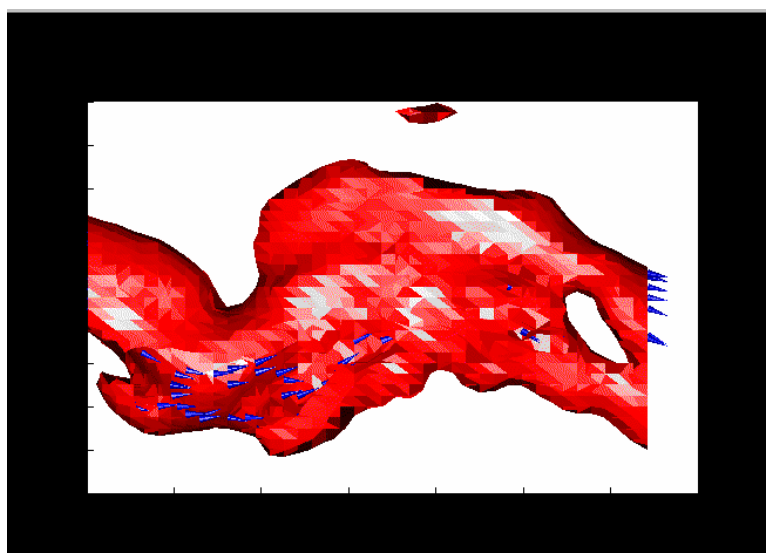


图 6-45　指定光源

（4）选择演示器　因为本例使用光照，MATLAB 必须使用 zbuffer 演示器或 OpenGL 演示器。OpenGL 演示器在显示快照时速度更快，此时需要设置地灯，其效果没有设置 phong 投影（可以与 zbuffer 演示器一起使用）时平滑。

```
lighting gouraud
set(gcf,'Renderer','OpenGL')
```

生成的图形如图 6-46 所示。

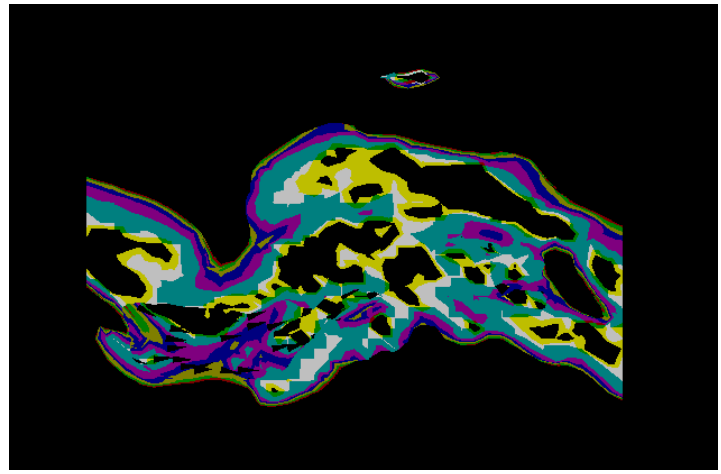


图 6-46　使用 OpenGL 演示器的效果图

或者使用 zbuffer 演示器：

```
lighting phong
set(gcf,'Renderer','zbuffer')
```

生成的图形如图 6-47 所示。

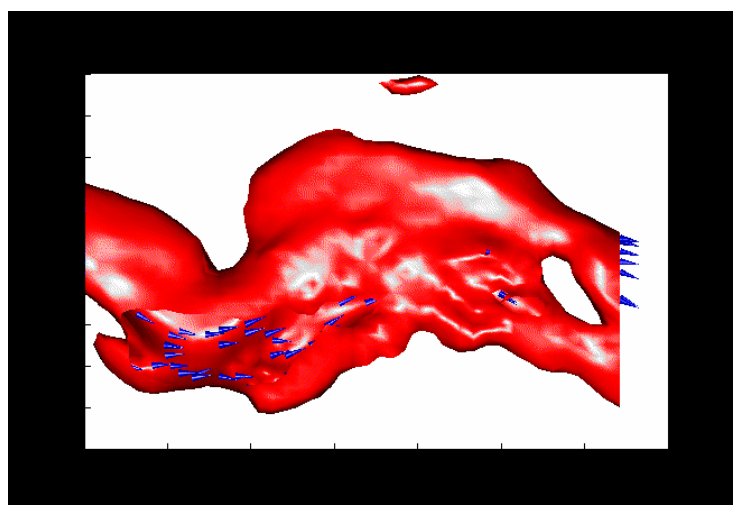


图 6-47　使用 zbuffer 演示器的效果图

2．飞越快照显示

（1）将相机移动途径定义为流线　流线表示了向量场中的流动方向。本例使用单条流线的 x，y 和 z 坐标数据绘制移动途径。步骤如下：

- **创建一个起点为 $x = 80$，$y = 30$，$z = 11$ 的流线图。**
- **获取流线的 x，y 和 z 坐标数据。**
- **删除流线。**

```
hsline = streamline(x,y,z,u,v,w,80,30,11);
xd = get(hsline,'XData');
yd = get(hsline,'YData');
```

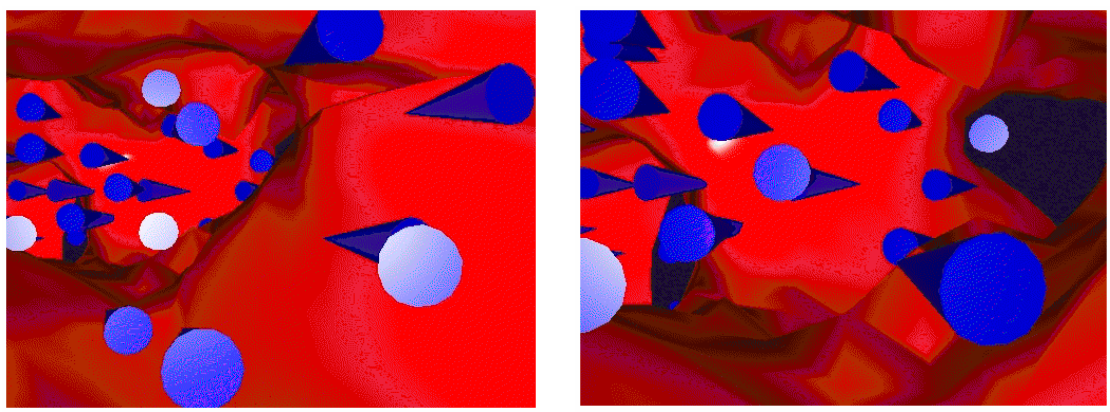
```
zd = get(hsline,'ZData');
delete(hsline)
```

（2）完成飞越　创建飞越时，沿同样的途径移动相机位置和相机目标。本例中，相机目标放在沿 *x* 轴比相机位置前 5 个单元的位置上。同样，在相机目标 *x* 位置上放了一个小值，防止相机位置和目标成为同一个点（当 xd(n) = xd(n+5)**时会出现这种情况）。**

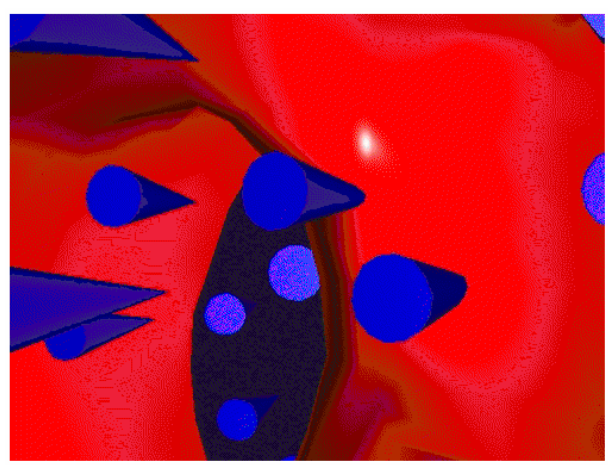
- **更新相机位置和相机目标，使得它们都沿流线坐标移动。**
- **沿相机移动光照。**
- **调用** drawnow **函数，显示每次移动的结果。**

```
for i=1:length(xd)-50
   campos([xd(i),yd(i),zd(i)])
   camtarget([xd(i+5)+min(xd)/100,yd(i),zd(i)])
   camlight(hlight,'headlight')
   drawnow
end
```

图 6-48 **中图**(a),(b)**和**(c)**分别为** i **值等于** 10，110 **和** 185 **时的快照视图。**



(a)　　(b)



(c)

图 6-48　**快照视图**

3．切片液流数据视图

本例对 flow M **文件生成的体积进行切片绘图。**

（1）数据探察

用下面的命令行生成体积数据：

```
[x,y,z,v] = flow;
```

通过找到坐标数据的最小值和最大值来确定体积的范围：

```
xmin = min(x(:));
ymin = min(y(:));
zmin = min(z(:));
xmax = max(x(:));
ymax = max(y(:));
zmax = max(z(:));
```

（2）在相对于 *x* **轴的某个角度上创建切片面板。创建不在坐标轴平面上的切片面板，首先定义表面并旋转到需要的方位。本例使用具有相同** *x* **和** *y* **坐标的表面。**

```
hslice = surf(linspace(xmin,xmax,100),…
    linspace(ymin,ymax,100),…
    zeros(100));
```

生成的图形如图 6-49 **所示。**

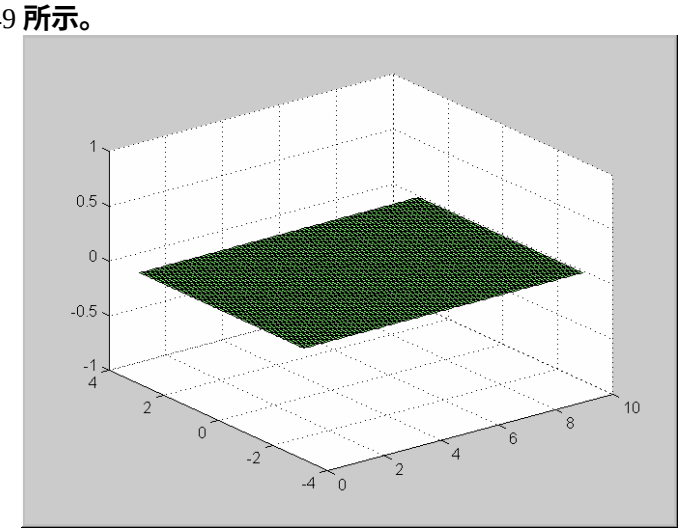


图 6-49　**创建切片面板**

绕 *x* **轴** 45 **度旋转表面，并保存表面** Xdata，Ydata **和** Zdata，**定义切片面板，然后删除表面。**

```
rotate(hslice,[-1,0,0],-45)
xd = get(hslice,'XData');
yd = get(hslice,'YData');
zd = get(hslice,'ZData');
delete(hslice)
```

（3）绘切片面板　绘制旋转的切片面板，设置 FaceColor **属性为** interp，**绘制色谱图并将** EdgeColor **设为** none。**将** DiffuseStrength **属性增加到** 0.8，**添加光源以后使该面板更明亮。**

```
h = slice(x,y,z,v,xd,yd,zd);
```

```
set(h,'FaceColor','interp',…
    'EdgeColor','none',…
    'DiffuseStrength',.8)
```

绘制结果如图 6-50 **所示。**

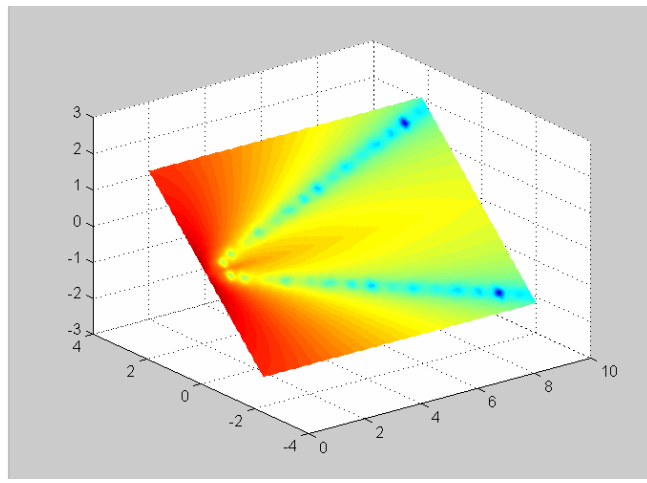


图 6-50　**旋转的切片面板**

将 hold **命令设置为** on**，并在** xmax**，**ymax **和** zmin **处添加** 3 **个以上的正交切片面板，提供第一个面板的承接。**

```
hold on
hx = slice(x,y,z,v,xmax,[],[]);
set(hx,'FaceColor','interp','EdgeColor','none')
hy = slice(x,y,z,v,[],ymax,[]);
set(hy,'FaceColor','interp','EdgeColor','none')
hz = slice(x,y,z,v,[],[],zmin);
set(hz,'FaceColor','interp','EdgeColor','none')
```

生成的图形如图 6-51 **所示。**

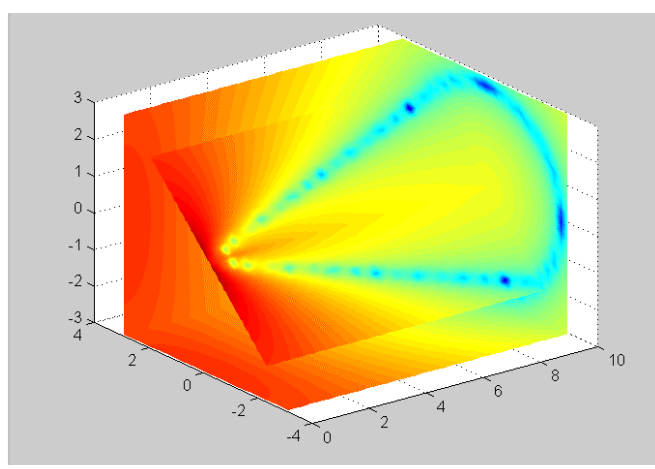


图 6-51　**添加** 3 **个正交的切片面板**

（4）定义视图　按正确的比例显示三维区域，将数据方向比 daspect 设置为 [1,1,1]。调整坐标轴，并打开一个方框，提供三维意义上的对象。坐标轴的方位可以用 rotate3d 函数选择并确定最佳的视图。缩小图景，提供三维区域更大的视野。选择透视投影类型，使方形实体更自然。

```
daspect([1,1,1])
axis tight
box on
view(-38.5,16)
camzoom(1.4)
camproj perspective
```

效果如图 6-52 所示。

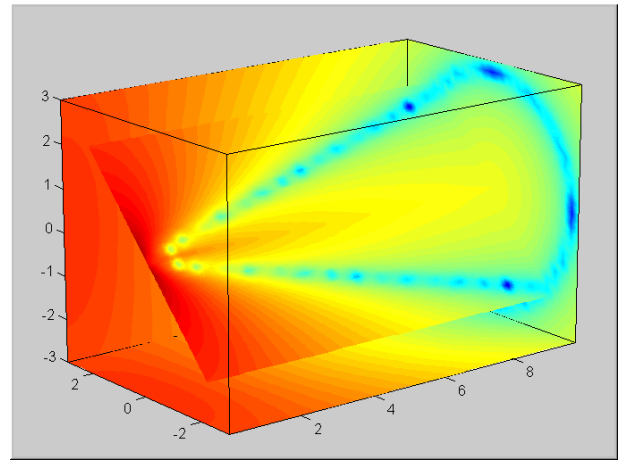


图 6-52　定义视图

（5）添加光照并指定颜色　给图景添加光照，使 4 个切片面板之间的边界更加明显。选择一个只有 24 色的色谱图（默认时为 64 色），帮助指示实体内部的变化。

```
lightangle(-45,45)
colormap (jet(24))
set(gcf,'Renderer','zbuffer')
```

生成的图形如图 6-53 所示。

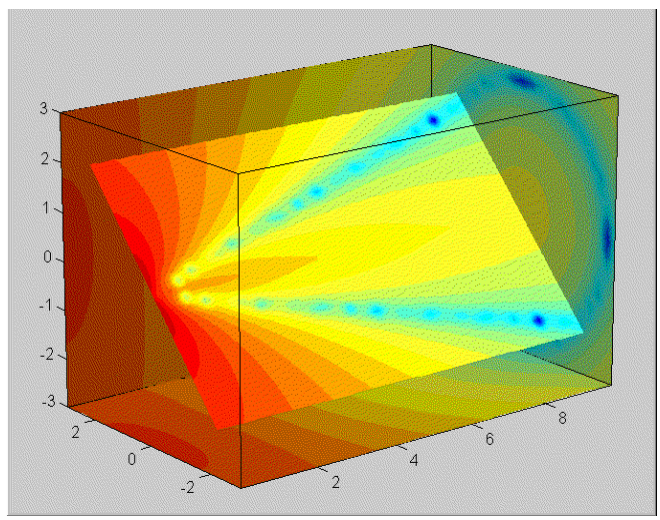


图 6-53　添加光照

6.12　综合实例

6.12.1　向量数据的流线图

MATLAB 含有一个称为 wind 的向量数据集合，它代表流过北美洲的气流。本例使用了下面一些技巧：

- 用流线跟踪气流速度。
- 用切片面板显示数据的横断面视图。
- 用切片面板上的等值线改进切片面板颜色的可视性。

（1）确定坐标范围　装载数据并确定切片面板和等值线图所处的位置。

```
load wind
xmin = min(x(:));
xmax = max(x(:));
ymax = max(y(:));
zmin = min(z(:));
```

（2）添加切片面板　计算向量场的大小（它代表气流流速），为 slice 命令生成标量数据。沿 *x* 轴在 xmin 处、沿 *y* 轴在 ymax 处、沿 *z* 轴在 zmin 处创建切片面板图。指定表面颜色，用切片的颜色指示气流速度。

```
wind_speed = sqrt(u.^2 + v.^2 + w.^2);
hsurfaces = slice(x,y,z,wind_speed,[xmin,100,xmax],ymax,zmin);
set(hsurfaces,'FaceColor','interp','EdgeColor','none')
```

生成的图形如图 6-54 所示。

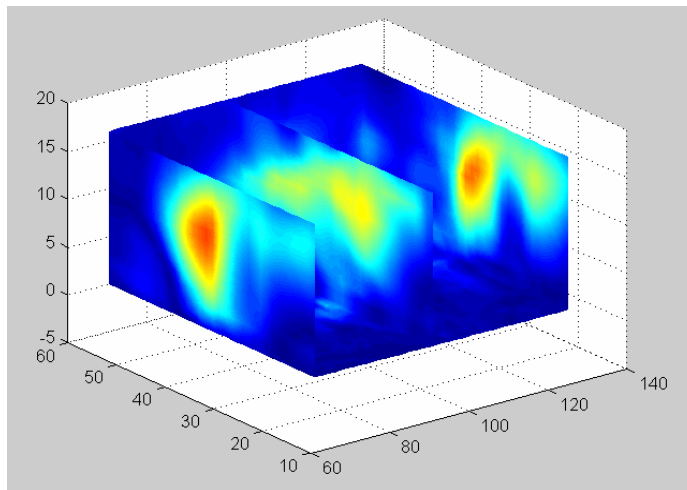


图 6-54　添加切片面板

（3）添加等值线到切片面板　在切片面板上绘色谱等值线图。

```
hcont = contourslice(x,y,z,wind_speed,[xmin,100,xmax],ymax,zmin);
set(hcont,'EdgeColor',[.7,.7,.7],'LineWidth',.5)
```

生成的图形如图 6-55 所示。

（4）定义流线　在本例中，所有的流线起始于 *x*=80 处，并在 *y* 方向 20～50，*z* 方向 0～15 的范围内扩展。保存流线的句柄并设置线条的宽度和颜色。

```
[sx,sy,sz] = meshgrid(80,20:10:50,0:5:15);
hlines = streamline(x,y,z,u,v,w,sx,sy,sz);
set(hlines,'LineWidth',2,'Color','r')
```

生成的图形如图 6-56 所示。

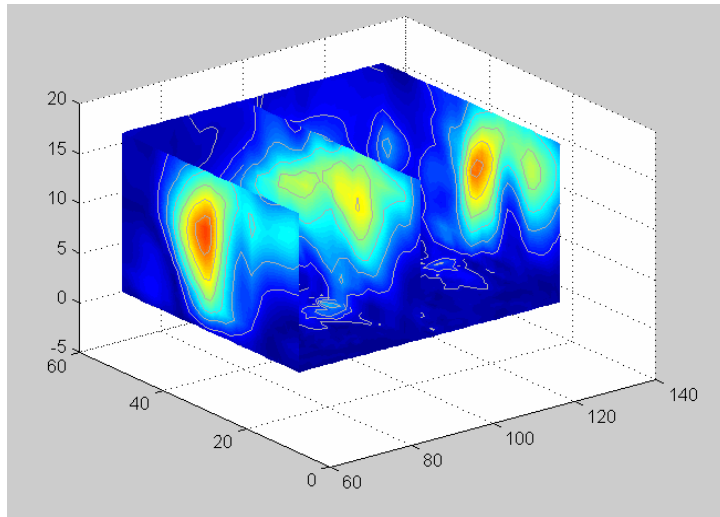


图 6-55　在切片面板中添加等值线

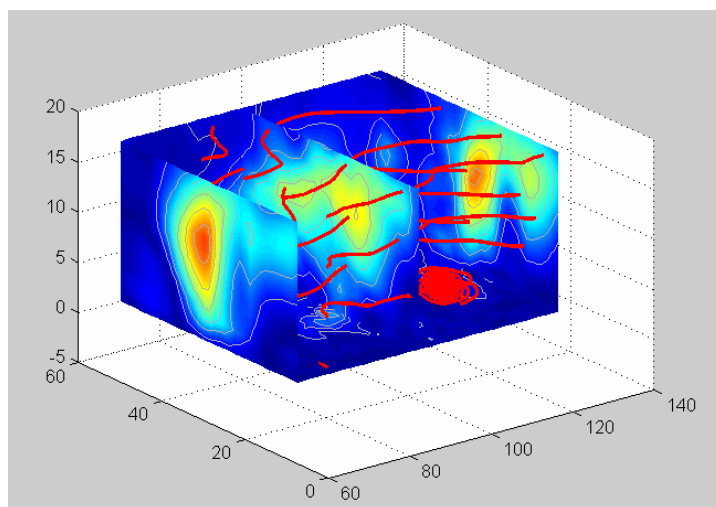


图 6-56　添加流线

（5）定义视角　抬升视角，扩展 *z* 轴，使得图形的阅读更加容易。

```
view(3)
daspect([2,2,1])
axis tight
```

生成的图形如图 6-57 所示。

6.12.2　用流动条带显示卷曲

与流线一样，流带也可以演示流动的方向，同时它还能演示流体沿轴线的旋转。使用 streamribbon 函数，可以指定流带上每个顶点的扭曲角度（用弧度表示）。

与 curl 函数联合使用，流带图可以帮助显示向量场的卷曲角速度。

（1）选择进行绘图的数据子集　用 subvolume 函数在 wind 数据集中选择一个感兴趣的区域，首先用所有的数据集进行绘图可以帮助作出选择。

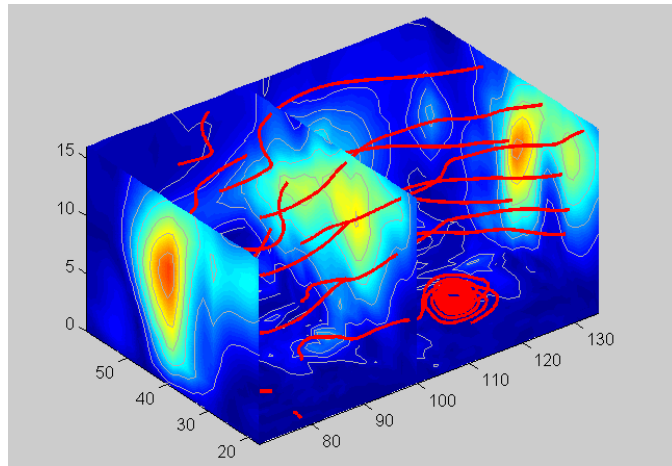


图 6-57　抬升视角以后的图形

```
load wind
lims = [100.64 116.67 17.25 28.75 -0.02 6.86];
[x,y,z,u,v,w] = subvolume(x,y,z,u,v,w,lims);
```

（2）计算卷曲角速度和气流流速

```
cav = curl(x,y,z,u,v,w);
wind_speed = sqrt(u.^2 + v.^2 + w.^2);
```

（3）生成流带图

```
[sx sy sz] = meshgrid(110,20:5:30,1:5);
verts = stream3(x,y,z,u,v,w,sx,sy,sz,.5);
h = streamribbon(verts,x,y,z,cav,wind_speed,2);
set(h,'FaceColor','r',…
    'EdgeColor',[.7 .7 .7],…
    'AmbientStrength',.6)
```

生成的图形如图 6-58 所示。

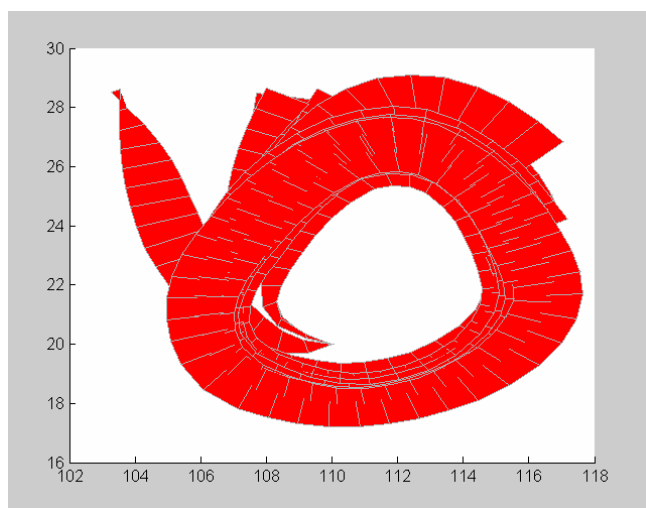


图 6-58　流带图显示卷曲

（4）设置视角并添加光照　用 volumebounds 命令设置坐标轴和颜色的范围比较方便。添加网格，设置三维视图。camlight 函数设置光线位置，lighting 函数将光照方法设置为 Phong。

```
axis(volumebounds(x,y,z,wind_speed))
grid on
view(3)
camlight right;
set(gcf,'Renderer','zbuffer'); lighting phong
```

生成的图形如图 6-59 所示。

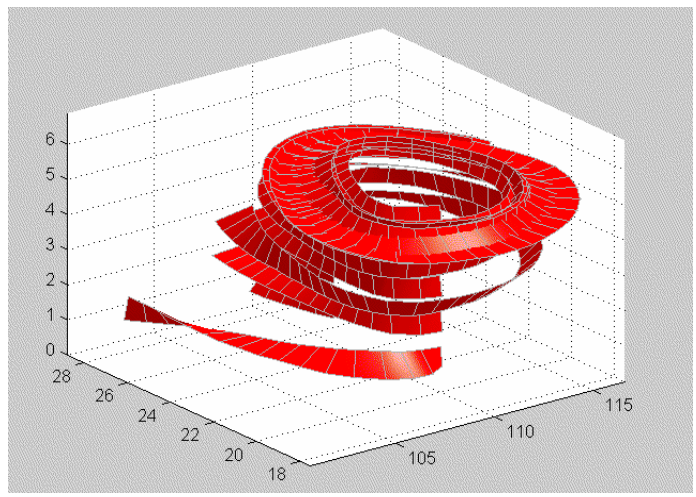


图 6-59　设置视角并添加光照以后的流带图

6.12.3　用流管显示差异

流管与流线相似，但流管有宽度，提供了另一维上的信息。默认时，MATLAB 用管体的宽度表示向量场的差异。

这里用到下面一些技巧：

- 流管表示气流数据集合的流动方向和向量场的差异。
- 用不同颜色表示的切片面板指示气流速度，添加等值线使得可视程度更高。

输入参数包括体积、向量场组分和流管起点位置的坐标。

（1）装入数据，计算需要的数值。

第一步装入数据并计算需要绘图的数值。这些数值包括：

- 切片面板的位置（x 的最小值、y 的最小值和高程）。
- 流管起点的最小值。
- 气流的流速(向量场的大小)。

```
load wind
xmin = min(x(:));
xmax = max(x(:));
ymin = min(y(:));
alt = 7.356; % z-value for slice and streamtube plane
wind_speed = sqrt(u.^2 + v.^2 + w.^2);
```

（2）绘切片面板　绘切片面板，设置表面属性，创建一个具有平滑颜色的切片。从 hsv 色谱图中取 16 种颜色。

```
hslice = slice(x,y,z,wind_speed,xmax,ymin,alt);
set(hslice,'FaceColor','interp','EdgeColor','none')
colormap hsv(16)
```

生成的图形如图 6-60 所示。

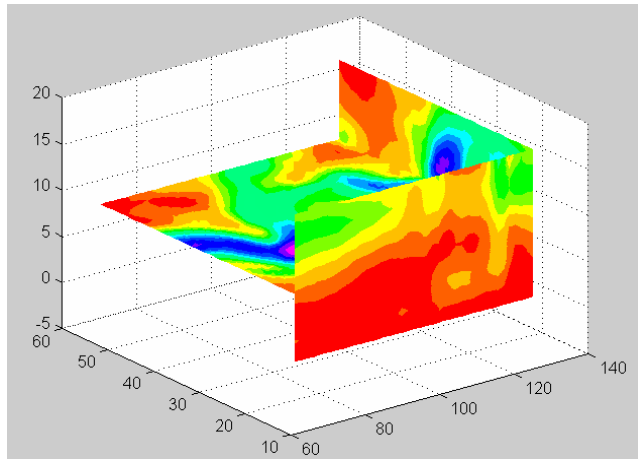


图 6-60　切片面板

（3）添加等值线图到切片面板中　调整等值线间距，使得在切片面板中直线与颜色边界匹配。调用 caxis 函数，获得当前颜色的范围；设置 contourslice 函数的参数确定内插的方法。

```
color_lim = caxis;
cont_intervals = linspace(color_lim(1),color_lim(2),17);
hcont = contourslice(x,y,z,wind_speed,xmax,ymin,…
    alt,cont_intervals,'linear');
set(hcont,'EdgeColor',[.4 .4 .4],'LineWidth',1)
```

生成的图形如图 6-61 所示。

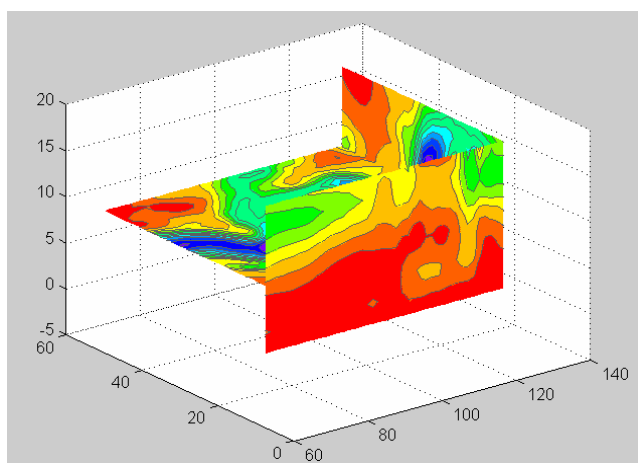


图 6-61　在切片面板中添加等值线

（4）创建流管图　使用网格线创建流管起点数组，其起点位于最小 x 值处，在 y 方向上从 20 到 50，在 z 方向上则位于某一面板中。

流管图在指定的地点绘制，其宽度是默认宽度的 1.25 倍，以强调差异显示。向量[1.25 30]中的第二个元素指示沿管的周长方向的点数（默认值为 20）。

在调用 streamtube 函数之前还要设置数据方向比（daspect）。

在通过设置表面属性控制流管的表面特性以前，流管为表面对象。本例将表面属性设置为高亮照射、红色表面。

```
[sx,sy,sz] = meshgrid(xmin,20:3:50,alt);
daspect([1,1,1]) % set DAR before calling streamtube
htubes = streamtube(x,y,z,u,v,w,sx,sy,sz,[1.25 30]);
set(htubes,'EdgeColor','none','FaceColor','r',…
    'AmbientStrength',.5)
```

绘制结果如图 6-62 所示。

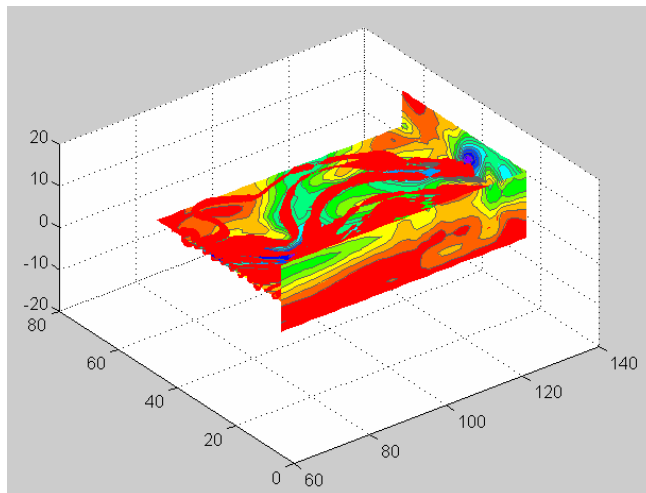


图 6-62　创建流管图

（5）定义视角，添加光照　最后一步是定义视角，添加光照。

```
view(-100,30)
axis(volumebounds(x,y,z,wind_speed))
set(gca,'Projection','perspective')
camlight left
```

生成的图形如图 6-63 所示。

6.12.4　创建流动微粒快照

流动微粒快照对于向量场流动方向和流动速度的可视化是很有帮助的。"particles"（用任意线型标记表示)沿特定的流线进行流体跟踪。快照中每一个微粒的速度与流线上任意给定点处向量场的大小成比例。

（1）指定绘图数据范围的起点　本例通过指定近似起点来确定所绘图形的体积区域。这里，流线的起点为 $x = 100$, y 界于 20 和 50 之间，并且在 $z = 5$ 的面板上。

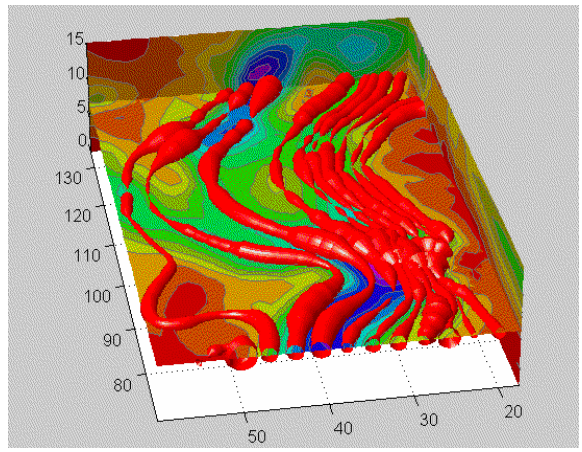


图 6-63　定义视角和添加光照

```
load wind
[sx sy sz] = meshgrid(100,20:2:50,5);
```

（2）创建流线图　表示微粒流动途径。本例使用流线跟踪快照微粒的流动路径。

```
verts = stream3(x,y,z,u,v,w,sx,sy,sz);
sl = streamline(verts);
```

生成的图形如图 6-64 所示。

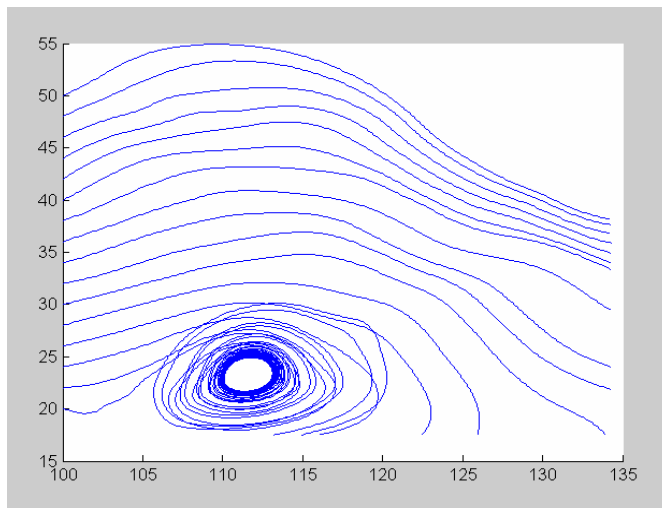


图 6-64　创建流线图

（3）定义视图　下面的设置提供了一个快照的清晰视图。

所选择的视点显示了包含大部分流线的平面和包含螺旋线的平面。选择数据方向比为 [2 2 0.125]**。设置坐标轴范围，使之与数据范围匹配，并绘坐标轴方框。**

```
view(-10.5,18)
daspect([2 2 0.125])
axis tight; box on
```

生成的图形如图 6-65 所示。

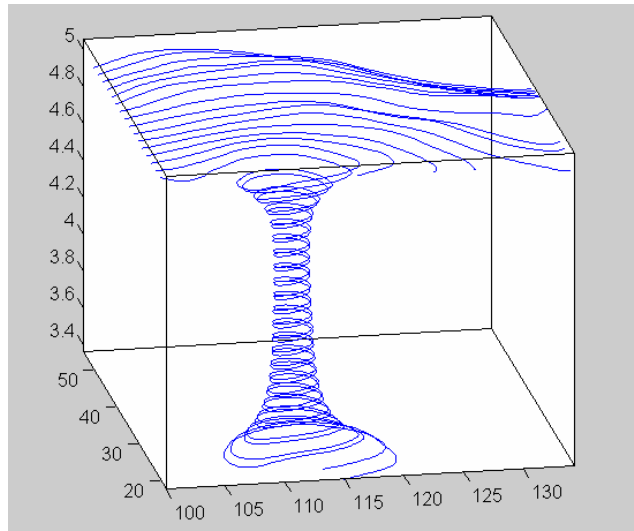


图 6-65　定义视图

（4）计算流动微粒顶点　第一步确定沿流线的微粒。interpstreamspeed 函数返回基于流线顶点的数据和向量数据的速度。将 DrawMode 属性设置为 fast，使快照的运行速度更快。

streamparticles 函数设置下列属性：

- Animate 属性设置为 10，快照 10 次。
- ParticleAlignment 属性设置为 on。
- MarkerFaceColor 属性设置为 red。
- Marker 属性设置为 0，绘圆形标记。也可以用其他直线标记。

输入下面的代码:

```
iverts = interpstreamspeed(x,y,z,u,v,w,verts,0.05);
set(gca,'drawmode','fast');
streamparticles(iverts,15,…
    'Animate',10,…
    'ParticleAlignment','on',…
    'MarkerEdgeColor','none',…
    'MarkerFaceColor','red',…
    'Marker','o');
```

生成的图形如图 6-66 所示。

6.12.5　带圆锥图的向量场

本例绘 wind 数据的速率向量场。生成该图采用了一系列技巧。

- 绘一个等表面，用于提供圆锥图的可视承接，并提供各种方法，为一系列圆锥选择指定的数据值。
- 进行光照，使等表面的形状清晰可见。

使用透视投影、相机位置设定和视角调整等方法确定最后的视角。

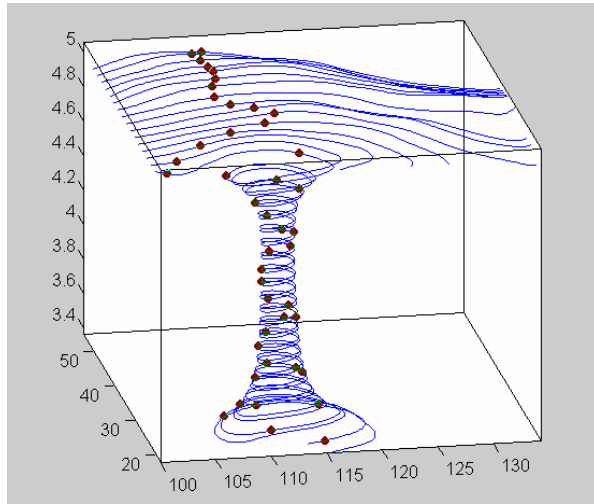


图 6-66　流动微粒快照

（1）创建等表面　为圆锥图在提供可视承接的矩形空间中显示等表面。创建等表面需要下面一些步骤：

- **计算向量场的大小，它代表气流速度；**
- **使用等表面和阴影来画一个演示矩形空间中气流流速与特定值相等的等表面。等表面内的矩形区域具有较高的流速，表面外的区域内流速较低；**
- **使用** isonormals **函数计算等表面的顶点范数。**
- **设置等表面的可视属性，设置其前景色为红色，无边界。**

```
load wind
wind_speed = sqrt(u.^2 + v.^2 + w.^2);
hiso = patch(isosurface(x,y,z,wind_speed,40));
isonormals(x,y,z,wind_speed,hiso)
set(hiso,'FaceColor','red','EdgeColor','none');
```

生成的图形如图 6-67 **所示。**

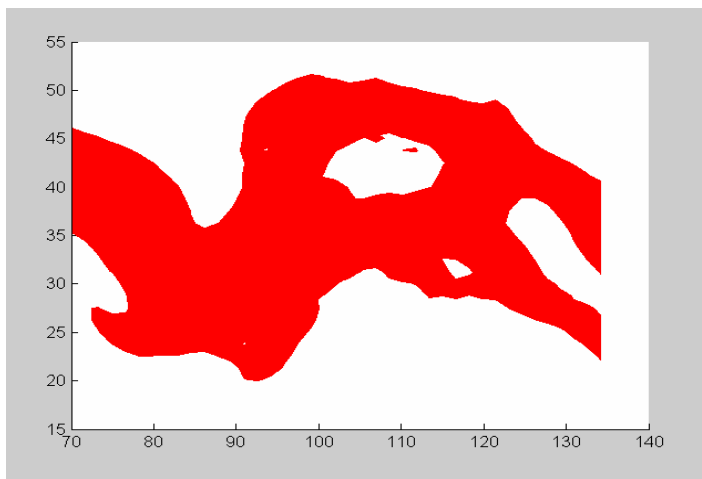


图 6-67　创建等表面

（2）添加等帽盖到等表面

```
hcap = patch(isocaps(x,y,z,wind_speed,40),…
    'FaceColor','interp',…
    'EdgeColor','none');
colormap hsv
```

生成的图形如图 6-68 所示。

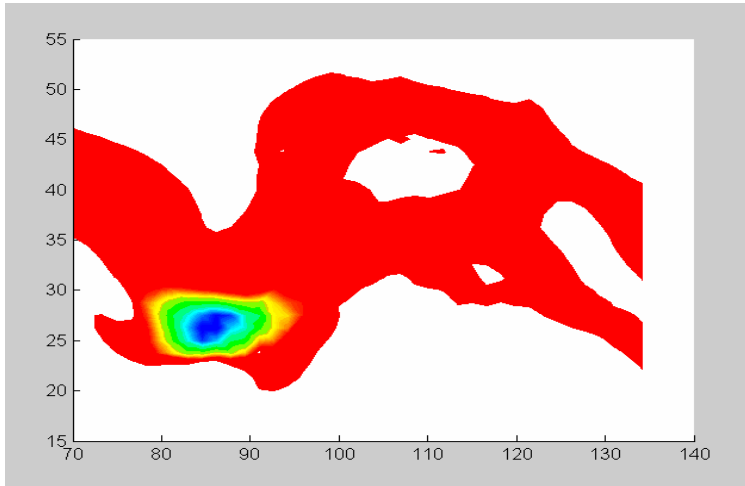


图 6-68　添加等帽盖

（3）创建第一个圆锥集　使用 daspect，在调用 coneplot 函数之前设置坐标轴的数据方向比，这样 MATLAB 可以确定圆锥的合适大小。

通过计算另外的具有更小相等值的等表面来确定放置圆锥的点（所以圆锥显示在第一个等表面之外），并用 reducepatch 函数减小侧面和顶点个数（这样，图上就不会因为有太多的锥体而显得拥挤）。

绘锥体并设置表面颜色为蓝色，不设置边缘颜色。

```
daspect([1,1,1]);
[f verts] = reducepatch(isosurface(x,y,z,wind_speed,30),0.07);
h1 = coneplot(x,y,z,u,v,w,verts(:,1),verts(:,2),verts(:,3),3);
set(h1,'FaceColor','blue','EdgeColor','none');
```

生成的图形如图 6-69 所示。

（4）创建第二个圆锥集　在扩展范围内的某值处创建第二个点集。绘第二个圆锥集合并设置其表面颜色为绿色，无边缘颜色。

```
xrange = linspace(min(x(:)),max(x(:)),10);
yrange = linspace(min(y(:)),max(y(:)),10);
zrange = 3:4:15;
[cx,cy,cz] = meshgrid(xrange,yrange,zrange);
h2 = coneplot(x,y,z,u,v,w,cx,cy,cz,2);
set(h2,'FaceColor','green','EdgeColor','none');
```

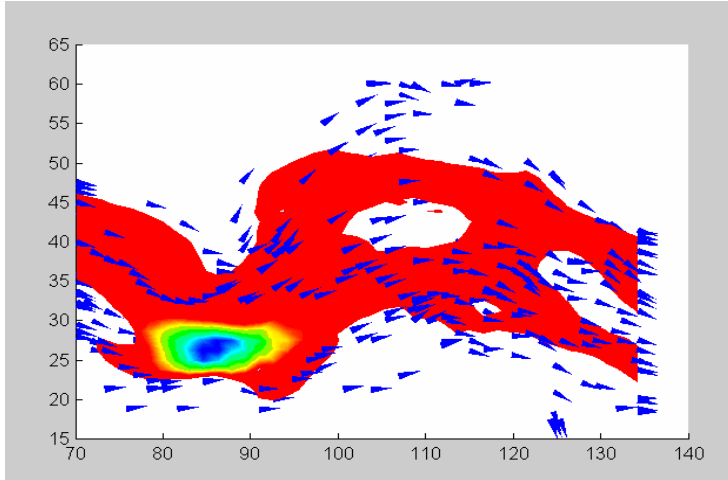


图 6-69　创建第一个圆锥集

生成的图形如图 6-70 所示。

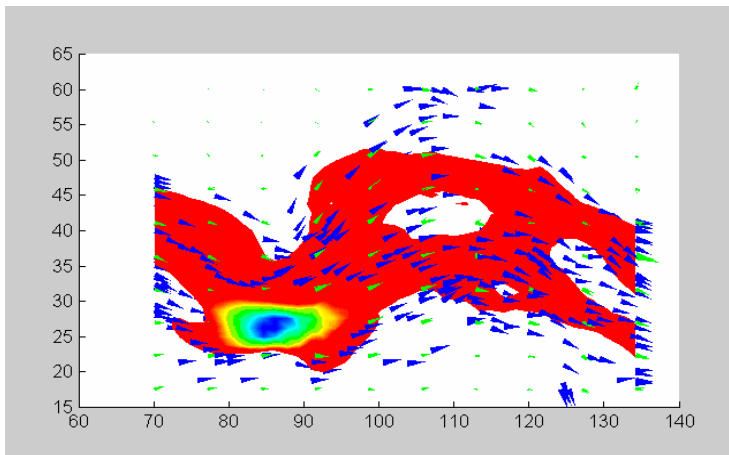


图 6-70　创建第二个圆锥集

（5）定义视角，设置投影类型　使用 axis 命令，令坐标轴范围等于最大值与最小值的差，并将图形封闭在盒形体积中，改善图形的整体性。设置投影类型为透视投影，使整个图形更自然。

设置视点并缩小图形，使视野更大。

```
axis tight
box on
camproj perspective
camzoom(1.25)
view(65,45)
```

生成的图形如图 6-71 所示。

（6）添加光照　添加光源并用 Phong 光照方法得到等表面的光照最平滑。增加 isocaps 上的背景光照长度，使它们更亮。

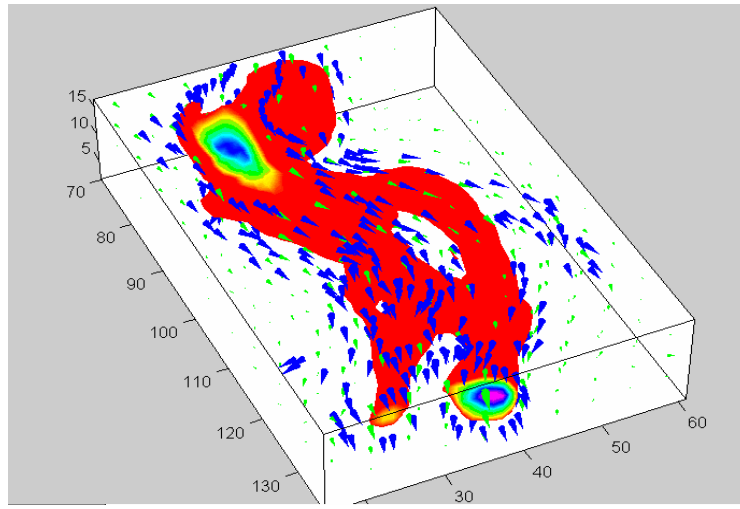


图 6-71　定义视角，设置投影类型

```
camlight(-45,45)
set(gcf,'Renderer','zbuffer');
lighting phong
set(hcap,'AmbientStrength',.6)
```

生成的图形如图 6-72 所示。

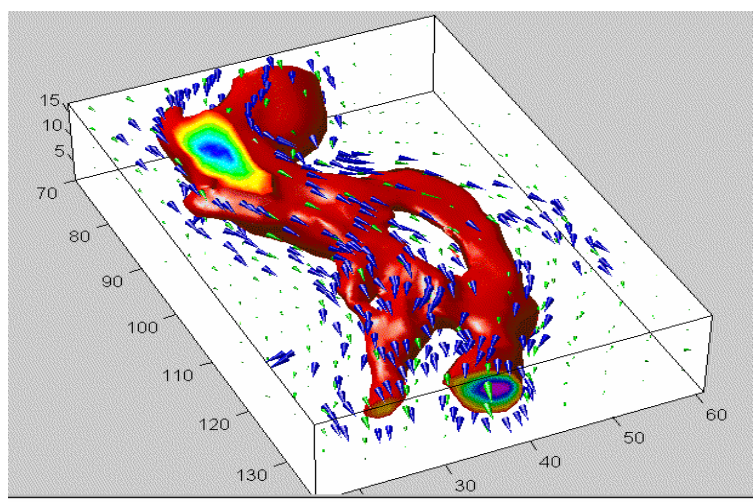


图 6-72　添加光照以后的效果

第 7 章　程序设计——M 文件

MATLAB 作为一种应用最广泛的科学计算软件，它不仅具有如前面几章所介绍的强大的数值计算、符号计算、画图功能，而且它还可以像 C，FORTRAN 等计算机高级语言一样，进行程序设计，编写一种以 m 作为文件扩展名的文件——M 文件。事实上，MATLAB 本身的许多函数就是 M 文件函数。MATLAB 提供的 M 文件编辑器与编译器，可以使用户方便地进行程序设计。下面，对编写 MATLAB 中的 M 文件进行介绍。

7.1　M 文件简介

所谓 M 文件，简单地说，就是用户把要实现的命令写在一个以 m 作为文件扩展名的文件中，然后由 MATLAB 系统进行解释，运行出结果。也就是说，M 文件实际上仅仅是一个命令集。正是由于 M 文件的如此特点，使得 MATLAB 具有强大的可开发性与可扩展性。事实上，MATLAB 中的许多函数本身就是由 M 文件扩展而成的。对 MATLAB 用户来说，他们也完全可以利用 M 文件来生成和扩充自己的函数库。另外，由于 MATLAB 是用 C 语言开发而成的，因此，M 文件的语法规则与 C 语言几乎完全一样。这对于 C 语言用户来说，就显得更容易了。

M 文件有两种格式，即函数式 M 文件与脚本式 M 文件。函数式 M 文件的第一句是以 function 语句作为引导的，脚本式 M 文件就是命令的简单叠加，它与批处理文件很相似。函数式 M 文件与脚本式 M 文件的相同之处在于：它们都是有以 m 作为扩展名的文本文件，它们都不进入命令窗口，而是由文本编辑器来创建外部文本文件。函数式 M 文件与脚本式 M 文件在通信方面是不一样的。函数式 M 文件与 MATLAB 命令空间之间的通信只通过传递给它的变量和它所创建的输出变量来进行，函数内的中间变量不出现在 MATLAB 的命令窗口内，也不与 MATLAB 的命令窗口交互操作。另外，脚本式 M 文件运行产生的所有变量都是全局变量，而函数式 M 文件中的所有变量除特别声明外，都是局部变量。在 MATLAB 中的 M 文件绝大多数是函数式 M 文件。

下面举一个 M 文件的例子。它是一个矩阵正交化的 M 文件 orth.m。

```
function Q = orth(A)
%ORTH     Orthogonalization.
%    Q = ORTH(A) is an orthonormal basis for the range of A.
%    That is, Q'*Q = I, the columns of Q span the same space as
%    the columns of A, and the number of columns of Q is the
%    rank of A.
%    See also SVD, RANK, NULL.
[U,S,V] = svd(A,0);
[m,n] = size(A);
if m > 1, s = diag(S);
```

```
    elseif m == 1, s = S(1);
    else s = 0;
end
tol = max(m,n) * max(s) * eps;
r = sum(s > tol);
Q = U(:,1:r);
```

结合上例，下面对 M 文件（主要针对函数式 M 文件）必须遵循的规则做简要介绍。

（1）函数名与文件名必须相同。例如，函数 orth 存储的文件名为 orth.m。

（2）脚本式 M 文件没有输入参数或输出参数，而函数式 M 文件有输入参数与输出参数。

（3）在 M 文件中，到第一个非注释行为止的注释行是帮助文本。当需要帮助时，返回该文本。例如，>> help orth 返回上述的注释行。

（4）函数可以有零个或多个输入变量，也可以有零个或多个输出变量。函数 nargin 与 nargout 包含输入与输出变量的个数。对函数进行调用时，可以按少于 M 文件中规定的输入与输出变量个数，但不能多于 M 文件中规定的输入与输出变量个数。当函数有一个以上的输出变量时，输出变量包含在括号内。

（5）函数式 M 文件中的所有变量除特别声明外，都是局部变量。它们在自己专有的工作空间中工作。如果说明变量是全局的，函数可以与其他函数、MATLAB 工作空间共享变量。但最好在实际编程时，少用或不用全局变量。全局变量通过关键字 global 声明。

（6）变量的命名可以包括字母、数字与下划线，但第一个必须是字母。M 文件设计中，字母是有大、小写区分的。所以，A 与 a 是不相同的。变量的长度不能超过系统函数 namelengthmax 所规定的值。

因为 M 文件一般是文本文件，所以可以使用一般的文本编辑器编辑 M 文件。为了能方便地编辑 M 文件，MATLAB 内部自带了 M 文件的编辑器与编译器。

为了进入 MATLAB 中的 M 文件的编辑器与编译器，可以选择 MATLAB 主菜单中的 New 子菜单，然后选择下一级子菜单中的 M-file 子菜单，就进入了 M 文件编辑器、编译器，如图 7-1 所示。

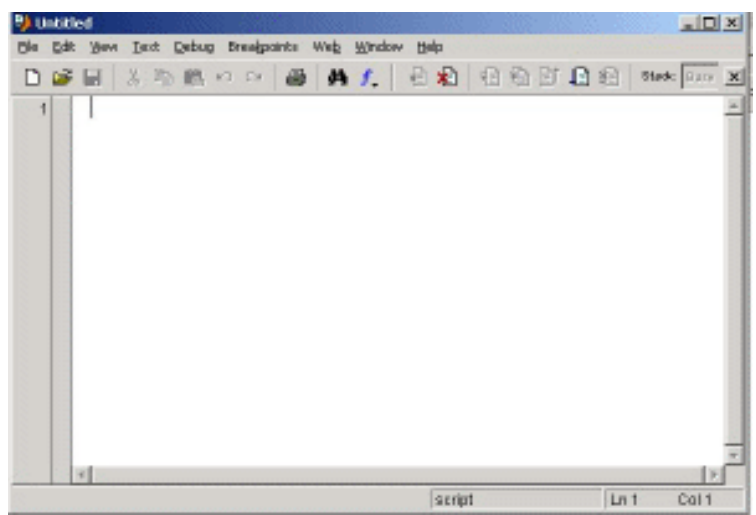


图 7-1　M 文件编辑器与编译器

然后，就可以在编辑窗口书写 M 文件了。同时，也可以像一般的程序设计语言一样，对 M 文件进行调试、运行。

7.2 M 文件的程序结构

程序结构一般分为顺序结构、循环结构、分支结构 3 种。理论上讲，只要有以上 3 种结构就可以构造功能强大的程序。与大多数计算机语言一样，MATLAB 也提供了上述 3 种程序结构。虽然 MATLAB 没有 C 语言那样具有丰富的控制结构，但 MATLAB 自身的强大功能弥补了这个不足，使用户在编程时几乎感觉不到困难。下面分别对 3 种程序结构进行介绍。

7.2.1 顺序结构

顺序结构就是依次顺序地执行程序的各条语句。顺序结构一般不含有其他子结构或控制语句，批处理文件就是典型的顺序结构的文件。下例就是一个典型的顺序结构程序。

```
a=1;
b=2;
c=4;
d=a+b;
f=c+d;
f
```

然后把上述语句存为 aa.m 文件。在 MATLAB 命令窗口中执行的结果如下：

```
aa
f =
     7
```

7.2.2 循环结构

在实际计算中，经常会碰到许多有规律的重复计算，此时就要对某些语句重复执行。一组被重复执行的语句称为循环体，每个循环语句都要有循环条件，以判断循环是否要继续进行下去。MATLAB 中的循环语句包括 For 循环与 While 循环。

1．For 循环

For 循环允许一组命令以固定的和预定的次数重复执行，For 循环的一般形式如下：

```
For v=expression（表达式）
        Statements;（执行语句）
End
```

在 for 与 end 语句之间的执行语句是按矩阵（或数组）中的每一列执行一次，即在每一次循环中，矩阵（或数组）元素一个一个地被赋给变量 v，然后由执行语句执行。例如，

```
for n=1:10
  x(n)=exp(n/5)+cos(n*pi/5);
end


x
```

```
x =
    2.0304
    1.8008
    1.5131
    1.4165
    1.7183
    2.5111
    3.7462
    5.2620
    6.8587
8.3891
```

用 For 循环语句需要注意以下事项。

（1）不能在 for 循环体内重新对循环变量 n 赋值来终止循环的执行。例如，

```
for n=1:10
    x(n)=exp(n/5)+cos(n*pi/5);
    n=10;
end

x
x =
    2.0304
    1.8008
    1.5131
    1.4165
    1.7183
    2.5111
    3.7462
    5.2620
    6.8587
    8.3891
```

（2）For 循环可以嵌套循环。

（3）循环语句内的“ ; ”可防止中间变量的输出。

2．While 循环

与 for 循环以固定的次数求一组命令的值相反，while 循环以不定的次数来求一组命令的值。While 循环的一般形式如下：

```
while expression（表达式）
    statements;（执行语句）
end
```

只要表达式 expression 中的元素为真，就执行 while 和 end 语句之间的命令。通常，表达式给出的是一个标量值，但数组（或矩阵）同样有效。若为数组（或矩阵），则要求所有的元素都必须为真。例如，

```
n=1;a=0;
while n<10
  a=a+log(n)/n;
  n=n+1;
end
a
a =
    2.4619
```

7.2.3 分支结构

在程序设计中，经常要根据一定的条件来执行不同的语句。当某些条件满足时，只执行其中的某个语句或某些语句。在这种情况下，分支结构就会是很好的选择了。在 MATLAB 中，分支结构语句包括 if-else-end 语句与 switch-case-otherwise 语句。

1．if-else-end 选择语句

最简单的 if-else-end 语句如下：

```
if expression
    statements;
end
```

如果表达式中的值为真，就执行 if 与 end 语句之间的命令串，否则跳过此命令串。如果有两个选择，if-else-end 语句如下：

```
if expression
    statements;
else
    statements;
end
```

如果有多个选择，if-else-end 语句如下：

```
if expression1
    statements1;
elseif expression2
    statements2;
elseif expression3
    statements3;
…
else
    statements;
end
```

例如，编写如下的 M 文件 fm.m：

```
function f=fm(n)
if n==0
   f=20;
elseif n==1
```

```
    f=40;
elseif n==2
    f=60;
else
    f=100;
end
```

然后，在 MATLAB **的命令窗口输入：**

```
fm(2)
ans =
      60
```

2．switch-case-otherwise **分支语句**

MATLAB **中的选择语句** switch-case-otherwise**，是特别让熟悉** C **等高级语言的用户方便地编写** M **文件而专门添加的。**switch-case-otherwise **语句的通用格式如下：**

```
switch switch-expression
case case-expression1,
    statements1;
case case-expresson2,
    statements2;
case case-expression3,
    statements3;
…
otherwise
    statements;
end
```

其中，switch-expression **给出了开关条件，当有** case-expression **与之匹配时，就执行其后的语句，如果没有** case-expression **与之匹配，就执行** otherwise **后面的语句。在执行过程中，只有一个** case **命令被执行。当执行完命令后，程序就跳出分支结构，执行** end **下面的语句。例如，**

```
function f=fm(n)
switch n
case 0
    f=20;
case 1
    f=40;
case 2
    f=60;
otherwise
    f=100;
end
```

然后，在 MATLAB **的命令窗口输入：**

```
fm(5)
```

```
ans =
     100
```

7.3 程序流控制

在许多程序设计语言中，经常要碰到提前终止循环、跳出子程序、显示出错信息等，此时就要用到控制程序流的命令。在 MATLAB 中，同样有这样的程序控制流命令。MATLAB 中的程序控制流命令有以下几个。

1．continue 命令

continue 命令经常与 for 或 while 语句一起使用，其作用是结束本次循环，即跳过循环体中下面尚未执行的语句，接着进行下一次是否执行循环的判断。

下面是 MATLAB 自带的 M 文件 magic.m 中运用 continue 语句的实例。

```
fid = fopen('magic.m','r');
count = 0;
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line) | strncmp(line,'%',1)
        continue
    end
    count = count + 1;
end
disp(sprintf('%d lines',count));
```

2．break 命令

break 命令也经常与 for 或 while 等语句一起使用，其作用是终止本次循环，跳出最内层的循环。使用 break 命令可以不必等到循环的自然结束，而是根据条件，退出循环。这在很多情况下是必须的。

下面为一个使用 break 语句的例子。

```
fid = fopen('fft.m','r');
s = '';
while ~feof(fid)
    line = fgetl(fid);
    if isempty(line)
        break
    end
    s = strvcat(s,line);
end
disp(s)
```

上述 break 命令当碰到第一条空行时，就退出 while 循环。

3．return 命令

return 命令能使当前正在运行的函数正常退出，并返回调用它的函数，继续运行。这个语句经常用于函数的末尾，以正常结束函数的运行。当然，也可用于函数的其他地方，对某

些条件进行判断，如果条件不符合要求，调用 return 语句终止当前运行，并返回调用它的函数或环境。实际上，MATLAB 中此语句的作用与在其他的程序设计语言中的作用相同。

4．echo 语句

通常，在 MATLAB 中执行 M 文件时，在命令窗口是看不到执行过程的。但在特殊情况下，比如需要对 M 文件演示时，要求 M 文件的每条命令都要显示出来，此时用 echo 命令就可以实现这样的操作。

对于脚本式 M 文件与函数式 M 文件，echo 命令略有不同。对于脚本式 M 文件，echo 命令可以用以下方式来实现：

```
echo on       %显示其后所有执行的命令文件的指令。
echo off      %不显示其后所有执行的命令文件的指令。
echo          %在上述两种情况下进行切换。
```

对于函数式 M 文件，echo 命令可以用以下方式来实现：

```
echo filename on       %使 filename 指定的 M 文件的执行指令显示出来。
echo filename off      %使 filename 指定的 M 文件的执行指令不显示出来。
echo on all       %其后的所有 M 文件的执行指令显示出来。
echo off all      %其后的所有 M 文件的执行指令不显示出来。
```

5．error 语句

在进行程序设计时，许多情况下有错误出现，此时如果能把出错信息显示出来，那样就会使用户了解到是什么原因引起错误，以采取适合的方式防止错误的再次发生。MATLAB 中的 error(出错信息)就用于实现上述功能。此命令能显示出错信息并终止当前函数的运行，将控制信息返回到键盘。

6．try … catch 语句

try…catch 语句的作用与 error 语句类似，是用于对异常情况进行处理的命令。把有可能引起异常的语句放在 try 控制块中，这样当 try 控制块中 statement 语句引起异常时，catch 控制块就可以捕获它，并针对不同的错误类型，进行不同的处理。它与 C++程序设计语言中的 try…catch 命令的作用一样。try…catch 命令的调用格式如下：

```
try,
    statement,
    …,
    statement,
catch,
    statement,
    …,
    statement,
end
```

7.4 M 文件举例

为了使读者对 M 文件编写有一个更深的了解，下面分别给出函数式 M 文件与脚本式 M 文件的实例。

```
function pp=bspline(t,window)
```

```
%BSPLINE Plots a B-spline and its polynomial pieces.
%
%     BSPLINE(T)    plots the B-spline with knot sequence T as well as the
%     polynomial pieces of which it is composed.
%
%     BSLINE(T,WINDOW) does the plotting in the WINDOW specified, then PAUSEs.
%
%     PP = BSPLINE(T) plots nothing, but returns the ppform of the specified
%     B-spline.
%
%     See also

%     Copyright (c) 1987-98 by C. de Boor and The MathWorks, Inc.
%     $Revision: 1.8 $

%     C. de Boor / latest change: May 22, 1989
%     C. de Boor / latest change: 11 May 1992 (correct misprint (missing blank))
%     C. de Boor / latest change: 21 June 1992 (replace zeros(xx))
%     cb: 14feb96 (v5 compatible); 07apr96 (no plotting if nargout>0)
%     cb: 07may96 (correct handling of hold on/off); 27sep96 (adjust colors)
%     cb: 05oct97 (replace use of ANS, to help compilation)
%     cb: 04may98 (standardize the help)
%    Generate the spline description
k=length(t)-1;
if (k>1)
     adds=ones(1,k-1);
     tt=[adds*t(1) t(:)' adds*t(k+1)];j=k+1;
     a=[adds*0 1 adds*0];

     %    From this, generate the pp description

     inter=find(diff(tt)>0); l=length(inter);
     tx=ones(l,1)*[2-k:k-1]+inter'*ones(1,2*(k-1));tx(:)=tt(tx);
     tx=tx-tt(inter)'*ones(1,2*(k-1));
     b=ones(l,1)*[1-k:0]+inter'*ones(1,k);b(:)=a(b);
     c=sprpp(tx,b);x=[tt(inter) tt(2*k)];
else
   l=1;x=t;c=1;
end

if nargout>0, pp=ppmak(x,c,1); return, end
```

```
%    Now generate a mesh…

step=100;xx=x(1)+[-10:step+10]*(x(l+1)-x(1))/step;nstep=length(xx);

%    …and generate and plot the polynomial pieces

if nargin>1, subplot(2,2,window), end
xxx=[xx(1) xx(nstep)]; yyy=[-1 2];
plot(xxx,yyy,'.b'),axis([xxx yyy]), grid off, hold on
bspl = spval(spmak(t,1),xx); plot(xx,bspl,'k','linew',2)
for j=1:(k+1), plot(t([j j]),yyy), end
temp = find(xx>=x(1));jh=temp(1);
jsmax=5;js=jsmax; co=['r';'g';'k';'m';'b'];
for j=1:l;js=js+1;if (js>jsmax), js=1;end
    jl=jh; temp = find(xx>=x(j+1));jh=temp(1);
    pval=polyval(c(j,:),xx-x(j));
    plot(xx,pval,co(js)); plot(xx(jl:jh),bspl(jl:jh),co(js),'linew',1.3)
end
pause
hold off
% if nargin>1 %, subplot
% else clg
% end
```

上例是画 B 样条曲线的函数式 M 文件，下面再看一例脚本式 M 文件。读者也可以对它们进行比较。

```
% An M-file script to produce          % Comment lines
% "flower petal" plots
theta = -pi:0.01:pi;                   % Computations
rho(1,:) = 2*sin(5*theta).^2;
rho(2,:) = cos(10*theta).^3;
rho(3,:) = sin(theta).^2;
rho(4,:) = 5*cos(3.5*theta).^3;
for i = 1:4
    polar(theta,rho(i,:))              % Graphics output
    pause
end
```

上例是在区间[-π　π]画出几个三角函数图形的脚本式 M 文件。

第 8 章　图形用户界面（GUI）设计

随着计算机技术的飞速发展，人与计算机之间的通信方式也发生了深刻变化。从传统的命令行通信方式（如 DOS 系统）演变成了图形界面下的交互通信方式（如 WINDOWS 系统）。在图形用户界面（GUI）下，用户可以通过鼠标等输入设备与计算机进行信息的交流，选择欲运行的计算机程序，并控制程序的运行。现在，绝大部分应用程序（如 Microsoft word）都是在图形用户界面（GUI）下运行的，并且，绝大部分的程序设计工具（如 Visual Basic，Visual C++）都可以进行图形用户界面（GUI）的设计与开发工作。图形用户界面（GUI）是包括窗口、图标、菜单、工具条等对象的用户界面。用户可以用鼠标等点击设备去选择或激活这些对象，以引起动作或发生变化。

作为强大的科学计算软件，MATLAB 也提供了图形用户界面的设计与开发功能。MATLAB 中的基本图形用户界面对象分为 3 类：用户界面控件对象（uicontrol）、下拉式菜单对象（uimenu）和内容式菜单对象（uicontextmenu）。其中，uicontrol 对象能建立按钮、列表框、编辑框等图形用户界面对象，uimenu 能建立下拉式菜单和子菜单等图形用户界面对象，uicontextmenu 能建立内容式菜单用户界面对象（类似于 Visual C++等程序设计软件中的弹出式菜单）。利用上述对象，进行周密的组织、设计，就可以设计出一个界面良好、操作简便、功能强大的图形用户界面。

本章首先对与图形用户界面设计有关的图形对象句柄进行介绍，然后，介绍 GUI 的设计工具，包括对象设计编辑器、菜单编辑器等。在此基础上，对 uicontrol，uimenu，uicontextmenu 等图形用户界面对象的建立、属性等进行详细地介绍。并且，讨论用鼠标等输入设备触发上述对象后产生的动作程序的编写问题。最后，提供几个进行图形用户界面设计的实例。

8.1　图形对象及其句柄

每个图形对象都有一个句柄，只有获取了图形对象的句柄，才可以对该图形对象进行控制、设置或修改对象的有关属性。在进行图形用户界面（GUI）设计时，应该先理解“句柄图形”，它是设计与实现 GUI 的前提。

那么，什么是句柄图形？简单地讲，句柄图形是对底层图形对象集合的总称，实际上它进行了生成图形的工作。通过句柄图形，可以定制图形的许多特性。比如，用户可以通过图形对象 uicontrol 的句柄，设置控件的背景颜色。

8.1.1　图形对象

对象是现代计算机行业最常用的术语之一，面向对象的程序设计，数据库对象，Web 对象设计，操作系统和应用程序接口等都有对象的概念。一个对象可以被定义为由一组紧密相关、形成惟一整体的数据结构或函数的集合。在 MATLAB 中，一幅图中可以有多个图形对象，每个图形对象可以被单独地操作。

在 MATLAB 中，由图形命令产生的每一个对象都是图形对象。图形对象不仅包括 uimenu，

uicontextmenu 和 uicontrol 对象，而且还包括图形、坐标轴、线条、曲面、文本和它们的子对象。计算机屏幕是根对象，它是所有其他对象的父对象；图形窗口是计算机屏幕的子对象；坐标轴，uimenu，uicontextmenu 和 uicontrol 是图形窗口的子对象；图像、线条、曲面、文本、矩形等是坐标轴的子对象。MATLAB 中的 GUI 对象层次结构如图 8-1 所示。

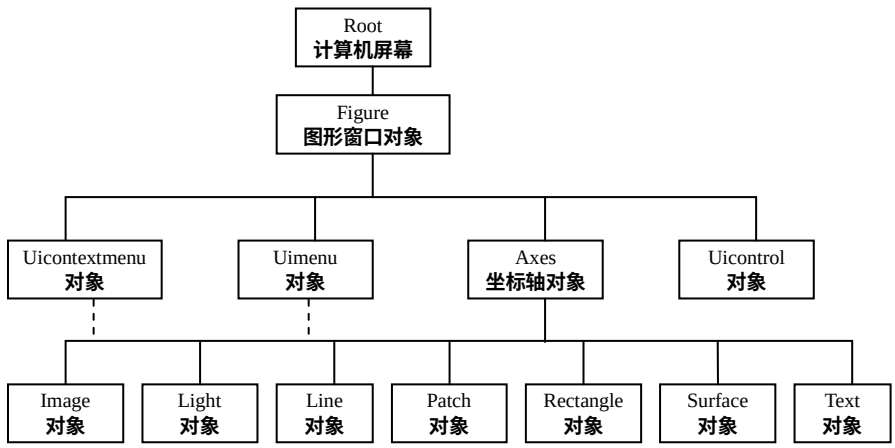


图 8-1　GUI 对象层次结构图

根对象可包含一个或多个图形窗口对象；每个图形窗口也可包含一个或多个 Uimenu 对象、Uicontextmenu 对象、Uicontrol 对象或坐标轴（Axes）对象；每个 Uimenu，Uicontextmenu 对象可包含一个或多个 Uimenu，Uicontextmenu 子对象；Uicontrol 对象虽无子对象结点，但它也有多种类型，例如按钮框、检验框、文本框、编辑框等；所有其他的对象都是坐标轴的子对象，并且在坐标轴上显示。

当利用 MATLAB 创建对象的函数来创建子对象时，如果父对象不存在，函数会自动创建子对象的父对象。例如，创建一个 Line 子对象，如果没有坐标轴父对象，则在创建 Line 子对象的同时，会创建坐标轴父对象。

8.1.2　图形对象句柄

在 MATLAB 中，每个图形对象都由一个数字来标识，叫做句柄（Handle）。每创建一个对象，就为它建立一个惟一的句柄，用来惟一地确定该对象。计算机屏幕作为根对象，其对象句柄通常为零；图形窗口的句柄为整数；其他的图形对象句柄是浮点值。

MATLAB 可以获得图形，坐标轴，Uimenu，Uicontextmenu，Uicontrol 和其他对象的句柄。具体地讲，可以通过下述 3 个函数来获得对象句柄值。

- gcf　　获得当前图形窗口的句柄值。
- gca　　获得当前图形窗口内当前坐标轴的句柄值。
- gco　　获得当前图形窗口内当前对象的句柄值。

获得对象的句柄值后，就可以利用该对象句柄设置或修改对象的属性。在 MATLAB 中，可以通过函数 get 来获得图形对象的属性，通过函数 set 来设置或修改图形对象的属性。例如：

```
>>get(Hl_a, 'color')
```

返回句柄 Hl_a 所属对象的颜色值。

```
>>set(Hl_a, 'color', 'r')
```

将句柄 Hl_a 所属对象的颜色值设置为红色。

也可以通过函数 delete 来删除句柄所属的图形对象。例如：

```
>>delete(Hl_a)
```

删除句柄 Hl_a 所属的图形对象。

可以通过对象生成函数来生成对象。所有生成对象的 MATLAB 函数都为所生成的对象返回一个句柄值。这些函数包括 figure，uicontrol，uimenu，uicontextsmenu，text，image，plot，mesh，surf 等。

8.2 GUI 设计模板及设计工具

对 MATLAB 中的 Figure 对象、Uimenu 对象、Uicontextmenu 对象、Uicontrol 对象或坐标轴（Axes）对象及其子对象的生成以及属性值的设置、修改，可以在 MATLAB 命令窗口内进行。例如，

```
x=-2*pi:pi/80:2*pi;
y=sin(x);
Hl_sin=plot(x,y);
set(Hl_sin,'Color',[1 0.5 0],'LineWidth',3)
```

上例利用 plot 命令画出正弦曲线，并对所画的曲线设置它的 RGB 颜色为[1 0.5 0]，曲线宽度为 3 个单位线宽。画出的图形如 8-2 所示。

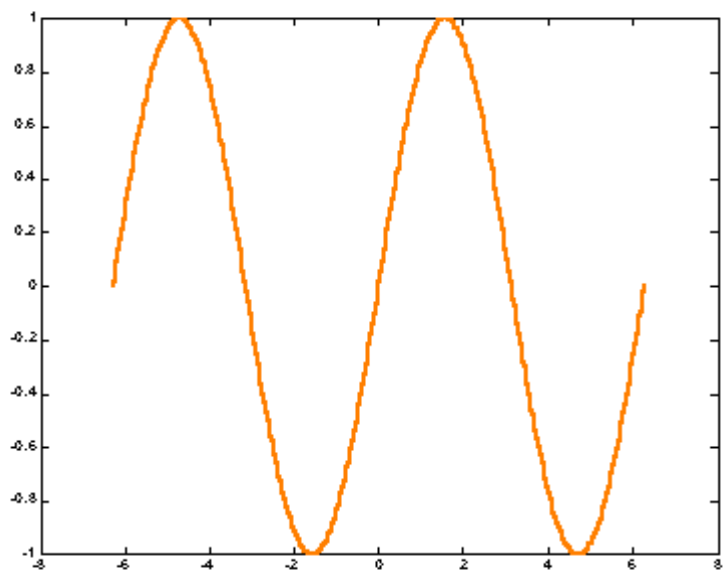


图 8-2　正弦曲线图

由于在画曲线时，还没有图形窗口对象，所以先创建了图形窗口父对象。函数 plot 返回了图形窗口对象的句柄值。set 函数正是借助于该句柄值来设置曲线的颜色与线宽属性。

与 Visual C++等程序设计软件一样，为了使用户更方便地进行 GUI 的程序设计与开发，

MATLAB 也提供了进行 GUI 设计与开发的模板。

同时，为了能够像 Visual Basic 等程序设计工具一样简单、方便地进行 GUI 的设计与开发工作，MATLAB 提供了一套方便、实用的 GUI 设计工具。MATLAB 中的 GUI 设计工具包括以下几个：

（1）对象设计编辑器（Layout Editor）——在图形窗口内创建、安排各种对象。

（2）菜单编辑器（Menu Editor）——创建、设置、修改下拉式菜单和内容式菜单。

（3）对象属性查看器（Property Inspector）——可查看每个对象的属性值，也可修改、设置对象的属性值。

（4）位置调整工具（Alignment Tool）——可利用该工具左右、上下对多个对象的位置进行调整。

（5）对象浏览器（Object Browser）——可观察当前设计阶段的各个句柄图形对象。

（6）Tab 顺序编辑器（Tab Order Editor）——通过该工具，设置当用户按下键盘上的 Tab 键时，对象被选中的先后顺序。

8.2.1 GUI 设计模板

在 MATLAB 命令窗口内，选择 File 主菜单的 New 子菜单，会看到一个 GUI 的二级子菜单，单击它，就会显示 GUI 的程序设计模板，如图 8-3 所示。

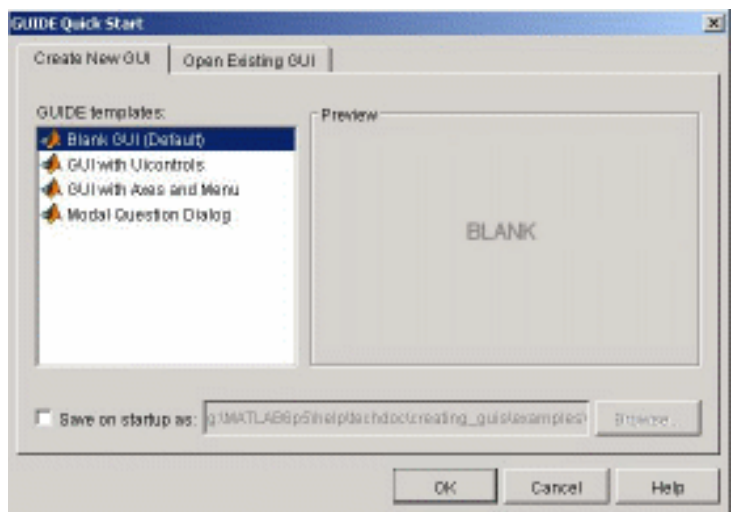


图 8-3　GUI 设计模板界面

从图 8-3 中可知，MATLAB 为 GUI 设计一共准备了 4 种模板，分别是 Blank GUI（默认）、GUI with Uicontrols（带控件对象的 GUI 模板）、GUI with Axes and Menu（带坐标轴与菜单的 GUI 模板）与 Modal Question Dialog（带模式问话对话框的 GUI 模板）。

当用户选择不同的模板时，在 GUI 设计模板界面的右边就会显示出与该模板对应的 GUI 图形。比如，选择 GUI with Axes and Menu 模板后的 GUI 设计模板界面如图 8-4 所示。

选择不同的 GUI 设计模板，在对象设计编辑器中显示的结果是不一样的。比如，选择 GUI with Uicontrols 模板后，在对象设计编辑器中的显示结果如图 8-5 所示。

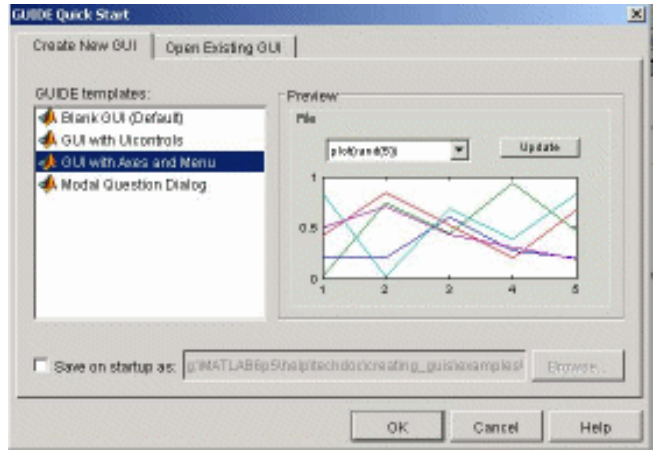


图 8-4　GUI 设计的 GUI with Axes and Menu 模板界面

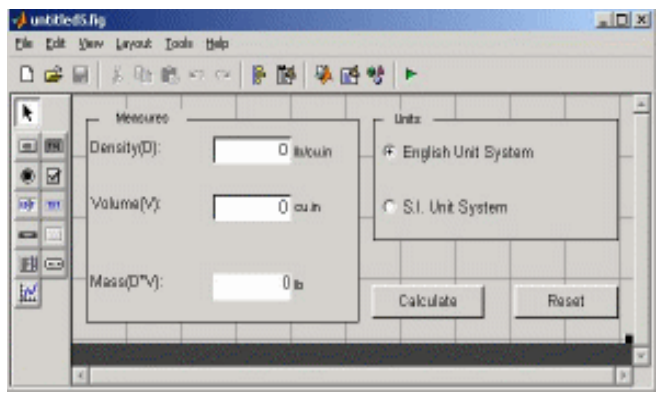


图 8-5　对象设计编辑器中显示的 GUI with Uicontrols 模板界面

8.2.2　对象设计编辑器

在 GUI 设计模板界面中选中一个模板，然后单击 OK 后，就会显示对象设计编辑器（Layout Editor）。图 8-6 所示为选择 Blank GUI 模板后显示的对象设计编辑器界面。

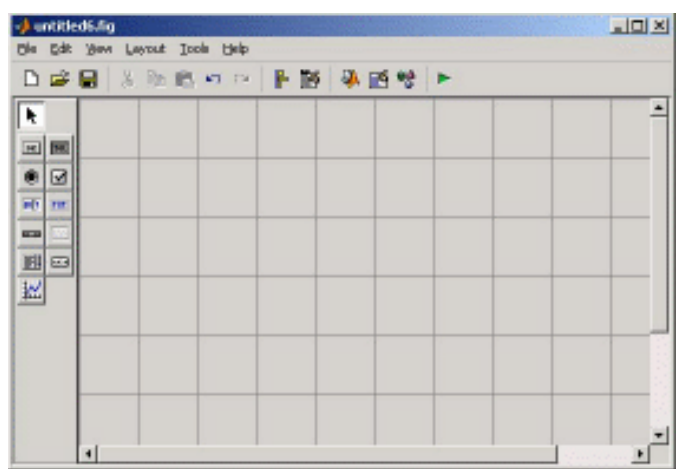


图 8-6　对象设计编辑器中的 Blank GUI 模板界面

图 8-6 中的左边图标是 Push Button，Toggle Button，Radio Button，CheckBox，Edit Text，Static Text，Slider，Frame，ListBox，Popup Menu 等控件对象和 Axes 坐标轴对象。从中选择一个对象，以拖曳方式在对象设计区（Layout Area）生成该对象，其对象创建方式与 Visual Basic 中进行 GUI 设计时创建对象一样方便、简单。创建图形对象后，通过双击该对象，就会显示该对象的属性编辑器（Property Inspector）。例如，创建一个 Push Button 对象，并设置该对象的属性值，如图 8-7 所示。

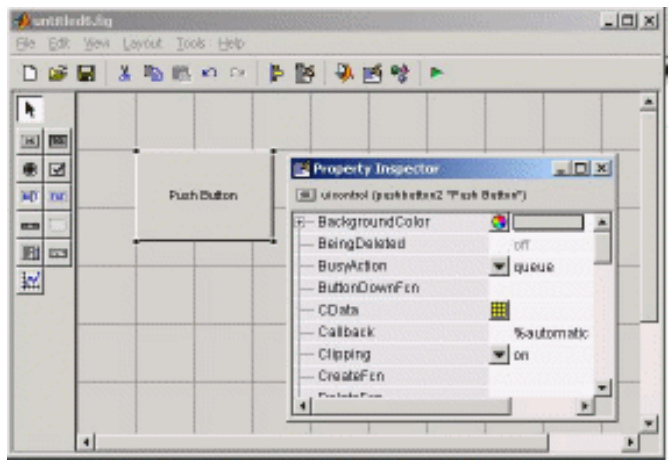


图 8-7　对象属性编辑器界面

在对象设计编辑器界面的工具条上，有菜单编辑器、位置调整器、属性编辑器、M 文件编辑器、对象浏览器的按钮以及激活创建的图形窗口（Activate Figure）的按钮 Run，通过它们可以方便地调用需要使用的 GUI 设计工具。而且，在菜单条上也有上述各个 GUI 设计工具的菜单，也可以通过菜单条方便地调用需要使用的 GUI 设计工具。另外，单击 Run 按钮，可以即时查看设计的图形用户界面的设计结果。

在选中图形对象的前提下，按鼠标右键，显示出一个弹出式菜单，用户可以从中选择某个子菜单项进行相应的设计。例如，选中已经创建的 Push Button 对象，按鼠标右键，将显示如图 8-8 所示的界面。

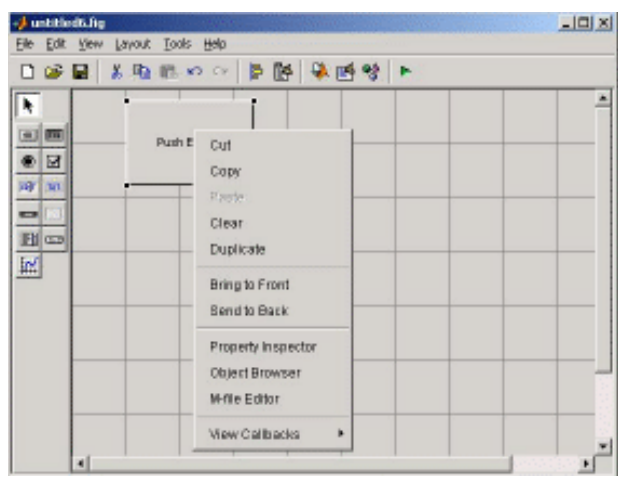


图 8-8　按鼠标右键显示的图形对象界面

特别注意，在图 8-8 中，View CallBacks 子菜单的 Edit CallBack，Edit ButtondownFcn，Edit CreateFcn，Edit DeleteFcn 等几个选项是在发生鼠标按下等事件时要发生的动作。通过单击这些选项，可以编写事件发生时需要执行的源代码。

在对象设计区（Layout Area）右击鼠标，会显示与编辑、设计整个图形窗口有关的弹出式子菜单。

8.2.3 菜单编辑器

利用菜单编辑器，可以创建、设置、修改下拉式菜单和内容式菜单。从对象设计编辑器界面的工具条上选择，或者选择 Layout 菜单下的 Menu Editor 子菜单，就可以看到菜单编辑器（Menu Editor）的界面，如图 8-9 所示。

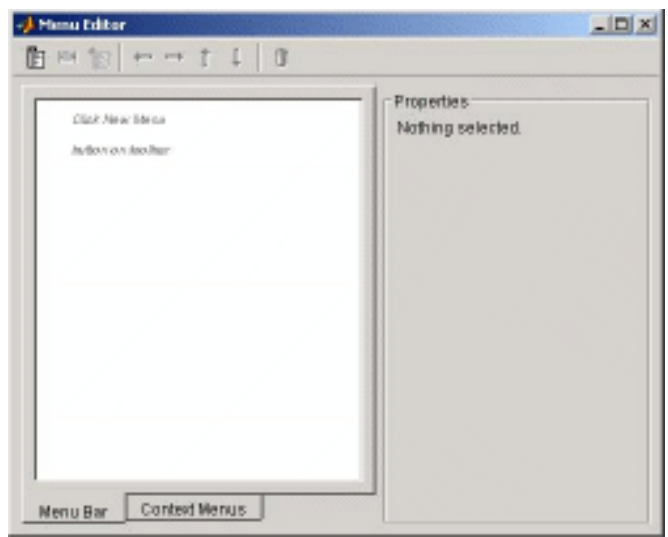


图 8-9　菜单编辑器界面

图 8-9 中左上角第一个按钮用于创建下拉式菜单。用户可以通过点击它，来创建下拉式主菜单。第二个按钮用于创建下拉式主菜单的子菜单，在选中已经创建的下拉式主菜单后，可以点击这个按钮来创建选中的下拉式主菜单的子菜单。选中创建的某个下拉式菜单后，菜单编辑器的右边就会显示该菜单的有关属性，用户可以在这里设置、修改菜单的属性。如图 8-10 所示，利用菜单编辑器创建了 File 与 Contact 两个主菜单，并且在 File 主菜单下，创建了 Open，Save，Save As…3 个子菜单，在 Contact 主菜单下创建了 Create New…子菜单。

菜单编辑器界面的左下角有两个按钮，选择第一个按钮，可以创建下拉式菜单，图 8-10 创建的 File 与 Contact 主菜单就是在它下面创建的。选择第二个按钮，可以创建 Context Menu 菜单（类似于 Visual C++等程序开发软件中创建的弹出式菜单）。选择它后，图中左上角的第三个按钮就会变成可用，点击它可以创建 Context Menu 主菜单。在选中已经创建的 Context Menu 主菜单后，可以点击第二个按钮创建选中的 Context Menu 主菜单的子菜单。与下拉式菜单一样，选中创建的某个 Context Menu 菜单，菜单编辑器的右边就会显示该菜单的有关属性，可以在这里设置、修改菜单的属性。如图 8-11 所示，利用菜单编辑器创建了 Context_Menu 主菜单，然后在该主菜单下又创建了 New 和 Save 两个子菜单。

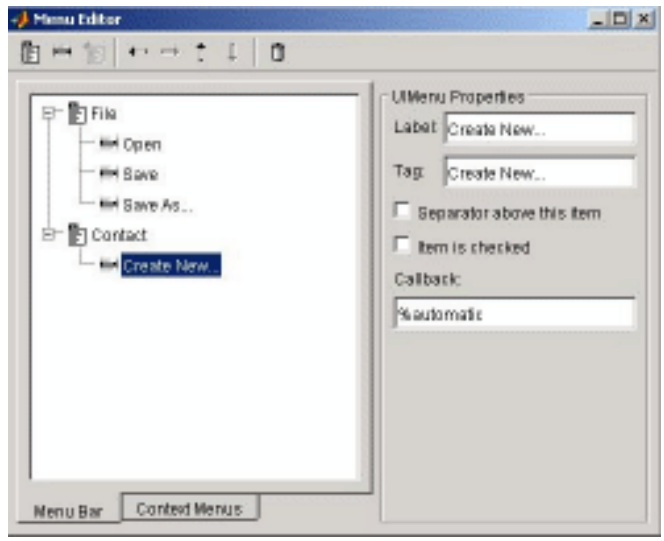


图 8-10　创建了 File 与 Contact 主菜单的菜单编辑器界面

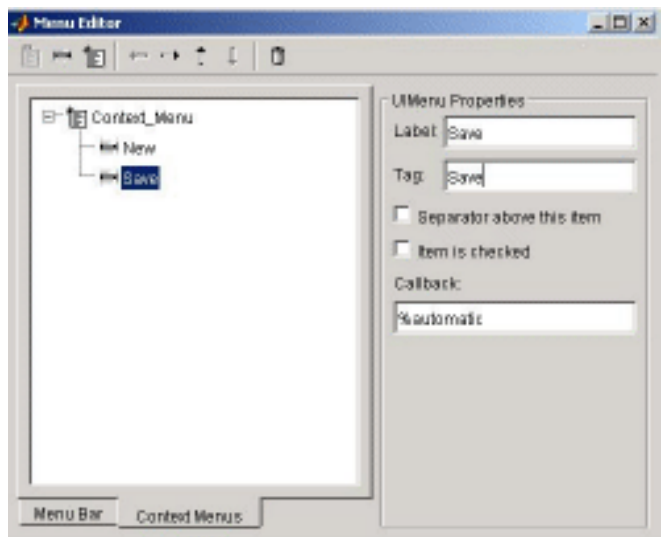


图 8-11　创建 Context Menu 菜单的菜单编辑器界面

点击菜单编辑器最右边的按钮，可以删除选中的菜单。另外，菜单编辑器左上角的第四个与第五个按钮用于对选中的菜单进行左移与右移；第六与第七个按钮用于对选中的菜单进行上移与下移。

8.2.4　对象属性查看器

利用对象属性查看器，可以查看每个对象的属性值，也可以修改、设置对象的属性值，从对象设计编辑器界面工具条上选择，或者选择 Edit 菜单或 Tools 菜单下的 Inspect Properties 子菜单，就可以看到对象属性查看器（Property Inspector）的界面。另外，在 MATLAB 命令窗口的命令行上输入 inspect，也可以看到对象属性查看器。如图 8-12 所示。

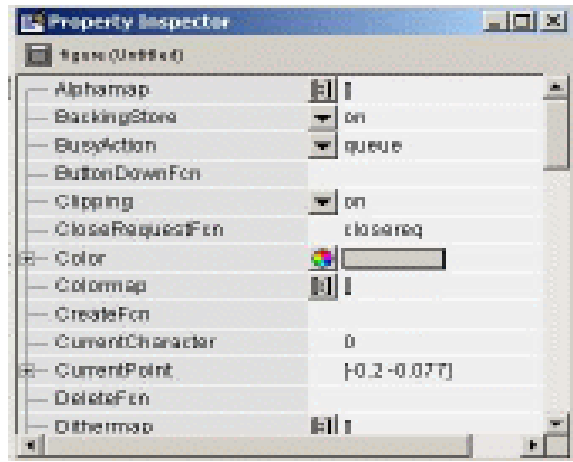


图 8-12　对象属性查看器的界面

在选中某个对象后，可以通过对象属性查看器，查看该对象的属性值，也可以方便地修改对象属性的属性值。

8.2.5　位置调整工具

利用位置调整工具，可以对对象设计编辑器中对象设计区(Layout Area)内的多个对象的位置方便地进行调整。从对象设计编辑器界面的工具条上选择，或者选择 Layout 菜单下的 Align Objects…子菜单，就可以看到对象位置调整器（Alignment Tool）的界面，如图 8-13 所示。

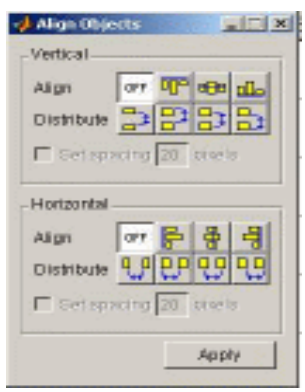


图 8-13　对象位置调整器的界面

图 8-13 中的第一栏是垂直方向的位置调整。其中，Align 表示对象间垂直对齐，Distribute 表示对象间的垂直距离。在选中 Distribute 中的某个按钮后，Set Spacing 就变成可用。然后，可以通过它来设置对象间的距离。注意，距离的单位是像素（Pixels）。

图 8-13 中的第二栏是水平方向的位置调整。与垂直方向的位置调整一样，Align 表示对象间水平对齐，Distribute 表示对象间的水平距离。在选中 Distribute 中的某个按钮后，Set Spacing 就成为可用。然后，可以通过它来设置对象间的距离。注意，距离的单位是像素（Pixels）。

在选中多个对象后，可以方便地通过对象位置调整器调整对象间的对齐方式、距离。

8.2.6　对象浏览器

利用对象浏览器，可查看当前设计阶段的各个句柄图形对象。从对象设计编辑器界面的工具条上选择，或者选择 Tool 菜单下的 Object Browser 子菜单，就可以看到对象浏览器（Object Browser）的界面。例如，在 Layout Area 内创建了 3 个对象，它们分别是 Edit Text，Push Button，ListBox 对象，如图 8-14 所示。

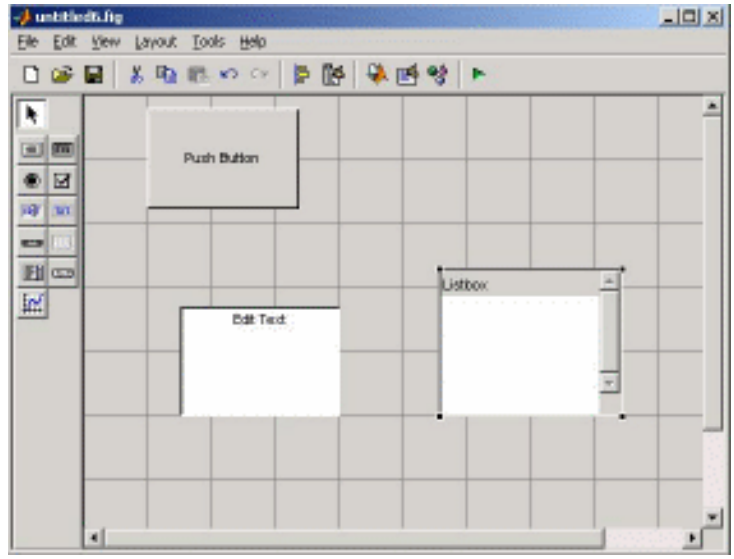


图 8-14　创建了 3 个对象的对象设计编辑器界面

此时单击对象浏览器按钮，可以看到对象浏览器的界面，如图 8-15 所示。

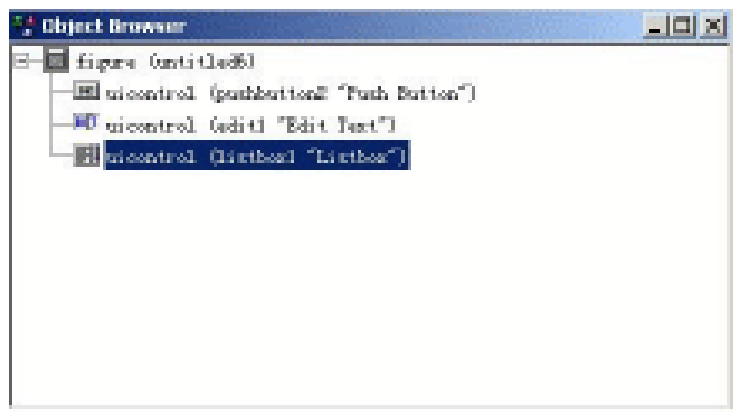


图 8-15　对象浏览器界面

在对象浏览器中，可以看到已经创建的 3 个对象以及图形窗口对象 Figure。用鼠标双击图中的任何一个对象，可以看到对象的属性查看器界面。

8.2.7　Tab 顺序编辑器

利用 Tab 顺序编辑器（Tab Order Editor），可以设置用户按键盘上的 Tab 键时，对象被选中的先后顺序。选择 Tools 菜单下的 Tab Order Editor…子菜单，就可以看到 Tab 顺序编辑器的界面。例如，图 8-14 中创建的 3 个对象，与它们相对应的 Tab 顺序编辑器的界面如图 8-16 所示。

在 Tab 顺序编辑器中，可以看到已经创建的 3 个对象。界面的左上角有两个按钮，分别用于设置对象按 Tab 键时选中的先后顺序。

利用上面介绍的 GUI 设计工具，可以设计出界面友好、操作简便、功能强大的图形用户界面。然后，再通过编写触发对象后产生的动作执行程序，就可以完成相应的任务。

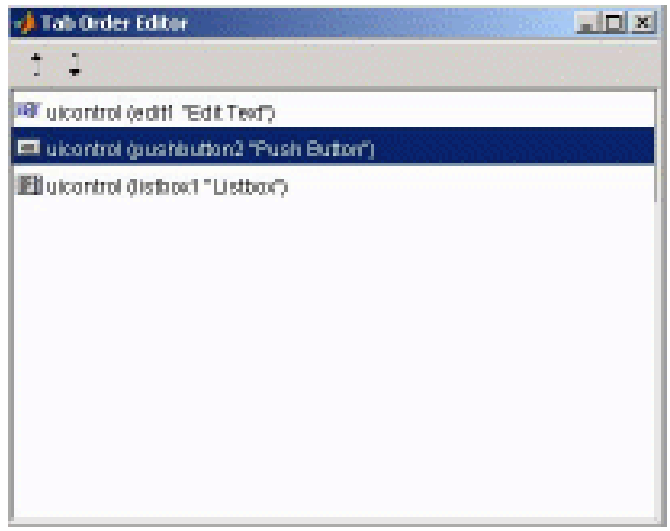


图 8-16　Tab 顺序编辑器界面

8.3　菜单

在绝大多数的图形用户界面下，都包含有菜单。通过选择各级菜单，可以执行相应的命令，实现相应的功能。一般地讲，从菜单的标题或名字可以大概了解该菜单的功能。在 WINDOWS 系统中，菜单一般位于图形用户界面的顶端。例如，在 WINDOWS 系统的 MATLAB 的主窗口中，就有一个主菜单栏，包括 File，Edit，View，Web，Windows，Help 等主菜单。在各级主菜单下，还有相应的子菜单。通过选择某个主菜单，就会显示该主菜单的下拉式子菜单。在子菜单下，还可以嵌套子菜单。此时，在该级子菜单旁一般有箭头出现，选择它又会显示下一级子菜单。在 MATLAB 图形用户界面（GUI）设计中，有两种菜单类型，分别是下拉式菜单类型 Uimenu 和内容式菜单类型 Uicontextmenu。下面对 MATLAB 中的菜单进行介绍。

8.3.1　菜单建立

在 MATLAB 6.0 中，可以通过命令行方式和 GUI 设计工具中的菜单编辑器 Menu Editor 两种方式来建立菜单。

1．命令行方式

在命令行方式下，可以通过函数 uimenu 来建立下拉式菜单对象。uimenu 函数有如下几种调用格式：

```
uimenu('PropertyName',PropertyValue,…)
uimenu(hf_g,'PropertyName',PropertyValue,…)
handle = uimenu('PropertyName',PropertyValue,…)
handle = uimenu(hf_g,'PropertyName',PropertyValue,…)
```

其中，handle 创建菜单项句柄值；hf_g 是菜单所在的图形窗口的句柄值或者子菜单所属的主菜单的句柄值；PropertyName 是菜单的某个属性的属性名；PropertyValue 是与菜单属性名相对应的属性值。其中，第一与第二种调用格式，不返回创建的菜单的句柄值。第一与第

三种调用格式，省略菜单所在的图形窗口的句柄值或者子菜单所属的主菜单的句柄值。通过函数 uimenu 可以设置菜单的多个属性的属性值。

使用函数 uimenu 可以创建主菜单与下拉式子菜单。当函数中的变量 hf_g 是菜单所在的图形窗口的句柄值时，创建的是主菜单；当 hf_g 是某个主菜单的句柄值时，创建的是该主菜单下的下拉式子菜单。例如：

```
hm=uimenu(gcf,'Label','File');
hm1=uimenu(hm,'Label','New');
hm2=uimenu(hm,'Label','Open');
hm3= uimenu(hm,'Label','Save', 'Separator','On');
```

上述命令，将在当前图形窗口中创建一个 File 主菜单，并在此主菜单下，创建 New，Open 与 Save 3 个子菜单。子菜单 Save 与 New，Open 间用分隔条分开。创建的菜单位于图形窗口 Help 菜单标题的后边。

在命令行方式下，可以通过函数 uicontextmenu 来创建内容式菜单对象。uicontextmenu 的调用形式如下：

```
Hm_contextmenu=uicontextmenu( 'PropertyName', 'PropertyValue',…)；
```

其中，Hm_contextmenu 是创建的菜单项的句柄值；PropertyName 是菜单的某个属性的属性名；PropertyValue 是与菜单属性名相对应的属性值。利用 uicontextmenu 函数生成内容式菜单后，再通过函数 uimenu 可以往创建的内容式菜单中添加子菜单。然后，可以通过函数 set，把创建的内容式菜单与某个对象相联系，通过设置对象的 UiContextMenu 属性，使内容式菜单依附于该对象。需要说明的是，内容式菜单必须依附于某个对象而存在。例如：

```
cmenu = uicontextmenu;   %定义内容式菜单。
hline = plot(1:10, 'UIContextMenu', cmenu);   %画直线，并把内容式菜单与直线联系起来。
% 定义内容式菜单子菜单项的 callback 属性值。
cb1 = ['set(hline, "LineStyle", "--")'];
cb2 = ['set(hline, "LineStyle", ":")'];
cb3 = ['set(hline, "LineStyle", "-")'];
% 定义内容式菜单的子菜单项。
item1 = uimenu(cmenu, 'Label', 'dashed', 'Callback', cb1);
item2 = uimenu(cmenu, 'Label', 'dotted', 'Callback', cb2);
item3 = uimenu(cmenu, 'Label', 'solid', 'Callback', cb3);
```

上述命令，将在当前图形窗口内画一条直线，并且当在直线附近单击鼠标右键时，就会显示创建的内容式菜单。内容式菜单中包括 dashed，dotted，solid 等 3 个子菜单项，当单击某一个子菜单项时，所画直线的线型就会发生变化，如图 8-17 所示。

2．GUI 设计工具菜单编辑器

在命令行方式下创建菜单时，需要记住用于创建菜单的函数，如果忘记了创建菜单的函数，那么就无法创建菜单。然而，利用 GUI 设计工具中的菜单编辑器，就可以方便地创建下拉式菜单与内容式菜单。具体创建方法，请参阅第二节的菜单编辑器（Menu Editor）。图 8-18 所示是用菜单编辑器创建的下拉式菜单。

图 8-18 所示的菜单中，有一个 File 主菜单，在该主菜单下有 New，Open 和 Save 3 个子菜单。其中，子菜单 Save 与 New，Open 间用分隔条分开。

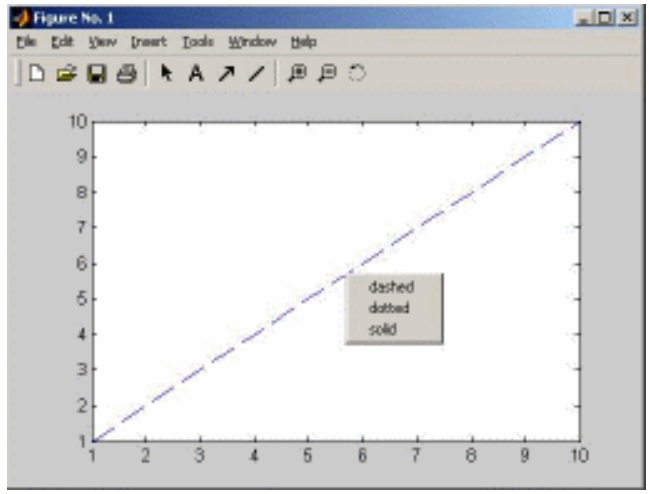


图 8-17　创建的内容式菜单

图 8-18　用菜单编辑器创建的下拉式菜单

8.3.2　菜单属性

利用函数 uimenu 建立菜单时，可以定义菜单属性的属性值；利用函数 set，可以设置、改变属性的属性值；利用函数 get，可以获得菜单属性的属性值，也可以通过 Property Inspector GUI 设计工具来设置、改变菜单的属性值。

1．uimenu 菜单的属性

（1）外观及风格控制属性

- Checked 属性　　Checked 属性的取值可以是 on 或 off。其中，off 是默认值。通过该属性，可以对选中的菜单项作一个核选标记“√”，也可以通过该属性删除核选标记。该属性对顶层的菜单项没有影响。
- Label 属性　　Label 属性的取值是一个字符串。该属性用于设置菜单的标题名称。可以在设置 Label 属性值的字符串中使用字符“&”，以此来设置菜单的快键方式。当在 Label 属性值中设置了“&”字符后，在“&”字符前面的那个字符，在菜单标题上会出现一条下划线。可以通过按 Alt 键并按下有下划线的那个字符键来激活菜单。字符

本身不显示在菜单标题上，如果要将“&”字符显示在菜单标题上，需要在字符串中使用两个“&”字符。

- Separator 属性　Separator 属性的取值可以是 on 或 off。其中，off 是默认值。通过该属性，可以在菜单上设置分隔条。在某个菜单项上设置了这个属性后，该菜单的上面会出现一条分隔条。
- Visible 属性　Visible 属性的取值可以是 on 或 off。其中，on 是默认值。通过该属性，可以控制菜单的可见状态。默认时，所有菜单都是可见的。当设置 Visible 的属性值为 off 时，菜单就成为不可见了，但菜单仍然存在，仍然可以查询、设置菜单的属性。

（2）基本信息属性

- Accelerator 属性　Accelerator 属性的取值是一个字符。该属性用于设置菜单的加速键。使用加速键，可以代替鼠标来选择菜单项。在 WINDOWS 系统中，加速键是 Ctrl+设置的字符，c，v，x 等字符被保留作为默认的菜单项；在 UNIX 环境下，加速键是 Ctrl+设置的字符，o，p，s 和 w 等字符被保留作为默认的菜单项。需要注意的是，只有对没有子菜单(children)的菜单项设置该属性的属性值才有效。通过加速键，可以执行菜单的 callback 功能。
- Children 属性　Children 属性的取值是子菜单项的句柄向量，句柄向量包括本菜单对象下的所有子菜单项。该属性用于设置本级菜单下的子菜单项。如果 Children 取值向量为空，表示该菜单下没有子菜单。可以通过 Children 属性重新排列各子菜单项的顺序。
- Enable 属性　Enable 属性的取值可以是 on 或 off。其中，on 是默认值。通过该属性，可以设置菜单项为有效或失效。该属性用于控制菜单项是否可用。如果把该属性设置为 off，那么菜单项的标题名称（用 Label 属性设置）就会成为灰色，表示该菜单不可用。
- Parent 属性　Parent 属性的取值是本级菜单的父对象的句柄。一个菜单的父对象是显示该菜单栏的图形窗口的窗口；或者是一个菜单项，此时本菜单一定是子菜单。通过设置该属性的属性值为另一个父对象句柄，可以把本菜单移到另一个图形窗口或另一个菜单上。
- Tag 属性　Tag 属性的取值是一个字符串。通过该属性，可以标记菜单项的名字，而后在进行程序设计时，可以通过该名字来指定菜单项，它与 Visual Basic 中控件对象的名称属性功能相同。Tag 属性提供了一种标记图形对象的办法，特别是在进行交互式图形程序设计时，无需把对象句柄设为全局变量，或把它们作为参数传递给 callback 函数中。直接通过该属性，就可以调用该图形对象。
- Type 属性　Type 属性是只读的字符串，用来标识图形对象的类型。对 uimenu 对象来说，该属性的属性值永远是字符串“uimenu”。
- UserData 属性　UserData 属性的取值是一个矩阵。通过该属性可以保存与该菜单对象有关的信息或数据。可以通过函数 set 和 get 来调用这些信息。

（3）位置信息属性

- Position 属性　Position 属性的取值是一个标量，默认值是 1。该属性用于确定菜单的位置，属性值标明了本菜单在菜单栏中的位置。顶层菜单在菜单栏上的位置是从左到右，Position 属性值为 1 表明本菜单在菜单栏的最左边。下拉式菜单在菜单栏中的

位置从上到下，Position 属性值为 1 表明本菜单在菜单栏的最上边。

（4）Callback 例程执行控制属性

- BusyAction 属性　　BusyAction 属性的取值是 cancel 或 queue，默认值是 queue。该属性用于决定执行菜单对象的 callback 例程的中断调用方式。如果有一个 callback 例程正在执行，那么当用户使用触发器（如鼠标单击）触发正在执行 callback 例程对象的另一个 callback 例程时，随后的 callback 例程总是试图中断先前正在执行的 callback 例程。随后要执行的 callback 例程只有包含 drawnow，figure，getframe，pause，waitfor 等命令时，先前正在运行的 callback 例程才能被中断。

  如果正在执行 callback 例程的对象的 Interruptible 属性值是 on（on 是 Interruptible 属性的默认值），那么正在执行的 callback 例程被中断。如果 Interruptible 属性值是 off，那么 BusyAction 属性决定 callback 例程的中断调用方式。如果 BusyAction 的属性值是 queue，那么随后要运行的 callback 例程加入事件队列中，正在执行当前 callback 例程的事件执行完后，才执行第二个调用 callback 例程的事件。如果 BusyAction 的属性值是 cancel，那么运行第二个 callback 例程的事件被取消。

  需要注意的是，如果调用第二个 callback 例程的事件是 DeleteFcn 或 CreateFcn，或者是图形对象的 CloseRequest 或 ResizeFcn，那么不管 Interruptible 的属性值是什么，都会中断正在运行的 callback 例程，而执行下一个包含 drawnow，figure，getframe，pause，waitfor 命令的 callback 例程。
- Callback 属性　　Callback 属性的取值是一个字符串，该属性定义菜单对象的控制动作。当单击菜单对象时，就执行 Callback 例程。定义的字符串是一个有效的 MATLAB 表达式，或者是一个 M 文件的名字，字符串在 MATLAB 的命令窗口内执行。

  一个有子菜单的菜单项，在显示子菜单前执行 Callback 例程。一个没有子菜单的菜单项，在松开鼠标时执行 Callback 例程。
- CreateFcn 属性　　CreateFcn 属性的取值是一个字符串。该字符串是一个有效的 MATLAB 表达式，或者是一个 M 文件的名字。该属性定义一个在菜单对象创建阶段执行的 Callback 例程。对菜单对象，必须定义该属性为菜单的默认值。例如：

  set(0,'DefaultUimenuCreateFcn','set(gcf,"IntegerHandle","off")')

  上述命令在根对象层定义菜单对象的默认值，设置了图形对象的 IntegerHandle 属性值为 off。改变该属性的属性值，对已经存在的菜单对象没有影响。MATLAB 在创建菜单对象时，先设置了所有属性的属性值后，再执行该属性定义的 Callback 例程。

  对于正在执行 CreateFcn 属性的对象的句柄，只有通过根对象的 CallbackObject 属性，才能访问。根对象的 CallbackObject 属性可以通过函数 gcbo 获得。
- DeleteFcn 属性　　DeleteFcn 属性的取值是一个字符串。该字符串是一个有效的 MATLAB 表达式，或者是一个 M 文件的名字。该属性定义了在菜单对象的删除阶段（例如，使用 delete 函数删除菜单对象，或包含菜单对象的图形窗口对象重新设置时）执行的 Callback 例程。MATLAB 在删除对象的所有属性前执行 DeleteFcn 属性定义的 Callback 例程，所以对象的所有属性值在该 Callback 例程中还是可用的。

  对于正在执行 DeleteFcn 属性的对象的句柄，只有通过根对象的 CallbackObject 属性，才能访问。根对象的 CallbackObject 属性可以通过函数 gcbo 获得。
- Interruptible 属性　　Interruptible 属性的取值是 on 或 off，默认值是 on。该属性决定

Callback 例程的中断调用模式。如果有一个 Callback 例程正在执行，那么当用户使用触发器（如鼠标单击）触发正在执行 callback 例程的对象的另一个 callback 例程时，随后的 callback 总是试图中断先前正在执行的 callback 例程。随后要执行的 callback 例程只有包含 drawnow，figure，getframe，pause，waitfor 等命令时，先前正在运行的 callback 例程才能被中断。

MATLAB 依据下列因素执行随后的 Callback 例程 :如果正在执行 Callback 例程的对象的 Interruptible 属性是 on，那么正在执行的 Callback 例程能够被中断，执行下一个包含 drawnow，figure，getframe，pause，waitfor 等命令的 Callback 例程。如果 Interruptible 属性值是 off，那么 BusyAction 属性决定 callback 例程的中断调用方式。

需要注意的是，如果第二个 callback 是 DeleteFcn 或 CreateFcn，或者是图形窗口对象的 CloseRequest 或 ResizeFcn，那么不管 Interruptible 的属性值是什么，都会中断正在运行的 callback 例程，而执行下一个包含 drawnow，figure，getframe，pause，waitfor 等命令的 Callback 例程。此时，按照上述规则，图形窗口对象的 WindowButtonDownFcn 属性的 Callback 例程与菜单对象的 ButtonDownFcn 属性的 Callback 例程被执行。

（5）控制操作属性

- HandleVisibility 属性　　HandleVisibility 属性的取值是 on，callback，off。其中，on 是默认值。该属性定义菜单对象句柄被命令行用户和 GUI 用户可访问的权限。通过该属性，可以决定对象句柄是否在父对象 Children 属性的属性值向量中可见。如果 HandleVisibility 属性值是 on，那么对象句柄一直是可见的；如果 HandleVisibility 属性值是 callback，那么在该 callback 例程或引起 callback 例程调用的函数中是可见的，但是在命令行调用的函数中不可见，这对于保护 GUI 用户免受命令行用户的影响很有用。HandleVisibility 属性对于阻止命令行用户任意利用该对象句柄画图或利用该对象句柄删除该对象很有用。如果 HandleVisibility 属性值是 off，那么对象句柄一直是不可见的。这在一个 callback 例程调用一个可能破坏 GUI 的函数时，可能避免对象受不必要的影响。当对象句柄不可见时，在父对象 Children 属性的属性值向量中见不到该句柄，也不能被查找对象句柄或者查询句柄属性的函数获得。这些函数包括 get，findobj，gca，gcf，gco，newplot，cla，clf，close 等。

  当菜单对象的 HandleVisibility 属性值设置为 callback 或 off 时，对象句柄在父对象 Children 属性值向量中不可见。在根对象的 CurrentFigure 属性中不会显示图形；在根对象的 CallbackObject 属性或者在图形窗口对象的 CurrentObject 属性中不会显示对象；在父对象的 CurrentAxes 属性中不会显示坐标轴。

  可以通过设置根对象的 ShowHiddenHandles 属性为 on，来查看所有对象的句柄，而不管对象的 HandleVisibility 属性设置为何值。这对 HandleVisibility 属性的属性值不会有影响。

  当对象的 HandleVisibility 属性被设置为隐藏时，对象句柄仍然是有效的。如果知道对象的句柄，可以通过把对象句柄传递给操作句柄的函数，来改变对象的属性。

2．uicontextmenu 菜单的属性

（1）外观及风格控制属性

- Visible 属性　　Visible 属性的取值可以是 on 或 off。其中，off 是默认值。通过该属性，可以控制 uicontextmenu 菜单的可见状态。默认时，所有菜单都是不可见的。Visible

属性的属性值表明 uicontextmenu 菜单是否可见。当 uicontextmenu 菜单可见时，它的值是 on；当不可见时，它的值是 off。当 uicontextmenu 菜单可见时，Position 属性决定 uicontextmenu 菜单的位置。

- Position 属性　　Position 属性的取值是一个二维向量。该属性的二维属性值用于定义 Visible 属性值为 on 的 uicontextmenu 菜单的位置。二维向量是[left bottom]，表明 uicontextmenu 菜单左下角与图形窗口对象左下角的水平与垂直距离。距离的单位是像素。

（2）基本信息属性

- Children 属性　　Children 属性的取值是子菜单项的句柄向量。该属性用于设置本级菜单的子菜单项。句柄向量包括本菜单对象下的所有子菜单项。如果 Children 取值向量为空，表示该菜单下没有子菜单项。子菜单项可以是定义的 uimenu 菜单。
- Parent 属性　　Parent 属性的取值是本级菜单的父对象的句柄。一个菜单的父对象是显示该菜单栏的图形窗口对象。通过设置该属性的属性值为另一个图形窗口对象句柄，可以把本菜单移到一个新的图形窗口上。
- Tag 属性　　Tag 属性的取值是一个字符串。通过该属性，可以标记菜单项的名字，而后在进行程序设计时，可以利用该名字来指定菜单项，它与 Visual Basic 中控件对象的名称属性功能相同。Tag 属性提供了一种标记图形对象的办法。特别是在进行交互式图形程序设计时，无需把对象句柄设为全局变量，或把它们作为参数传递给 callback 函数。直接通过该属性，就可以调用该图形对象。
- Type 属性　　Type 属性的取值是只读的字符串，用来标识图形对象的类型。对 uicontextmenu 对象来说，此属性的属性值永远是字符串“uicontextmenu”。
- UserData 属性　　UserData 属性的取值是一个矩阵。通过该属性可以保存与该菜单对象有关的信息或数据。可以通过函数 set 和 get 来调用这些信息。

（3）Callback 例程执行控制属性

- BusyAction 属性　　BusyAction 属性的取值是 cancel 或 queue，默认值是 queue。该属性用来决定执行菜单对象的 callback 例程的中断调用方式。如果有一个 callback 例程正在执行，那么随后执行 callback 例程的事件总是试图中断先前正在执行的 callback 例程。

  如果正在执行 callback 例程对象的 Interruptible 属性值是 on（on 是 Interruptible 属性的默认值），那么在下一个事件队列运行点，正在执行的 callback 例程被中断。如果 Interruptible 属性值是 off，那么 BusyAction 属性决定 callback 例程的中断调用方式。如果 BusyAction 的属性值是 queue，那么随后要运行的 callback 例程加入事件队列中，当前正在运行的 callback 例程执行完后，就执行第二个 callback 例程。如果 BusyAction 的属性值是 cancel，那么执行第二个 callback 例程的事件被取消。
- Callback 属性　　Callback 属性的取值是一个字符串，该属性定义菜单对象的控制动作。当用鼠标右击菜单对象时，就执行 callback 例程。在 uicontextmenu 显示之前，该例程先被执行。定义的字符串是一个有效的 MATLAB 表达式，或者是一个 M 文件的名字，字符串在 MATLAB 的命令窗口内执行。
- CreateFcn 属性　　CreateFcn 属性的取值是一个字符串。该字符串是一个有效的 MATLAB 表达式，或者是一个 M 文件的名字。该属性定义一个当 MATLAB 创建 uicontextmenu 菜单对象时执行的 Callback 例程。对 uicontextmenu 菜单对象，必须定

义该属性为菜单的默认值。例如：

```
set(0,'DefaultUicontextmenuCreateFcn','set(gcf,"IntegerHandle","off")')
```

上述命令在根对象层定义菜单对象的默认值，设置了图形对象的 IntegerHandle 属性值为 off。改变该属性的属性值，对已经存在的菜单对象没有影响。MATLAB 在创建菜单对象时，先设置了所有属性的属性值后，再执行该属性定义的 Callback 例程。

对于正在执行 CreateFcn 属性的对象的句柄，只有通过根对象的 CallbackObject 属性，才能访问。根对象的 CallbackObject 属性可以通过函数 gcbo 获得。

- DeleteFcn 属性　DeleteFcn 属性的取值是一个字符串。该字符串是一个有效的 MATLAB 表达式，或者是一个 M 文件的名字。该属性定义了在 uicontextmenu 菜单对象的删除阶段（例如，使用 delete 函数删除菜单对象，或清除包含 uiconextmenu 菜单对象的图形窗口对象等）执行的 Callback 例程。MATLAB 在删除对象的所有属性前，执行 DeleteFcn 属性定义的 Callback 例程，所以对象的所有属性值在该 Callback 例程中还是可用的。

  对于正在执行 DeleteFcn 的对象的句柄，只有通过根对象的 CallbackObject 属性，才能访问。根对象的 CallbackObject 属性可以通过函数 gcbo 获得。
- Interruptible 属性　Interruptible 属性的取值是 on 或 off，默认值是 on。该属性决定 uicontextmenu 菜单的 Callback 例程的中断调用模式。在默认模式下，正在执行的 Callback 例程能够被中断。只有为 ButtonDownFcn 事件定义的 Callback 属性才被此属性影响。随后要执行的例程中只有包含 drawnow，figure，getframe，pause，waitfor 等命令时，MATLAB 才检查先前正在执行的 callback 例程是否能被中断。

（4）控制操作属性

- HandleVisibility 属性　HandleVisibility 属性的取值是 on，callback，off。其中，on 是默认值。该属性定义菜单对象句柄被命令行用户和 GUI 用户访问的权限。通过该属性，可以决定对象句柄是否在父对象 Children 属性的属性值向量中可见。如果 HandleVisibility 属性值是 on，那么对象句柄一直是可见的。如果 HandleVisibility 属性值是 callback，那么在该 callback 例程或引起 callback 例程调用的函数中是可见的，但在命令行调用的函数中不可见，这对于保护 GUI 用户免受命令行用户的影响很有用。HandleVisibility 属性对于阻止命令行用户任意利用该对象句柄画图，或者利用该对象句柄删除该对象时很有用。如果 HandleVisibility 属性值是 off，那么对象句柄一直是不可见的，这在一个 callback 例程调用一个可能破坏 GUI 的函数时，可以避免对象受不必要的影响。当对象句柄是不可见时，在父对象 Children 属性的属性值向量中见不到该句柄，也不能被查找对象句柄或者查询句柄属性的函数获得。这些函数包括 get，findobj，gca，gcf，gco，newplot，cla，clf，close 等。

  当菜单对象的 HandleVisibility 属性值设置为 callback 或 off 时，对象句柄在父对象 Children 属性值向量中不可见。在根对象的 CurrentFigure 属性中不会显示图形；在根对象的 CallbackObject 属性或者在图形窗口对象的 CurrentObject 属性中不会显示对象；在父对象的 CurrentAxes 属性中不会显示坐标轴。

  可以通过设置根对象的 ShowHiddenHandles 属性为 on，来查看所有对象的句柄，而不管对象的 HandleVisibility 属性设置为何值。这对 HandleVisibility 属性的属性值不会有影响。

当对象的 HandleVisibility 属性被设置为隐藏时，对象句柄仍然是有效的。如果知道对象的句柄，可以把对象句柄传递给操作句柄的函数，就可以改变对象的属性。

在命令行方式下或者在 GUI 设计方式下，用户可以方便地对 uimenu 菜单与 uicontextmenu 菜单的属性进行设置、修改。在本章的第 8 节——GUI 的 M 文件举例中将介绍设置菜单的属性。

8.4 控件

在绝大多数的图形用户界面下，都包含有控件。控件是图形对象，它与菜单一起用于建立图形用户界面。通过使用各种类型的控件，可以建立起操作简便、功能强大的图形用户界面。在各种进行程序设计的软件中，如 Visual Basic，Dephi，Visual C++，都可以使用大量的控件建立图形用户界面。例如，WINDOWS 系统的 Visual Basic 程序设计软件中，利用 Grid 控件可以实现与 Microsoft Excel 电子表格相类似的界面。MATLAB 也提供了多种控件，可以把它们放置在图形窗口的任何位置，并用鼠标激活它们。MATLAB 支持的控件对象有 10 种。下面，对 MATLAB 中的控件进行介绍。

8.4.1 控件对象类型

MATLAB 支持复选框 Check boxes，可编辑文本框 Editable text，框架 Frames，列表框 List boxes，弹出式菜单 Pop-up menus（其功能与 Visual C++等程序设计软件中的组合框一样），命令按钮 Push buttons，单选按钮 Radio buttons，滑标 Sliders，静态文本框 Static text，开关按钮 Toggle buttons 等 10 种类型的控件对象。下面对它们进行简单的介绍。

1．复选框（Check boxes）

复选框有一个标志文本，在标志文本的左边有一个小方框。它对于用户进行大量的独立选择很有用。为了激活复选框，可以使用鼠标单击复选框对象，使复选框在选中与不选中两种状态间进行切换。当选中时，复选框的小方框内会有一个“×”，此时复选框的 Value 属性值是 1；当没有选中时，复选框的小方框内为空，此时复选框的 Value 属性值为 0。复选框的 Style 属性值是 checkbox。

2．可编辑文本框（Editable text）

当需要输入文本时，可以使用可编辑文本框。通过可编辑文本框，用户可以方便地输入或修改已经存在的文本，这与文本编辑器的功能是一样的。可编辑文本框的 String 属性中存储输入的文本。

在 WINDOWS 系统中，如果可编辑文本框获得了焦点，单击菜单条不会引起可编辑文本框的 Callback 例程的执行。但在 UNIX 系统中，单击菜单条会引起可编辑文本框的 Callback 例程的执行。因此，在 WINDOWS 系统中，在菜单条上单击后，即使在屏幕上的文本串已经发生了变化，命令 get(edit_handle,'String')也不会返回当前可编辑文本框中的文本串，因为 MATLAB 必须执行更新 String 属性值的 Callback 例程。

可编辑文本框可以是单行或多行文本模式。当可编辑文本框是单行模式时，只允许输入单行文本串；当可编辑文本框是多行模式时，可以输入多行文本串。可编辑文本框的 Style 属性值是 edit。

3．框架（Frames）

框架对象是在图形窗口内，视觉上封闭起来的一个区域，只有控件可以在框架中使用。

一般把作用相关的一组控件用框架框起来，这样的用户界面很容易被用户理解。框架没有 Callback 例程。

框架是不透明的，所以，设置框架与框架中控件的顺序很重要。如果框架内的控件先于框架被设置，那么框架设置后就会覆盖原先设置的控件。一般地讲，应在定义了框架后再定义框架中的控件。框架的 Style 属性值是 frame。

4．列表框（List boxes）

列表框中列出列表框的 String 属性的字符串项。用户可以方便地选择一个或多个列表项。列表框的 Max 与 Min 属性控制选择模式；Value 属性标明选择的列表项的索引值。当列表框上的鼠标松开后，MATLAB 会调用 Callback 例程。一般地，用鼠标单击与双击列表框的效果是不一样的。列表框的 Style 属性值是 listbox。

5．弹出式菜单（Pop-up menus）

弹出式菜单的功能与 Visual C++等程序设计软件中组合框的功能是一样的。它有一个显示信息的框，框的右边有一个下拉式箭头。单击下拉式箭头，就会显示一个列表，里面包含 String 属性定义的属性值。当没有打开列表时，信息框内显示的是当前选择的表项。当打开列表，从中选择一个表项并单击后，该表项就会出现在信息显示框内。弹出式菜单对于用户进行大量的互相不同的选择是很有用的。如果不用弹出式菜单，那么就必须设置大量互不相同的单选按钮。弹出式菜单的 Style 属性值是 popupmenu。

6．命令按钮（Push buttons）

命令按钮是一个矩形的凸出对象。在命令按钮对象上标有一个字符串，用于标识该命令按钮。单击命令按钮，会产生相应的动作。用鼠标单击命令按钮后，命令按钮会凹下，但当松开鼠标后，命令按钮又会弹起，这与下面将要介绍的开关按钮不同。命令按钮的 Style 属性值是 pushbutton。

7．单选按钮（Radio buttons）

单选按钮与复选框相似，单选按钮有一个标志文本，在标志文本的左边有一个小圆圈。它对于用户进行功能互斥的选择很有用。在一组单选按钮中，一次只能有一个单选按钮被选中，这与可以同时选中多个的复选框不同。为了激活单选按钮，可以使用鼠标单击单选按钮对象，使单选按钮在选中与不选中两种状态间进行切换。当选中时，复选框的小圆圈内有一个黑点“ · ”，此时单选按钮的 Value 属性值是 1；当没有选中时，单选按钮的小圆圈内为空，此时单选按钮的 Value 属性值为 0。单选按钮的 Style 属性值是 radiobutton。

8．滑标（Sliders）

滑标，其功能类似于滚动条，它由 3 个部分组成，分别是滚动槽、滚动槽内的指示条和滚动槽两端的箭头。其中，滚动槽表明滑标的有效值范围，指示条表明滑标的当前值，通过箭头可以左、右移动指示条。用户在选中指示条后通过鼠标拖动指示条，可以改变滑标的值，也可以通过单击滚动槽两端的箭头来改变滑标的值。可以通过函数设置滑标的最小值、最大值与当前值。滑标的 Style 属性值是 slider。

9．静态文本框（Static text）

静态文本框静态显示文本字符串。静态文本框通常用于显示别的控件的有关信息。例如，如果与滑标相连，可以在静态文本框中显示滑标的当前值。与可编辑文本框不同，用户不能交互地改变静态文本框中的内容。静态文本框没有 Callback 例程。静态文本框的 Style 属性值是 text。

10．开关按钮（Toggle buttons）

开关按钮的外观与命令按钮类似，是一个矩形的凸出对象，同时，在开关按钮对象上也标有一个字符串，用于标识该开关按钮。与命令按钮不同的是，当用鼠标单击开关按钮并松开后，开关按钮不会弹起，再单击一次，它才会弹起，这可以表明开关按钮的状态。单击开关按钮，会产生相应的动作，执行相应的 Callback 例程。在进行工具条的设计时，开关按钮是非常有用的。开关按钮的 Style 属性值是 togglebutton。

8.4.2 控件建立

与菜单对象一样，可以通过命令行方式与 GUI 设计工具两种方式来建立控件。

1．命令行方式

在命令行方式下，可以通过函数 uicontrol 来建立控件对象。uicontrol 函数有如下几种调用格式：

```
handle = uicontrol(parent)
handle = uicontrol(…,'PropertyName',PropertyValue,…)
```

其中，handle 创建的是控件对象的句柄值；parent 是控件所在的图形窗口的句柄值；PropertyName 是控件的某个属性的属性名；PropertyValue 是与属性名相对应的属性值。第一种调用方式，采用 uicontrol 的 Style 属性的默认属性值，在图形窗口的左下角创建一个命令按钮。第二种调用方式，省略控件所在图形窗口的句柄值，表示在当前图形窗口中创建控件对象，如果此时无图形窗口，MATLAB 会自动创建一个图形窗口，然后在其中创建控件对象。在第二种调用方式中，可以设置所要创建的控件的多个属性的属性值。

函数 uicontrol 可接受的输入参数有：属性名/属性值输入对，结构（structures），向量等。函数的输出参数，即创建的控件对象的句柄值是可选的。在创建控件对象后，可以通过函数 set 来设置或修改控件的属性值；也可以通过函数 get 来查看控件对象的属性值。

下面的例子是：创建一个命令按钮控件，按钮控件上标的字符串是 Clear。按钮控件的 Callback 属性值是 cla，它用于清除当前的坐标轴。因此，当用户单击按钮控件时，清除当前的坐标轴。

```
h = uicontrol('Style', 'pushbutton', 'String', 'Clear',…
'Position', [20 150 100 70], 'Callback', 'cla');
```

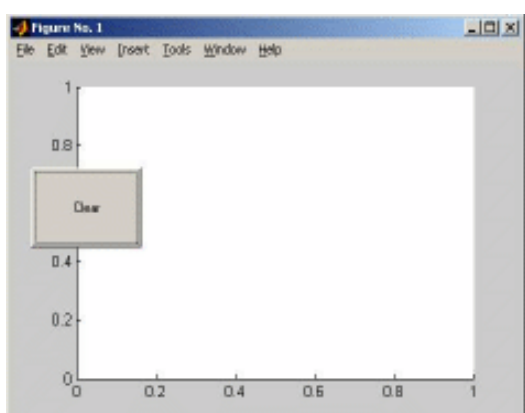


图 8-19　创建用于清除坐标轴的按钮控件

用户单击按钮控件后的结果如图 8-19 所示。

下面所示为创建弹出式控件（其功能与 Visual C++等程序设计软件中的组合框一样）。该弹出式控件中有 black，white，blue，red 等 4 种可选的表项，用户单击任何一个表项，就会调用 Callback 例程，该例程是一个脚本式 M 文件，用于改变图形窗口的颜色。

```
hpop = uicontrol('Style', 'popup',…
    'String', 'black|white|blue|red',…
    'Position', [20 320 100 50],…
    'Callback', 'setmap');
```

下面所示为脚本式 M 文件 setmap.m。

```
val = get(hpop,'Value');
if val == 1
    set(gcf,'color',[0 0 0])
elseif val == 2
    set(gcf,'color',[1 1 1])
elseif val == 3
    set(gcf,'color',[0 0 1])
elseif val == 4
    set(gcf,'color',[1 0 0])
end
```

在该脚本式 M 文件中，用函数 set 来设置图形窗口的颜色。

由于控件对象是图形窗口对象的子对象，因此当把控件对象放在图形窗口后，不需要设置专门的坐标轴，这从上面两个实例中也可以看出。

2．GUI 设计工具

在命令行方式下，创建控件对象时，需要记住用于创建控件对象的函数。而且，要设置控件的属性，必须记住控件的大量属性的属性名，这对于修改控件的属性非常不方便。然而，利用 GUI 设计工具中的对象设计编辑器（Layout Editor），可以容易地创建 MATLAB 支持的各种控件，而且通过对象属性查看器（Property Inspector），可以方便地修改、设置创建的控件的属性值（具体创建方式，请参阅第二节 GUI 设置工具的介绍）。图 8-20 所示，是用对象设计编辑器（Layout Editor）创建的控件界面。

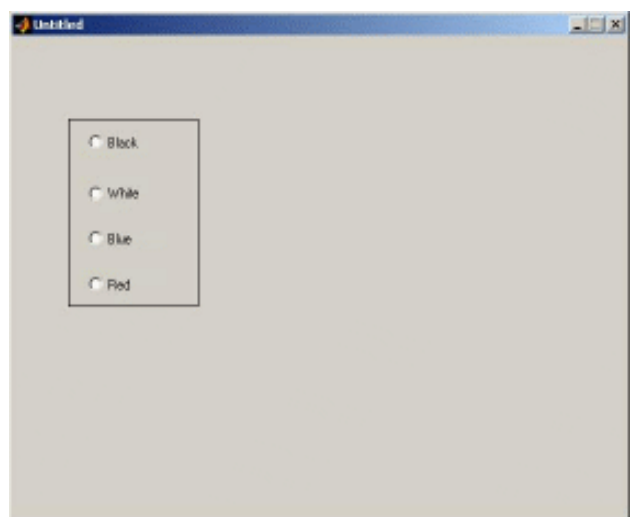


图 8-20　用对象设计编辑器创建的控件界面

图 8-20 中，在创建的控件界面中，有一个 Frame 控件与 4 个分别标识为 Black，White，Blue，Red 的单选按钮控件。在创建了控件界面后，再编写与每个单选按钮控件的 Callback 例程，就成为一个完全可以使用的简单 GUI 界面了。下面编写的是与图 8-20 所示界面相匹配的 Callback 例程的程序代码。

```
function varargout = radiobutton1_Callback(h, eventdata, handles, varargin)
```

```
% 当用户单击第一个单选按钮后，调用的 Callback 例程
% 在该 Callback 例程中，设置本图形窗口的颜色为黑色，并使别的单选按钮为没有选中
% 即设置它们的 Value 属性值为 0
disp('radiobutton1 Callback not implemented yet.')
set(handles.figure1,'color',[0 0 0])
set(handles.radiobutton2,'value',0)
set(handles.radiobutton3,'value',0)
set(handles.radiobutton4,'value',0)

% --------------------------------------------------------------------
function varargout = radiobutton2_Callback(h, eventdata, handles, varargin)
% 当用户单击第二个单选按钮后，调用的 Callback 例程
% 在该 Callback 例程中，设置本图形窗口的颜色为白色，并使别的单选按钮为没有选中
disp('radiobutton2 Callback not implemented yet.')
set(handles.figure1,'color',[1 1 1])
set(handles.radiobutton1,'value',0)
set(handles.radiobutton3,'value',0)
set(handles.radiobutton4,'value',0)

% --------------------------------------------------------------------
function varargout = radiobutton3_Callback(h, eventdata, handles, varargin)
% 当用户单击第四个单选按钮后，调用的 Callback 例程
% 在该 Callback 例程中，设置本图形窗口的颜色为蓝色，并使别的单选按钮为没有选中
disp('radiobutton3 Callback not implemented yet.')
set(handles.figure1,'color',[0 0 1])
set(handles.radiobutton2,'value',0)
set(handles.radiobutton1,'value',0)
set(handles.radiobutton4,'value',0)

% --------------------------------------------------------------------
function varargout = radiobutton4_Callback(h, eventdata, handles, varargin)
% 当用户单击第四个单选按钮后，调用的 Callback 例程
% 在 Callback 例程中，设置本图形窗口的颜色为红色，并使别的单选按钮为没有选中
disp('radiobutton4 Callback not implemented yet.')
set(handles.figure1,'color',[1 0 0])
set(handles.radiobutton2,'value',0)
set(handles.radiobutton3,'value',0)
set(handles.radiobutton1,'value',0)
```

8.4.3 控件属性

（1）外观及风格控制属性

- BackgroundColor 属性　BackgroundColor 属性用于设置控件的背景颜色，默认值是系统定义的颜色。该属性的取值可以是一个 1 行 3 列向量，此时设置的是一个 RGB 颜色，例如，[0 0 0]表示设置为黑色。向量中元素的取值必须在区间[0 1]内，向量中 3 个元素分别代表 red，green，blue。也可以取 MATLAB 中预定义的 8 个颜色名（可以使用长颜色名或短颜色名）来设置该属性的属性值。可以通过查看 MATLAB 中的函数 colorSpec 来了解关于颜色的更详细信息。表 8-1 是 MATLAB 中预定义的颜色名及与之对应的 RGB 值。

表 8-1　MATLAB 定义的颜色名及 RGB 颜色

RGB 颜色值	短 颜 色 名	长 颜 色 名
[1 1 0]	y	yellow
[1 0 1]	m	magenta
[0 1 1]	c	cyan
[1 0 0]	r	red
[0 1 0]	g	green
[0 0 1]	b	blue
[1 1 1]	w	white
[0 0 0]	k	black

- Cdata 属性　Cdata 属性的取值是一个矩阵。该属性表明显示在控件上的图像的颜色值。该属性的取值是一个 1 行 3 列向量的 RGB 颜色，向量中元素的取值必须在区间[0 1]内，向量中 3 个元素分别代表 red，green，blue。
- ForegroundColor 属性　ForegroundColor 属性用于设置控件上显示的文本的颜色，即用于确定控件的 String 属性包含的字符串的颜色，默认属性值是黑色。该属性的取值可以是一个 1 行 3 列向量的 RGB 颜色，向量中元素的取值必须在区间[0 1]内，向量中 3 个元素分别代表 red，green，blue。也可以取 MATLAB 中预定义的 8 个颜色名（如表 8-1 所示）。可以通过查看 MATLAB 中的函数 colorSpec 来了解颜色的更详细信息。
- SelectionHighlight 属性　SelectionHighlight 属性的取值可以是 on 与 off。其中，on 是默认值。该属性用于确定当控件被选中时，是否显示被选中的状态。SelectionHighlight 属性要与 Selected 属性一起使用，共同控制控件对象的选中状态。
- String 属性　String 属性的取值是一个字符串。该属性用于设置控件上显示的文本串。对于复选框、可编辑文本框、命令按钮、单选按钮、静态文本框和开关按钮控件，字符串显示在控件界面上；对于列表框与弹出式菜单，字符串显示在控件的列表项中。

  对于只能显示一行文本的控件对象，如果字符串是一个矩阵字符串，那么只有第一个元素的几个字符能被显示，后面的字符被忽略。对于静态文本框，String 矩阵中的每行，单元数组的每块，从字符“\n”定义的地方开始分行。对于包含多个列表项的列表框与组合框，可以定义 String 的属性值是一个字符矩阵，或定义成一个中间被字符“|”隔开的字符串。对于可编辑文本框，String 属性的属性值是用户输入可编辑文本框中的字符串。
- Visible 属性　Visible 属性的取值可以是 on 或 off。其中，on 是默认值。通过该属性，可以控制控件的可见状态。默认时，所有控件都是可见的。当设置 Visible 的属性值是

off 时，控件就成为不可见了，但控件仍然存在，仍然可以查询、设置控件的属性。

（2）基本信息属性

- Enable 属性　　Enable 属性的取值可以是 on, inactive 或 off。其中，on 是默认值。通过该属性，可以使控件有效或失效。该属性用于决定鼠标单击控件时控件的反应情况，包括控件的 Callback 例程的执行与否。如果属性值是 on，表示控件是可用的；如果属性值是 inactive，表示控件是不可用的，但是外表看起来，控件与属性值是 on 时一样；如果属性值是 off，表示控件是不可用的，而且外表看起来是灰色。

  当设置属性值是 on 时，用鼠标左键单击控件，那么 MATLAB 按次序执行下列动作：设置图形窗口的 SelectionType 属性，执行控件的 Callback 例程；不设置图形窗口的 CurrentPoint 属性，也不执行控件的 ButtonDownFcn 事件与图形窗口的 WindowButtonDownFcn 事件。

  当设置属性值是 off 或 inactive 时，用鼠标左键单击控件；或者当属性有值时，用鼠标右键单击控件，那么 MATLAB 按次序执行下列动作：设置图形窗口的 SelectionType 属性与 CurrentPoint 属性 执行图形窗口的 WindowButtonDownFcn 事件; 执行控件的 ButtonDownFcn 事件的 Callback 例程；如果控件与某个 context menu 菜单连在一起，那么当鼠标右键单击控件时，会显示那个 context menu。此时，如果选中某个子菜单项，那么会执行该菜单项的 Callback 例程。
- Parent 属性　　Parent 属性的取值是本级控件的父对象的句柄。一个控件的父对象是显示该控件的图形窗口。通过设置 Parent 的属性值为另一个父对象句柄，可以把本控件移到另一个图形窗口对象。
- Selected 属性　　Selected 属性的取值可以是 on 或 off。其中，on 是默认值。该属性用于确定控件对象是否被选中。当属性值是 on 时，并且 SelectionHighlight 属性值也是 on 时，MATLAB 显示选中的控件的句柄。例如，可以在 ButtonDownFcn 事件的 Callback 例程中，设置这个属性的属性值，以允许用户使用鼠标选择控件对象。
- SliderStep 属性　　该属性只对滑标控件有效。通过该属性，可以控制滑标每次移动的步长。它的取值是一个包含两个元素的向量[min_step max_step]，分别表示最小步长与最大步长。当鼠标单击滑标两端的箭头时，滑标移动的是最小步长；当鼠标在滑槽中单击时，滑标移动的是最大步长。向量中的两个元素取值必须在区间[0 1]内，默认值是[0.01 0.10]，表示当鼠标单击滑标两端的箭头时，滑标移动距离为整个滑标范围的 1%；当鼠标在滑槽中单击时，滑标移动距离为整个滑标范围的 10%。例如，创建如下的滑标：

  ```
  uicontrol('Style','slider','Min',1,'Max',7,…
            'SliderStep',[0.1 0.6])
  ```

  那么鼠标单击滑标两端的箭头时，滑标的移动距离=0.1*(8-1)，即 0.7000；当鼠标在滑槽中单击时，滑标的移动距离=0.6*(8-1)，即 4.2000。
- Style 属性　　Style 属性用于决定所创建的控件的类型。Style 属性可以取如下所示的属性值：pushbutton，togglebutton，radiobutton，checkbox，edit，text，slider，frame，listbox，popupmenu。其中，pushbutton 是默认的属性值。
- Tag 属性　　Tag 属性的取值是一个字符串。通过该属性，可以标记控件的名字，而后在进行程序设计时，可以利用该名字来指定控件。它与 Visual Basic 中控件对象的名

称属性功能相同。Tag 属性提供了一种标记图形对象的办法。特别是在进行交互式图形程序设计时，无需把对象句柄设为全局变量，或把它们作为参数传递给 callback 函数。直接通过该属性，就可以调用该图形对象。

- TooltipString 属性　　TooltipString 属性的取值是一个字符串。设置该属性后，当用户把鼠标移到控件上时，就会显示该属性定义的字符串。通过该属性，可以简单地说明控件的作用。
- Type 属性　　Type 属性是只读的字符串，用来标识图形对象的类型。对 uicontrol 对象来说，此属性的属性值永远是字符串“uicontrol”。
- UserData 属性　　UserData 属性的取值是一个矩阵。通过该属性可以保存与该控件对象有关的信息或数据。可以通过函数 set 和 get 来调用这些信息。

（3）位置信息属性

- Position 属性　　Position 属性用于确定控件的位置及大小，属性值标明了本控件在图形窗口的位置及大小。属性的取值是位置向量[left bottom width height]，默认值是[20 20 60 20]。其中，元素 left，bottom 表示控件对象的左下角距离图形窗口左下角的水平与垂直距离；元素 width，height 表示控件的宽度与高度。距离的单位由属性 units 决定。

  在 WINDOWS 系统中，弹出式菜单 pop-up menu 的高度由字体的尺寸自动决定。在属性 Position 内设置的高度值对弹出式菜单无效。

  对于滑标控件，该属性的宽度值与高度值决定了滑标的方向。如果宽度值比高度值大，那么滑标是水平向的；如果宽度值比高度值小，那么滑标是垂直向的。
- Units 属性　　Units 属性用于决定控件大小、控件与图形窗口距离等的单位，在 Position 属性中的距离单位就由该属性决定。该属性可以取以下值：pixels，normalized，inches，points centimeters，characters 等值。其中，pixels 是默认属性值。所有的单位都假设图形窗口的左下角为起点。其中，normalized 假设图形窗口左下角为（0,0），右上角为(1,1)。Pixels，inches，centimeters，points(1 point = 1/72 inch)是绝对单位。Character 是应用于字符的单位，一个字符的宽度是字母“x”的宽度，字符的高度是两行文本基线之间的距离。

  如果使用 units 属性，在完成自己的计算后，最好返回 units 属性的默认值，以便不要影响那些假定 units 属性为默认值的函数。

（4）字体控制属性

- FontAngle 属性　　FontAngle 属性的取值可以是 normal，italic，oblique。其中，normal 是默认值。该属性用于确定字符的倾向。MATLAB 使用该属性从用户计算机的系统中选择一个可用的字体，设置该属性为 italic，oblique 时，如果系统中有此类字体，那么就从系统中选择了一种倾斜字体。
- FontName 属性　　FontName 属性的取值是一个字符串。该属性用于设置字体的名字，表明字体的类型。为了能正常地显示与打印，使用该属性设置的字体必须是用户计算机的系统支持的字体类型。默认的属性值是系统的默认字体。例如，为了使用 fixed-width 字体，设置如下：

```
set(uicontrol_handle, 'FontName', 'FixedWidth')
```

  通过设置根的 FixedWidthFontName 属性为合适的值，终端用户在自己的个人环境中，可以使 MATLAB 采用特定的字体。设置了根的 FixedWidthFontName 属性的属性

值，MATLAB 立即就会采用新设的字体。

- FontSize 属性　FontSize 属性的取值是一个数值。该属性用于确定字体的大小。字体大小的单位由属性 FontUnits 决定。该属性的默认值是系统默认的字体大小。
- FontUnits 属性　FontUnits 属性的取值可以是：points，normalized，inches，centimeters，pixels。其中，points 是默认值。该属性用于确定字体大小的单位，设置的属性值影响 FontSize 属性。normalized 设置 FontSize 属性的属性值为控件大小的一部分。当控件大小改变时，MATLAB 相应地改变 FontSize 属性值的大小。Pixels，inches，centimeters，points(1 point = 1/72 inch)是绝对单位。
- FontWeight 属性　FontWeight 属性的取值可以是：light，normal，demi，bold。其中，normal 是默认值。该属性定义字体的粗细。MATLAB 使用该属性从用户计算机的系统中选择一种可用的字体。若设置该属性值为 bold，MATLAB 就从系统中选用一种 bold 版本的字体。
- HorizontalAlignment 属性　HorizontalAlignment 属性的值可取：left，center，right。其中，center 是默认值。该属性用于定义控件上显示的字符（String 属性的属性值）的调整方式。如果取值为 left，那么控件上显示的字符相对于控件左对齐；如果取值为 center，那么控件上显示的字符相对于控件居中；如果取值为 right，那么控件上显示的字符相对于控件右对齐。在 WINDOWS 系统中，该属性仅影响可编辑文本框与静态文本框。

（5）Callback 例程执行控制属性

- BusyAction 属性　BusyAction 属性的取值是 cancel 或 queue，默认值是 queue。该属性用于决定执行控件对象的 callback 例程的中断调用方式。如果有一个 callback 例程正在执行，那么当用户使用触发器（如鼠标单击）触发正在执行 callback 例程的控件对象的另一个 callback 例程时，随后的 callback 例程总是试图中断先前正在执行的 callback 例程。随后要执行的 callback 例程只有包含 drawnow，figure，getframe，pause，waitfor 等命令时，先前正在运行的 callback 例程才能够被中断。

  如果正在执行 callback 例程的对象的 Interruptible 属性值是 on（on 是 Interruptible 属性的默认值），那么正在执行的 callback 例程被中断。如果 Interruptible 属性值是 off，那么 BusyAction 属性决定 callback 例程的中断调用方式。如果 BusyAction 的属性值是 queue，那么随后要运行的 callback 例程加入事件队列中，正在执行当前 callback 例程的事件执行完后，才执行第二个调用 callback 例程的事件。如果 BusyAction 的属性值是 cancel，那么运行第二个 callback 例程的事件被取消。

  需要注意的是，如果调用第二个 callback 例程的事件是 DeleteFcn 或 CreateFcn，或者是图形窗口对象的 CloseRequest 或 ResizeFcn，那么不管 Interruptible 的属性值是什么，都会中断正在运行的 callback 例程，而执行下一个包含 drawnow，figure，getframe，pause，waitfor 命令的 Callback 例程。
- ButtonDownFcn 属性　ButtonDownFcn 属性的取值是一个字符串。该字符串是一个有效的 MATLAB 表达式，或者是一个 M 文件的名字。该属性定义当鼠标在控件对象上按下时执行的 Callback 例程。只要在距离控件对象 5 个像素的范围内按下鼠标键，就会执行该属性定义的 Callback 例程。

  当控件对象的 Enable 属性设置为 inactive 或者 off 时，在距离控件对象 5 个像素

的范围内按下鼠标键时，仍然会执行该属性定义的 Callback 例程。这对于交互式修改控件对象的属性值时很有用。例如，交互式修改 position 属性的属性值。

- Callback 属性　　Callback 属性的取值是一个字符串，该属性定义控件对象的控制动作。当单击控件对象时，就执行 callback 例程。定义的字符串是一个有效的 MATLAB 表达式，或者是一个 M 文件的名字。字符串在 MATLAB 的命令窗口内执行。

  为了执行可编辑文本框的 Callback 属性，当键入一些字符串后，采取以下方式可以执行控件对象的 Callback 属性：把输入焦点从控件对象上移走（可以在界面上别的地方单击鼠标）；然后，对于只能输入单行文本的可编辑文本框，按 Return 键；对于可输入多行文本的可编辑文本框，按 Ctrl+Return 键。

  对于 Frame 与 Static text 类型的控件，不会执行 Callback 属性。

- CreateFcn 属性　　CreateFcn 属性的取值是一个字符串。该字符串是一个有效的 MATLAB 表达式，或者是一个 M 文件的名字。该属性定义在控件对象的创建阶段执行的 Callback 例程。对控件对象，必须定义该属性为控件的默认值。例如：

```
et(0,'DefaultUicontrolCreateFcn',…
    'set(gcf,"IntegerHandle","off")')
```

  上述命令在根对象层定义控件对象的默认值，设置了图形对象的 IntegerHandle 属性值为 off。改变该属性的属性值，对已经存在的控件对象没有影响。MATLAB 在创建控件对象时，先设置了所有属性的属性值后，再执行该属性定义的 Callback 例程。

  对于正在执行 CreateFcn 属性的对象的句柄，只有通过根对象的 CallbackObject 属性，才能访问。根对象的 CallbackObject 属性可以通过函数 gcbo 获得。

- DeleteFcn 属性　　DeleteFcn 属性的取值是一个字符串。该字符串是一个有效的 MATLAB 表达式，或者是一个 M 文件的名字。该属性定义了在控件对象的删除阶段（例如，使用 delete 函数删除控件对象，或包含控件对象的图形窗口对象重新设置时）执行的 Callback 例程。MATLAB 在删除对象的所有属性前执行 DeleteFcn 属性定义的 Callback 例程，所以对象的所有属性值在该 Callback 例程中还是可用的。

  对于正在执行 DeleteFcn 属性的对象的句柄，只有通过根对象的 CallbackObject 属性，才能访问。根对象的 CallbackObject 属性可以通过函数 gcbo 获得。

- Interruptible 属性　　Interruptible 属性的取值是 on 或 off，默认值是 on。该属性决定 Callback 例程的中断调用模式。如果有一个 callback 例程正在执行，那么当用户使用触发器（如鼠标单击）触发正在执行 callback 例程的对象的另一个 callback 例程时，随后的 callback 总是试图中断先前正在执行的 callback 例程。随后要执行的 callback 例程只有包含 drawnow，figure，getframe，pause，waitfor 等命令时，先前正在运行的 callback 调用才能够被中断。

  MATLAB 依据下列因素执行随后的 Callback 例程 :如果正在执行 Callback 例程的对象的 Interruptible 属性是 on，那么正在执行的 Callback 例程能够被中断，执行下一个包含 drawnow，figure，getframe，pause，waitfor 等命令的 Callback 例程。如果 Interruptible 属性值是 off，那么 BusyAction 属性决定 callback 例程的中断调用方式。

  需要注意的是，如果第二个 callback 是 DeleteFcn 或 CreateFcn，或者是图形窗口对象的 CloseRequest 或 ResizeFcn，那么不管 Interruptible 的属性值是什么，都会中断正在运行的 callback 例程，而执行下一个包含 drawnow，figure，getframe，pause，waitfor

等命令的 Callback 例程。此时，按照上述规则，图形窗口对象的 WindowButtonDownFcn 属性的 Callback 例程，控件对象的 ButtonDownFcn 属性的 Callback 例程被执行。

- UIContextMenu 属性　　UIContextMenu 属性的取值是一个 context menu 菜单的对象句柄。通过该属性，某个 context menu 菜单对象就与控件联系起来。当鼠标右键单击控件对象时，MATLAB 就会显示 context menu 菜单。context menu 菜单可以通过函数 uicontextmenu 来创建。

（6）当前状态信息属性

- ListboxTop 属性　　ListboxTop 属性的取值是一个标量，默认值是 1。该属性只应用于列表框控件，用于确定显示在列表框最上头的字符串的索引号。ListboxTop 属性是 String 属性定义的字符串向量的某个元素的索引值，取值必须在 1 与字符串向量的长度之间。
- Max 属性　　Max 属性的取值是一个标量，该属性定义的是 Value 属性允许的最大值。在不同的控件类型中，该属性的意义不同。在复选框中，当复选框被选中时，复选框的 Value 属性值即为该属性值。在可编辑文本框中，如果 Max-Min>1，那么可编辑文本框可以进行多行输入；如果 Max-Min<=1，那么可编辑文本框只能进行单行输入。在列表框中，如果 Max-Min>1，那么列表框允许进行多个列表项的选择；如果 Max-Min<=1，那么列表框不允许进行多个列表项的选择，只能单选。在单选按钮中，当单选按钮被选中时，单选按钮的 Value 属性值即为该属性值。在滑标控件中，该属性值定义了滑标的最大取值，并且，该属性值必须比 Min 属性值大，默认为 1。在开关按钮中，当开关按钮被选中时，开关按钮的 Value 属性值即为该属性值，默认为 1。对于 Frames，pop-up menus，push buttons 和 static text 类型的控件对象，没有 Max 属性。
- Min 属性　　Min 属性的取值是一个标量，该属性定义的是 Value 属性允许的最小值。在不同的控件类型中，该属性的意义不同。在复选框中，当复选框没有被选中时，复选框的 Value 属性值即为该属性值。在可编辑文本框中，如果 Max-Min>1，那么可编辑文本框可以进行多行输入；如果 Max-Min<=1，那么可编辑文本框只能进行单行输入。在列表框中，如果 Max-Min>1，那么列表框允许进行多个列表项的选择；如果 Max-Min<=1，那么列表框不允许进行多个列表项的选择，只能单选。在单选按钮中，当单选按钮没有被选中时，单选按钮的 Value 属性值即为该属性值。在滑标控件中，该属性值定义了滑标的最小值，并且该属性值必须比 Max 属性值小，默认值为 0。在开关按钮中，当开关按钮没有被选中时，开关按钮的 Value 属性值即为该属性值，默认值为 0。对于 Frames，pop-up menus，push buttons 和 static text 类型的控件对象，没有 Min 属性。
- Value 属性　　Value 属性的取值是一个标量或者向量，该属性决定控件的当前值。在不同的控件类型中，该属性的意义不同。在复选框中，当复选框被选中时，该属性的值为 Max 属性值，当没有被选中时，该属性的值为 Min 属性值。在列表框中，设置该属性为向量形式，表明已经选中多个列表项，1 表示是列表框中的第一个列表项。弹出式控件 Pop-up menus 设置该属性值为已经选中的列表项的索引值，1 相对于控件对象中的第一个列表项。在单选按钮中，当单选按钮被选中时，该属性的值为 Max 属性值，当没有被选中时，该属性的值为 Min 属性值。在滑标控件中，设置该属性值为滑

槽内的指示条的当前值。在开关按钮中，当开关按钮被选中时，该属性的值为 Max 属性值，当没有被选中时，该属性的值为 Min 属性值。对于 Editable text，Frames，push buttons 和 static text 类型的控件对象，没有 Value 属性。

可以用鼠标通过交互方式来设置 Value 属性值，也可以调用 set 函数来设置该属性。

（7）控制操作属性

- HandleVisibility 属性　　HandleVisibility 属性的取值是 on，callback，off。其中，on 是默认值。该属性定义控件对象句柄被命令行用户和 GUI 用户可访问的权限。通过该属性，可以决定对象句柄是否在父对象 Children 属性的属性值向量中可见。如果 HandleVisibility 属性值是 on，那么对象句柄一直是可见的。如果 HandleVisibility 属性值是 callback，那么在该 callback 例程或引起 callback 例程调用的函数中是可见的，但在命令行调用的函数中不可见，这对于保护 GUI 用户免受命令行用户的影响很有用。HandleVisibility 属性对于阻止命令行用户任意利用该对象句柄画图或利用该对象句柄删除该对象很有用。如果 HandleVisibility 属性值是 off，那么对象句柄一直是不可见的。这在一个 callback 例程调用一个可能损坏 GUI 的函数时，可以避免对象受不必要的影响。当对象句柄不可见时，在父对象 Children 属性的属性值向量中见不到该句柄，也不能被查找对象句柄或者句柄属性的函数获得。这些函数包括 get，findobj，gca，gcf，gco，newplot，cla，clf，close 等。

  当控件对象的 HandleVisibility 属性值设置为 callback 或 off 时，对象句柄在父对象 Children 属性值向量中不可见；在根对象的 CurrentFigure 属性中不会显示图形；在根对象的 CallbackObject 属性或者在图形窗口对象的 CurrentObject 属性中不会显示对象；在父对象的 CurrentAxes 属性中不会显示坐标轴。

  可以通过设置根对象的 ShowHiddenHandles 属性为 on，来查看所有对象的句柄，而不管对象的 HandleVisibility 属性设置为何值。这对 HandleVisibility 的属性值不会有影响。

  当对象的 HandleVisibility 属性被设置为隐藏时，对象句柄仍然有效。如果知道对象的句柄，可以通过把对象句柄传递给操作句柄的函数来改变对象的属性。

8.4.4　控件属性设置

在命令行方式下或者 GUI 设计方式下，可以方便地设置、修改控件属性的属性值。在命令行方式下，利用函数 uicontrol 建立控件时，可以定义控件属性的属性值；利用函数 set 可以设置、改变属性的属性值；利用函数 get 可以获得属性的属性值。在 GUI 设计方式下，也可以通过 Property Inspector GUI 设计工具来设置控件的属性值。

下例所示，在当前图形窗口内画一条曲线，然后在图形窗口的[80 80 250 65]位置创建一个静态文本框，文本内的字符串由属性 string 决定。在图形窗口对象的右上角输出字符串 str2。

```
str1(1) = {'Center each line in the Uicontrol'};
str1(2) = {'Also check out the textwrap function'};
str2(1) = {'Each cell is a quoted string'};
str2(2) = {'You can specify how the string is aligned'};
str2(3) = {'You can use LaTeX symbols like \pi \chi \Xi'};
str2(4) = {'\bfOr use bold \rm\itor italic font\rm'};
```

```
str2(5) = {'\fontname{courier}Or even change fonts'};
plot(0:6,sin(0:6))
uicontrol('Style','text','Position',[80 80 250 65],…
          'String',str1);
text(5.75,sin(2.5),str2,'HorizontalAlignment','right')
```

创建结果如图 8-21 **所示。**

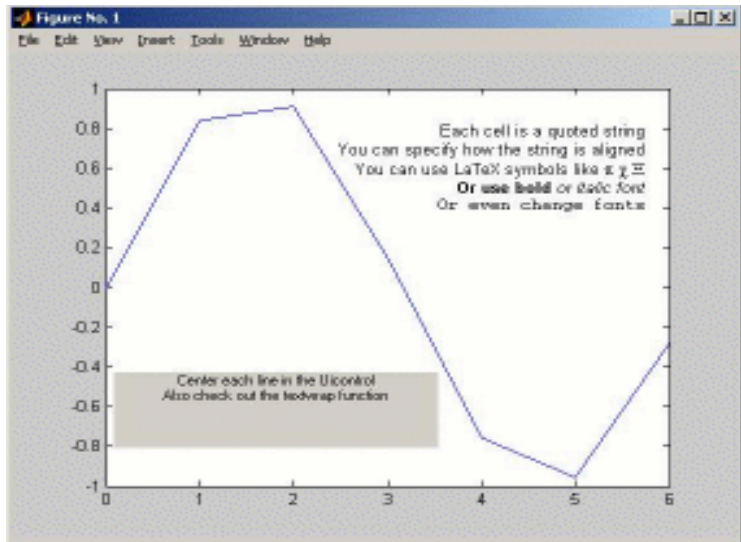


图 8-21　**在图形窗口内创建静态文本控件**

下例所示，创建一个命令按钮。创建的命令按钮的位置在[0.5　0.5　0.2　0.1]**，单位是** normalized**。单击该命令按钮，按钮的位置会随机变动。**

```
b = uicontrol('Style','pushbutton',…
        'Units','normalized',…
        'Position',[.5 .5 .2 .1],…
'String','click here','Callback','
set(b,"Position",[.8*rand .9*rand .2 .1])');
```

下例是在图形窗口对象内创建一个可编辑文本框与滑标控件。当移动滑标控件的位置时，滑标的当前值会显示在可编辑文本框内；当在可编辑文本框内输入滑标 Max **属性值与** Min **属性值之间的一个有效值时，滑标会随着移动到那个值的位置。如下所示是** Callback **例程** Slider **的** M **函数：**

```
function varargout = Slider(varargin)
fig = openfig(mfilename,'reuse');
handles = guihandles(fig);
handles.errorString = 'Total number of errors: ';
handles.numberOfErrors = 0;
guidata(fig,handles);
% 设置可编辑文本框内的值为滑标控件的当前值
set(handles.edit1,'String',…
    num2str(get(handles.slider1,'Value')));
```

```
val = str2num(get(handles.edit1,'String'));

if isnumeric(val) & length(val)==1 &…
      val >= get(handles.slider1,'Min') &…
      val <= get(handles.slider1,'Max')
 % 设置滑标的 Value 值为可编辑文本框内的值
    set(handles.slider1,'Value',val);
else
% 错误记录
      handles.numberOfErrors = handles.numberOfErrors+1;
      set(handles.edit1,'String',…
      [handles.errorString,num2str(handles.numberOfErrors)]);
      guidata(gcbo,handles); % store the changes…
end
```

在本例中，设置滑标的 Max **属性值等于** 1，Min **属性值等于** 0。**运行上述函数，可以看到图** 8-22 **所示的结果。**

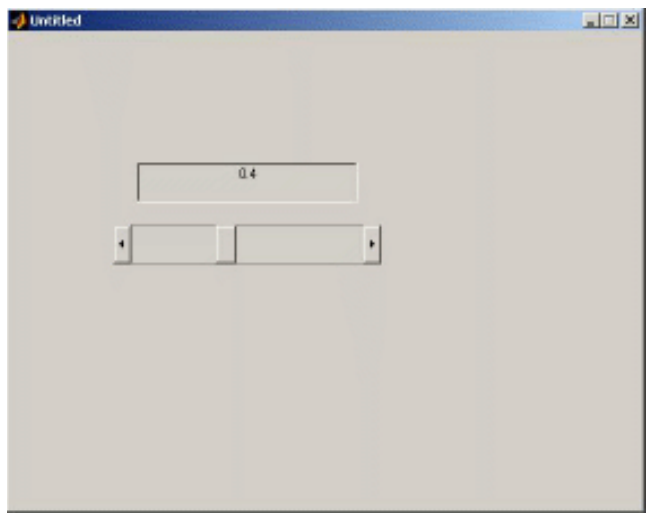


图 8-22　**滑标与可编辑文本框结合的** GUI **界面**

8.5　对话框

在 GUI **程序设计中，对话框是最重要的显示信息和取得用户数据的用户界面对象。对话框一般包含一个或多个按钮以供用户输入；或者弹出显示的信息，由用户决定要采取什么措施。使用对话框，可以使图形用户界面更加友好，易于为用户理解。在绝大部分的程序设计软件中，如** Visual Basic，Dephi，Visual C++，**都可以方便地进行对话框设计。**MATLAB **也提供了几个进行对话框设计的有用函数。总的说来，**MATLAB **中的对话框分为两大类：第一类是公共对话框；第二类是一般对话框，其中包括请求对话框。下面对** MATLAB **中的对话框进行介绍。**

8.5.1 公共对话框

MATLAB 与 Visual C++等程序设计软件一样，拥有大量标准的公共对话框。它们是：文件打开对话框 uigetfile，文件保存对话框 uiputfile，颜色设置对话框 uisetcolor，字体设置对话框 uisetfont，打印页面设置对话框 pagesetupdlg 与 pagedlg，打印预览对话框 printpreview 以及打印对话框 printdlg。下面分别对它们进行介绍。

1．文件打开对话框

文件打开对话框用于打开某个文件。在 WINDOWS 系统中，几乎所有的应用软件都提供了文件打开对话框。MATLAB 中，调用文件打开对话框的函数为 uigetfile。该函数的调用格式有如下几种：

```
uigetfile
uigetfile('FilterSpec')
uigetfile('FilterSpec','DialogTitle')
uigetfile('FilterSpec','DialogTitle',x,y)
[fname,pname] = uigetfile(…)
```

其中，uigetfile 调用格式显示的文件打开对话框中，列出当前目录下 MATLAB 能认识的所有文件。

uigetfile('FilterSpec')调用格式显示的文件打开对话框中，列出当前目录下的由参数 FilterSpec 指定的类型文件。参数 FilterSpec 是一个文件类型过滤字符串，用于指定要显示的文件类型。例如，'*.m'（这是默认值）显示当前目录下，所有后缀为 m 的文件，即显示 MATLAB 中的 M 文件。

uigetfile('FilterSpec','DialogTitle')调用格式与第二种调用格式 uigetfile('FilterSpec')基本相同，只是该调用格式设定了文件打开对话框的标题名。文件打开对话框的默认标题名是字符串“Select file to open”。

uigetfile('FilterSpec','DialogTitle',x,y) 调用格式与第三种调用格式 uigetfile('FilterSpec','DialogTitle')基本相同，只是该调用格式还指定了对话框显示时的位置。显示的位置由参数 *x* 与 *y* 决定。*x*，*y* 是从屏幕左上角算起的水平与垂直距离。距离的单位是像素。在有些操作系统中，可能不支持对话框的位置参数的设置。

[fname,pname] = uigetfile(…) 调用格式返回了打开的文件的文件名与路径。如果按了文件打开对话框中的 OK 按钮，那么就会返回文件名与路径。其中，输出参数 fname 存放的是打开的文件名，pname 包含文件的路径。如果按了文件打开对话框中的 Cancel 按钮，或者打开文件时有出错信息出现，那么输出参数 fname 与 pname 的返回值都是零。该调用方式的输入参数与前面的调用方式中的输入参数一样。

如果打开文件时，所要打开的文件不存在，或者有出错信息出现，此时可以选择打开另外一个文件或者单击 Cancel 按钮，取消打开文件。

请看下面的例子：

```
[fname,pname] = uigetfile('*.m','Open the M File')
```

在该例中，列出当前目录下的所有 M 文件，打开文件对话框的标题名为“Open the M File”。打开的文件的文件名与路径返回到输出参数 fname 与 pname 中。显示的文件打开对话框的外观依赖于不同的操作系统。

2．文件保存对话框

文件保存对话框用于保存某个文件。MATLAB 中，调用文件保存对话框的函数为 uiputfile。该函数的调用格式有如下几种：

```
uiputfile
uiputfile('InitFile')
uiputfile('InitFile','DialogTitle')
uiputfile('InitFile','DialogTitle',x,y)
[fname,pname] = uiputfile(…)
```

其中，uiputfile 调用格式显示用于保存文件的对话框，对话框中列出了当前目录下的所有文件。

uiputfile('InitFile')调用格式显示的文件保存对话框中，列出当前目录下的由参数 InitFile 指定的类型文件。参数 InitFile 是一个文件类型过滤字符串，用于指定要保存的文件类型。例如，'*.m'（这是默认值）显示当前目录下所有后缀为 m 的文件，即显示 MATLAB 中的 M 文件。

uiputfile('InitFile','DialogTitle')调用格式与第二种调用格式 uiputfile('InitFile')基本相同，只是该调用格式设定了文件保存对话框的标题名。文件保存对话框的默认标题名是字符串“Select file to Write”。

uiputfile('InitFile','DialogTitle',x,y) 调用格式与第三种调用格式 uiputfile('InitFile','DialogTitle')基本相同，只是该调用格式还指定了对话框显示时的位置。显示的位置由参数 x 与 y 决定。x，y 是从屏幕左上角算起的水平与垂直距离。距离的单位是像素。在有些操作系统中，可能不支持对话框位置参数的设置。

[fname,pname] = uiputfile(…)调用格式返回了保存文件的文件名与路径。如果按了文件保存对话框中的 OK 按钮，那么就会返回文件名与路径。其中，输出参数 fname 存放的是保存的文件名，pname 包含文件的路径。如果按了文件保存对话框中的 Cancel 按钮，或者保存文件时有出错信息出现，那么输出参数 fname 与 pname 的返回值都是零。该调用方式的输入参数与前面的调用方式中的输入参数一样。

如果保存文件时，所要保存的文件已经存在，就会显示一个信息对话框，询问是否覆盖已经存在的文件。如果选择了 OK，那么新文件会成功覆盖原来已存在的文件；如果选择了 Cancel，那么控制焦点又会返回到保存文件对话框，等待用户采取进一步的措施。

请看下面的例子：

```
[newfile,newpath] = uiputfile('animinit.m','Save file name')
```

在该例中，准备把文件 animinit.m 保存到当前目录下。文件保存对话框中列出了当前目录下的所有文件，保存文件对话框的标题名为“Save file name”。保存的文件的文件名与路径返回到输出参数 newfile 与 newpath 中。显示的文件保存对话框的外观依赖于不同的操作系统。

3．颜色设置对话框

颜色设置对话框可用于交互式设置某个图形对象的前景或背景颜色等。在绝大部分的程序设计软件中，都提供了这个公共对话框。在 MATLAB 中，调用颜色设置对话框的函数为 uisetcolor。该函数的调用格式如下：

```
c = uisetcolor(h_or_c, 'DialogTitle');
```

输入参数中的 h_or_c 可以是一个图形对象的句柄，也可以是一个 3 色 RGB 向量。如果

是图形对象的句柄，那么该图形对象必须有一个颜色属性；如果是 3 色 RGB 向量，那么输入的必须是一个有效的 RGB 向量（例如，[1 0 0]表示红色），此时输入的颜色初始化颜色对话框。如输入的参数是图形对象句柄，颜色对话框被初始化为黑色。输入参数中的“Dialog Title”字符串用于标明颜色对话框的标题名。输出参数 c 返回用户选择的 RGB 向量值。如果用户在颜色对话框上单击 Cancel，或者设置颜色时有错误出现，那么如果输入参数 h_or_c 设置了 RGB 颜色值，输出参数 c 的值为输入参数的 RGB 值。如果输入参数 h_or_c 没有设置 RGB 颜色值，此时输出参数 c 的值为零。

比如，在 MATLAB 命令窗口内输入：

```
c=uisetcolor([0 1 0],'Select Color for Graphics Object')
```

返回的颜色设置对话框如图 8-23 所示。

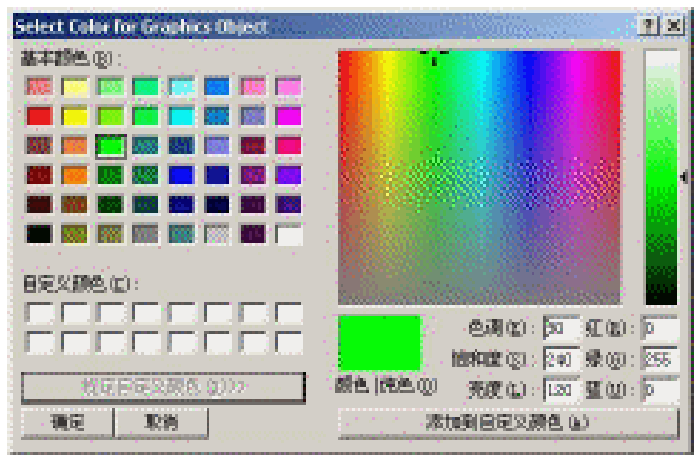


图 8-23　颜色设置对话框

4．字体设置对话框

字体设置对话框可用于交互式修改文本字符串、坐标轴或控件对象的字体属性。可以修改的字体属性包括 FontName，FontUnits，FontSize，FontWeight，FontAngle 等。MATLAB 中，调用字体设置对话框的函数为 uisetfont。该函数的调用格式如下：

```
uisetfont
uisetfont('DialogTitle')
uisetfont(h)
uisetfont(S)
uisetfont(h,'DialogTitle')
uisetfont(S,'DialogTitle')
S = uisetfont(…)
```

其中，uisetfont 调用格式显示用于进行字体设置的对话框，对话框中列出了字体、字形、字体大小等字段，返回的是选择的字体的属性值。

uisetfont('DialogTitle')调用格式与 uisetfont 调用格式基本相同，只是该调用格式设定了字体设置对话框的标题名。字体设置对话框的默认标题名是字符串“Font”。

uisetfont(h)调用格式中的输入参数 h 是一个对象句柄。该调用格式用对象句柄中的字体属性值初始化字体设置对话框中的属性值，用户可以利用字体设置对话框重新设置对象的字

体属性值，返回重新设置后的字体的属性值。

uisetfont(h,'DialogTitle')调用格式与 uisetfont(h)的调用格式基本相同。只是该调用格式设定了字体设置对话框的标题名。字体设置对话框的默认标题名是字符串“Font”。

uisetfont(S)调用格式中的输入参数 S 是一个字体属性结构。该调用格式用字体属性结构 S 中的成员值来初始化字体设置对话框中的属性值。输入参数 S 必须是一个包含一个或多个下列属性的合法值：FontName，FontUnits，FontSize，FontWeight，FontAngle。如果在 S 中输入别的属性的属性值，该属性值会被忽略。用户可以利用字体设置对话框重新设置对象的字体属性值，返回重新设置后的字体的属性值。

uisetfont(S,'DialogTitle') 调用格式与 uisetfont(S)的调用格式基本相同。只是该调用格式还设定了字体设置对话框的标题名。字体设置对话框的默认标题名是字符串“Font”。

S = uisetfont(…)调用格式返回字体属性（FontName，FontUnits，FontSize，FontWeight，FontAngle）的属性值，被保存在结构 S 中。如果用户按了字体设置对话框中的 Cancel 按钮，或者设置字体时有出错信息出现，那么输出参数 S 的返回值为零。该调用方式的输入参数与前面的调用格式中的输入参数一样。

请看下面的例子：

```
h = text(.5,.5,'Figure Annotation');
uisetfont(h,'Update Font')
```

在该例中，创建了一个文本对象，然后显示改变文本对象字体的对话框，使用户可以对字符串“Figure Annotation”进行字体的设置。

下例中，首先创建了两个命令按钮控件，然后先对其中的一个命令按钮进行字体设置，并利用该字体设置返回的属性值（保存在结构 d 中）再进行第二个命令按钮的字体设置。

```
c1 = uicontrol('Style', 'pushbutton',…
    'Position', [10 10 100 20], 'String', 'ABC');
c2 = uicontrol('Style', 'pushbutton',…
    'Position', [10 50 100 20], 'String', 'XYZ');
d = uisetfont(c1)
set(c2, d)
```

5．打印页面设置对话框

打印页面设置对话框可用于对打印输出时的页面进行设置。在许多应用软件中，都提供了进行打印页面设置的对话框。MATLAB 中，进行打印页面设置对话框的函数为 pagesetupdlg。该函数的调用格式如下：

```
dlg = pagesetupdlg(fig)
```

该调用格式用默认的页面设置属性为图形窗口对象创建一个打印页面设置对话框。单击图形窗口的 File 菜单的 Page Setup…子菜单也可以显示出该对话框。与下面将要介绍的 pagedlg 对话框不同的是，pagesetupdlg 函数只支持单个图形窗口对象的页面设置。输入参数 fig 必须是单个图形窗口的句柄，而不是一个图形窗口向量。如果省略参数 fig，那么默认的图形窗口对象是当前窗口对象，输出参数返回的是设置了的打印页面属性值。

下面的例子是对当前图形窗口对象进行打印页面的设置。

```
dlg=pagesetupdlg
```

打印页面设置对话框如图 8-24 所示。

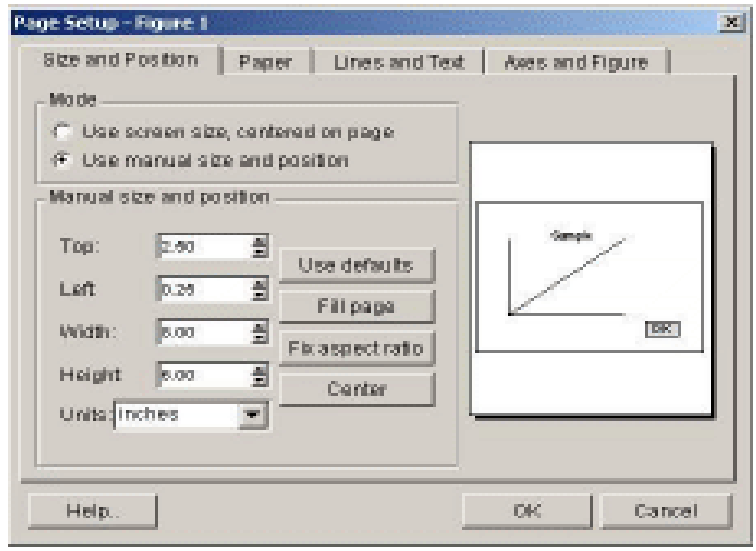


图 8-24　打印页面设置对话框

在 MATLAB 中，还提供了一个进行打印页面设置的对话框函数，该函数名为 pagedlg。这个函数是以前的 MATLAB 版本进行打印页面设置的对话框函数，现在已很少使用。为了使 MATLAB 的老用户使用方便，这里也对它进行介绍。该函数的调用格式如下：

pagedlg

pagedlg(fig)

pagedlg **调用格式显示一个打印页面设置对话框，用默认的页面设置属性对当前图形窗口对象进行打印页面设置。**pagedlg(fig)**调用格式用默认的页面设置属性对指定的图形窗口对象进行打印页面设置，输入参数是指定的图形窗口对象的句柄。**pagedlg **可进行多个图形窗口对象的页面设置。**

图 8-25 所示为函数 pagedlg **调用的打印页面设置对话框。**

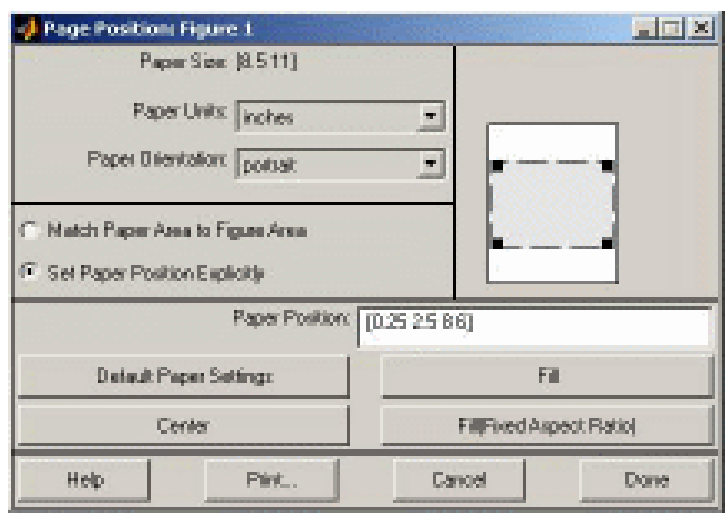


图 8-25　**pagedlg **函数调用的打印页面设置对话框

6．打印预览对话框

打印预览对话框用于对打印输出的页面进行预览。在许多应用软件中，都提供了进行打印预览设置的对话框。MATLAB 中，进行打印预览设置对话框的函数为 printpreview。该函数的调用格式如下：

```
printpreview
printpreview (fig)
```

printpreview 调用格式显示当前图形窗口对象的打印预览对话框。打印预览对话框显示了将要打印的图形在页面上的尺寸与大小。printpreview (fig) 调用格式显示指定的图形窗口对象 fig 的打印预览对话框。

单击图形窗口 File 菜单的 Print Preview…子菜单也可以显示出打印预览对话框。

在打印预览对话框的上面，有 4 个按钮，分别是：Print，Page Setup，Zoom In，Zoom Out。现对它们的作用说明如下：

Print 按钮：单击该按钮，就会显示打印对话框。通过它就可以立即打印出图形。

Page Print 按钮：单击该按钮，就会显示出打印页面设置对话框。在这里，可以对将要打印的页面进行设置，这些设置直接影响要打印的图形。

Zoom In 按钮：单击该按钮，就会放大预览页面。此时，如果看不到页面中的某些图形，可以通过上下、左右滚动条来进行查看。

Zoom Out 按钮：单击该按钮，就会缩小预览页面，把页面恢复成初始状态。

图 8-26 所示就是对当前图形窗口内的图形进行预览。

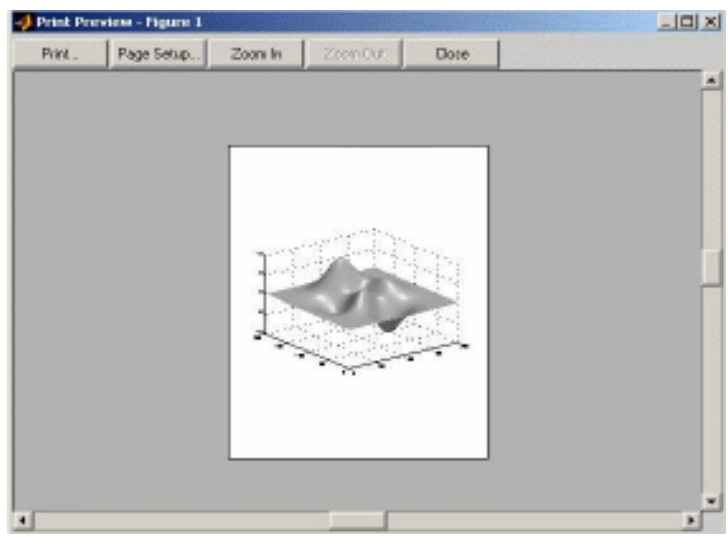


图 8-26　打印预览对话框

7．打印对话框

打印对话框是专门进行打印用的对话框。这是任何图形用户界面的应用软件都会提供的对话框。MATLAB 中，调用打印对话框的函数为 printdlg。该函数的调用格式如下：

```
printdlg
printdlg(fig)
printdlg('-crossplatform',fig)
```

printdlg('-setup',fig)

其中，printdlg **调用格式显示出标准的** WINDOWS **打印对话框（在** WINDOWS **系统中）。它打印当前图形窗口内的图形对象。单击图形窗口的** File **菜单的** Print**…子菜单也可以显示出打印对话框。**

printdlg(fig)**调用格式也显示出标准的** WINDOWS **打印对话框，但它打印由输入参数** fig **指定的图形窗口内的对象。输入参数** fig **是将要打印的图形窗口的句柄。**

printdlg('-crossplatform',fig)**调用格式显示标准的** crossplatform **模式的** MATLAB **打印对话框。输入参数** fig **是将要打印的图形窗口的句柄。**

printdlg('-setup',fig)**调用格式显示打印对话框的开始模式。在这里可以不进行打印而设置打印的默认选项。单击图形窗口的** File **菜单的** Print setup**…子菜单也可以显示出打印开始对话框。**

图 8-27 **显示的是** Cross-platform **模式的打印对话框。**

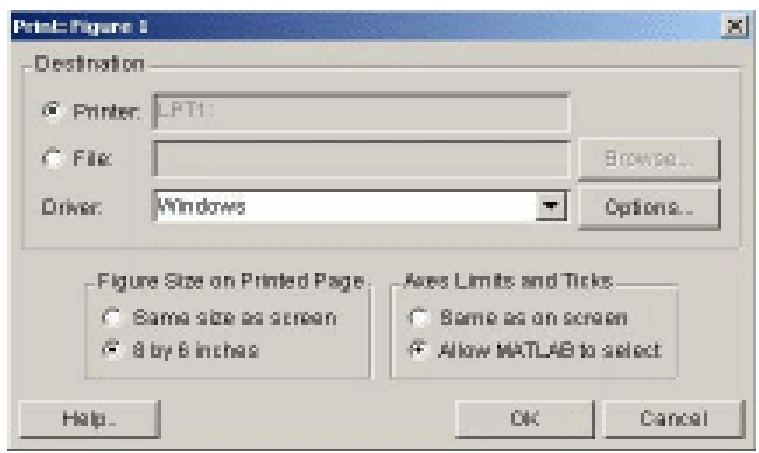


图 8-27　Cross-platform **模式的打印对话框**

图 8-28 **显示的是开始模式的打印对话框。**

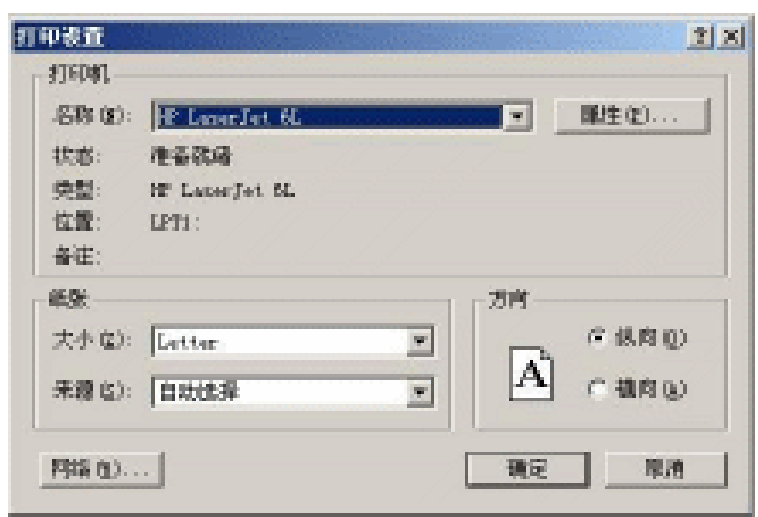


图 8-28　**开始模式的打印对话框**

8.5.2 一般对话框

除了提供大量标准的公共对话框外，MATLAB 还提供了大量的一般对话框与请求对话框。MATLAB 提供的一般对话框与请求对话框有：帮助对话框 helpdlg，出错信息显示对话框 errordlg，信息提示对话框 msgbox，问题显示对话框 questdlg，警告信息显示对话框 warndlg，变量输入对话框 inputdlg，列表选择对话框 listdlg 等。下面分别对它们进行介绍。

1．帮助对话框

在操作应用软件时，当用户不知道该如何操作时，此时如果有帮助信息，那么它就会帮助用户进行正确地操作。显示帮助信息的对话框就是帮助对话框。MATLAB 提供的创建帮助对话框的函数是 helpdlg。该函数的调用格式如下：

```
helpdlg
helpdlg('helpstring')
helpdlg('helpstring','dlgname')
h = helpdlg(…)
```

其中，helpdlg 调用格式创建一个默认的帮助对话框。默认对话框的名字叫“Help Dialog”，在对话框内包含一个名为“This is the default help string”的字符串。

helpdlg('helpstring') 调用格式创建一个帮助对话框。该对话框的名字仍然叫“Help Dialog”，但对话框内显示的帮助信息由输入参数 helpstring 决定。

helpdlg('helpstring','dlgname') 调用格式创建一个帮助对话框。该对话框的名字由输入参数 dlgname 决定，对话框内显示的帮助信息由输入参数 helpstring 决定。

h = helpdlg(…) 调用格式返回创建的帮助对话框的句柄，句柄存放在输出参数 h 中。输入参数与前面的调用格式的输入参数相同。

MATLAB 会自动设置帮助对话框的宽度，使得能够显示出 helpstring 字符串的全部帮助信息。

下面所示的例子中，创建一个名为 Point Selection 的对话框。显示的帮助信息是字符串 Choose 10 points from the figure。

```
>> helpdlg('Choose 10 points from the figure','Point Selection');
```

创建的帮助对话框如图 8-29 所示。

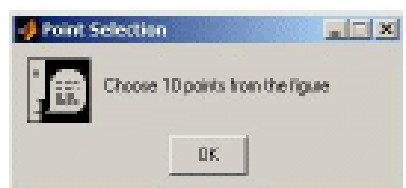


图 8-29 帮助对话框

2．出错信息显示对话框

在开发的应用软件中，当用户进行了错误的操作后，应该显示出错信息对话框，使用户知道出错的原因，以便采取正确的操作。此时，就要用到出错信息显示对话框。MATLAB 提供的创建出错信息显示对话框的函数是 errordlg。该函数的调用格式如下：

```
errordlg
errordlg('errorstring')
```

```
errordlg('errorstring','dlgname')
errordlg('errorstring','dlgname','on')
h = errordlg(…)
```

其中，errordlg 调用格式创建一个默认的出错信息显示对话框。默认对话框的名字叫‘ Error Dialog ”，在对话框内包含一个名为“ This is the default error string ”的字符串。

errordlg('errorstring') 调用格式创建一个出错信息显示对话框。该对话框的名字仍然叫“ Error Dialog ”，但对话框内显示的出错信息由输入参数 errorstring 决定。

errordlg('errorstring','dlgname') 调用格式创建一个出错信息显示对话框。该对话框的名字由输入参数 dlgname 决定，对话框内显示的出错信息由输入参数 errorstring 决定。

errordlg('errorstring','dlgname','on') 调用格式创建一个出错信息显示对话框。当要创建的对话框已经存在时，输入参数 on 把已经存在的对话框显示在屏幕的最前端，而不再创建同名的新对话框。

h = errordlg(…)调用格式返回创建的出错信息显示对话框的句柄，句柄存放在输出参数 h 中。输入参数与前面的调用格式的输入参数相同。

MATLAB 会自动设置出错信息显示对话框的宽度，使其能显示出 errorstring 字符串的全部出错信息。显示的出错信息对话框的外观依赖于不同的操作系统。

下面所示的例子中，创建一个名为 File Error 的对话框。显示的出错信息是字符串“ File not found ”。

```
errordlg('File not found','File Error');
```

创建的出错信息显示对话框如图 8-30 所示。

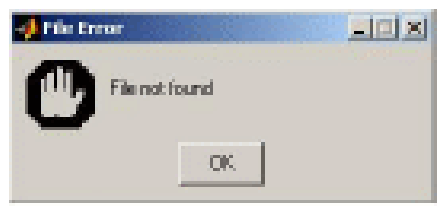


图 8-30　出错信息显示对话框

3．信息提示对话框

当面临多种选择或应该显示某种提示情况时，一般就会显示信息提示。这时，就要借助于信息提示对话框。MATLAB 提供的创建信息提示对话框的函数是 msgbox。该函数的调用格式如下：

```
msgbox(message)
msgbox(message,title)
msgbox(message,title,'icon')
msgbox(message,title,'custom',iconData,iconCmap)
msgbox(…,'createMode')
h = msgbox(…)
```

其中，msgbox(message) 调用格式创建一个信息提示对话框。创建的对话框会自动设置对话框的宽度，以便能显示全部提示信息。输入参数 message 存储的是要显示的提示信息。该参数的取值可以是一个字符串向量或字符串矩阵。

msgbox(message,title) 调用格式创建的信息提示对话框与 msgbox(message)创建的信息指示对话框基本相同，只是该对话框有一个标题名。标题名由输入参数 title 决定。参数 title 是一个字符串。

msgbox(message,title,'icon') 调用格式创建的信息提示对话框除了包含有提示信息与标题名外，对话框上还有一些图标。图标由参数 icon 决定。icon 可选的值有：'none'，'error'，'help'，'warn'，'custom'。默认值是'none'。

msgbox(message,title,'custom',iconData,iconCmap) 调用格式创建的信息提示对话框中的图标是用户自定义的图标。定义图标的图像数据存放在参数 iconData 中，定义图像的颜色数据存放在参数 iconCmap 中。

msgbox(…,'createMode')调用格式中的参数 CreateMode 用于决定创建的信息提示对话框是模式对话框还是无模式对话框。如果是模式对话框，用户关闭对话框后控制焦点才从调用的对话框上返回。如果是无模式对话框，用户可以在打开对话框的情况下，把控制焦点从打开的对话框上移开。参数 CreateMode 的可选值有：'modal'，'non-modal'和'replace'。其中，'replace'值用标题名相同的对话框代替另外一个已经打开的对话框。

h = msgbox(…)调用格式返回创建的信息提示对话框的句柄，句柄存放在输出参数 h 中。输入参数与前面的调用格式的输入参数相同。

下面所示的例子中，创建一个名为 Information 的信息提示对话框。显示的提示信息是字符串“ This is a first Information Dialog Box! ”。对话框上有一个警告的图标。它是一个无模式对话框。

```
msgbox('This is a first Information Dialog Box!','Information','warn','non-modal');
```

创建的信息提示对话框如图 8-31 所示。

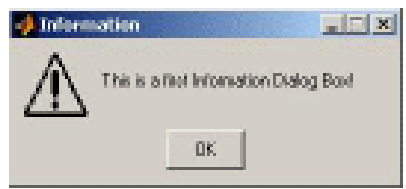


图 8-31 信息提示对话框

4．问题显示对话框

当对问题的解决可能存在多种选择时，就会显示一个问题显示对话框，由用户决定应该采取什么步骤。例如，保存的文件的文件名与当前目录中存在的某个文件名相同时，就会显示问题显示对话框。MATLAB 提供的创建问题显示对话框的函数是 questdlg。该函数的调用格式如下：

```
button = questdlg('qstring')
button = questdlg('qstring','title')
button = questdlg('qstring','title','default')
button = questdlg('qstring','title','str1','str2','default')
button = questdlg('qstring','title','str1','str2','str3','default')
```

其中，button = questdlg('qstring')调用格式创建一个问题显示的模式对话框。该对话框有 3 个命令按钮，分别为 Yes，No，Cancel。显示的问题由输入参数 qstring 决定。参数 qstring 是一个字符串。MATLAB 会自动设置对话框的宽度，以使字符串 qstring 能全部显示。输出

参数 button 返回的是用户按下的命令按钮的名字。

button = questdlg('qstring','title')调用格式创建的问题显示对话框的标题由参数 title 决定。该标题显示在对话框的标题栏上。

button = questdlg('qstring','title','default')调用格式创建的问题显示对话框，当用户按下键盘上的回车键时，返回参数 button 中的值是参数 default 设置的值。Default 必须是'Yes'，'No'，'Cancel'中的一个。

button = questdlg('qstring','title','str1','str2','default')调用格式创建的问题显示对话框有两个命令按钮，命令按钮上显示的字符由参数 str1 与 str2 决定。Default 设置当用户按下键盘上的回车键时返回的参数值。Default 必须是 str1 或 str2 中的一个。

button =questdlg('qstring','title','str1','str2','str3','default') 调用格式创建的问题显示对话框有 3 个命令按钮，命令按钮上显示的字符由参数 str1，str2 与 str3 决定。Default 设置当用户按下键盘上的回车键时返回的参数值。Default 必须是 str1，str2 或 str3 中的一个。

下面的例子中，创建一个问题显示对话框来询问用户是否继续创建文件。如果用户按下 Yes 按钮，返回的是字符串 Creating file；按下 No 按钮，返回的是 Canceled file operation 字符串；按下 Help 按钮，返回的是 Sorry,no help available 字符串。默认的按钮返回值是 No。

```
button = questdlg('Do you want to continue?',…
'Continue Operation','Yes','No','Help','No');
if strcmp(button,'Yes')
    disp('Creating file')
elseif strcmp(button,'No')
    disp('Canceled file operation')
elseif strcmp(button,'Help')
    disp('Sorry, no help available')
end
```

创建的问题显示对话框如图 8-32 所示。

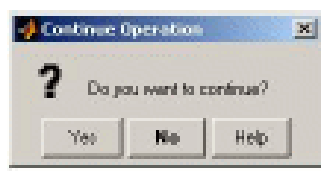


图 8-32　问题显示对话框

5. 警告信息显示对话框

在应用软件中，当用户进行了不恰当的操作后，就会显示警告信息对话框，使用户知道该操作可能导致的错误，以便采取正确的操作。此时，就要用到警告信息显示对话框。MATLAB 提供的创建警告信息显示对话框的函数是 warndlg。该函数的调用格式如下：

```
warndlg
warndlg('warnstring')
warndlg('warnstring','dlgname')
warndlg('warnstring','dlgname','on')
h = warndlg(…)
```

其中，warndlg 调用格式创建一个默认的警告信息显示对话框。默认对话框的名字叫“warning Dialog”，在对话框内包含一个名为“This is the default warning string”的字符串。

warndlg('warnstring') 调用格式创建一个警告信息显示对话框。该对话框的名字仍然叫“warning Dialog”，但对话框内显示的警告信息由输入参数 warnstring 决定。

warndlg('warnstring','dlgname') 调用格式创建一个警告信息显示对话框。该对话框的名字由输入参数 dlgname 决定，对话框内显示的警告信息由输入参数 warnstring 决定。

warndlg('warnstring','dlgname','on') 调用格式创建一个警告信息对话框。当要创建的对话框已经存在时，输入参数 on 把已经存在的对话框显示在屏幕的最前端，而不再创建同名的新对话框。

h =warndlg(…)调用格式返回创建的警告信息显示对话框的句柄，句柄存放在输出参数 h 中。输入参数与前面的调用格式的输入参数相同。

MATLAB 会自动设置警告信息显示对话框的宽度，使得能显示出 warnstring 字符串中的全部警告信息。显示的警告信息对话框的外观依赖于不同的操作系统。

下面所示的例子中，创建一个名为“!!Warning !!”的对话框。显示的警告信息是字符串“Pressing OK will clear memory”。

```
warndlg('Pressing OK will clear memory','!! Warning !!')
```

创建的警告信息对话框如图 8-33 所示。

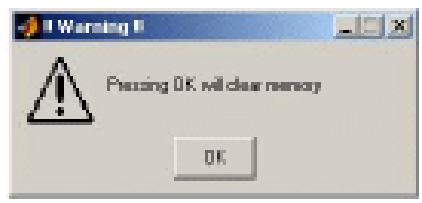


图 8-33　警告信息对话框

6. 变量输入对话框

在许多应用软件中，当需要用户输入变量时，就会显示一个输入对话框，这样就使变量的输入变得非常容易。这个对话框就是变量输入对话框。MATLAB 提供的创建变量输入对话框的函数是 inputdlg。该函数的调用格式如下：

```
answer = inputdlg(prompt)
answer = inputdlg(prompt,title)
answer = inputdlg(prompt,title,lineNo)
answer = inputdlg(prompt,title,lineNo,defAns)
answer = inputdlg(prompt,title,lineNo,defAns,Resize)
```

其中，answer = inputdlg(prompt) 调用格式创建一个模式变量输入对话框。输入参数是提示输入信息的字符串。返回值 answer 存储用户输入的变量值。

answer = inputdlg(prompt,title) 调用格式创建的模式变量输入对话框的标题名由参数 title 决定。该参数是一个字符串。

answer = inputdlg(prompt,title,lineNo) 调用格式创建的模式变量输入对话框中用于输入变量的可编辑文本框的行数由参数 lineNo 决定。参数 lineNo 可以是标量、列向量或矩阵。如果是标量值，在所有的提示输入信息下面的可编辑文本框的行数都是 lineNo；如果是一个列

向量，那么在每个提示输入信息下面的可编辑文本框的行数由向量中相对应的元素决定；如果是一个矩阵，它应该是 $m\times2$，m 表示提示输入信息的个数，矩阵每行指向一个提示输入信息，第一列标明可编辑文本框的行数，第二列标明字符串的宽度。LineNo 的默认值是 1。当有多个可编辑文本框时，用户输入的值都存储在参数 answer 中，只是此时要求 answer 参数是一个向量值。

answer = inputdlg(prompt,title,lineNo,defAns) 调用格式创建的变量输入对话框的每个可编辑文本框中的默认值由参数 defAns 决定。DefAns 的值就显示在每个可编辑文本框中。参数 DefAns 是一个字符向量，其元素的个数必须与参数 prompt 中元素的个数相等。

answer = inputdlg(prompt,title,lineNo,defAns,Resize)调用格式中的输入参数用于决定创建的变量输入对话框的大小能否被调整。如果取值是字符串'on'，那么创建的对话框的大小可以被调整，此时它就不是一个模式对话框；如果取值是'off '，那么创建的对话框的大小不能被调整。

下面的例子中，创建了一个变量输入对话框，可以输入一个整数与一个颜色名。其中，第一个可编辑文本框的行数为 2 行，其默认值为 20；第二个可编辑文本框的行数为 4 行，其默认值为 hsv。

```
prompt   = {'Enter matrix size:','Enter colormap name:'};
title    = 'Input for peaks function';
lines= [2 4]';
def      = {'20','hsv'};
answer   = inputdlg(prompt,title,lines,def);
```

创建的变量输入对话框如图 8-34 所示。

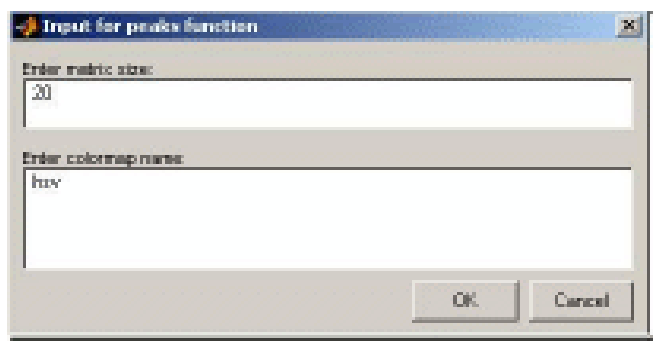


图 8-34　变量输入对话框

7．列表选择对话框

当存在多个选择项时，最好提供给用户一个列表框，把所有可能的选择项都列出来，使用户能从中选择一个自己需要的值。在这种情况下就要用到列表选择对话框。MATLAB 提供的创建列表选择对话框的函数是 listdlg。该函数的调用格式如下：

[Selection,ok] = listdlg('ListString',S,…)

上面的调用格式用来创建一个列表选择模式对话框。从列表框中可以选择一个或多个列表项。输出变量是一个向量，存储的是选择的列表项的索引号。在单模式选择状态下，它的长度是 1。当输出变量 OK 是零时，变量 Selection 是空向量[]。如果用户按了列表选择对话框的 OK 按钮，输出变量 OK 值是 1；如果按了列表选择对话框中的 Cancel 按钮或关闭列表

选择对话框，输出变量 OK 是 0。双击选择的列表项或选中列表项，然后单击对话框的 OK 按钮，其效果是一样的。在多模式选择状态下，按一下对话框中的 Select all 按钮，可以选择所有的列表项。

列表选择对话框可选的输入参数可以选取表 8-2 中所列的值。

表 8-2　列表选择对话框可选的输入参数

参　数	参数功能简介
'ListString'	该参数用于设置列表框中的列表项，是一个字符向量
'SelectionMode'	该参数用于设置是单模式选择还是多模式选择，默认是多模式选择。参数取值是一个字符串，可以取'single'或'multiple'
'ListSize'	该参数用于设置列表框的大小，是一个两元素向量[width height]，默认值是[160 300]，单位是像素
'InitialValue'	该参数用于说明初始选择的列表项的索引，默认是 1，即第一项。参数的取值是一个向量
'Name'	该参数用于设置列表选择对话框的标题名。默认是空字符串
'PromptString'	该参数是一个字符矩阵或字符向量，用于标明在列表框上面的说明性文本。默认是空
'OKString'	该参数用于设置 OK 按钮上的文本。默认是字符串 OK
'CancelString'	该参数用于设置 Cancel 按钮上的文本。默认是字符串 Cancel
'uh'	该参数用于设置控件按钮的高度，默认是 18，单位是像素
'fus'	该参数用于设置框架与控件对象之间的距离。默认是 8，单位是像素
'ffs'	该参数用于设置框架与图形窗口对象之间的距离，默认是 8，单位是像素

下面的例子中，显示一个列出当前目录下的所有文件的列表选择对话框。用户可以从中选择文件。函数返回的是向量。第一个返回值存储用户选择的文件索引号。如果没有选择任何文件，那么第二个输出参数返回 0；如果选择了文件，那么第二个输出参数返回 1。

```
d = dir;
str = {d.name};
[s,v] = listdlg('PromptString','Select a file:',…
                'SelectionMode','single',…
                'ListString',str)
```

创建的列表选择对话框如图 8-35 所示。

图 8-35　列表选择对话框

除了前面介绍的对话框外，MATLAB 还提供了一个用于创建对话框的函数 dialog。通过该函数，用户可以创建任何类型的对话框。在 MATLAB 中，所有的对话框都是基于该函数创建的。它本身不是一个句柄图形对象，而是由一系列句柄图形对象构成的 M 文件。实际上，对话框是由框架、可编辑文本框、按钮等构成的图形窗口对象（关于利用函数 dialog 创建对话框，可以参考 MATLAB 的帮助）。

8.6　GUI 的编程

从前面的介绍可以看到，如果只在 MATLAB 命令窗口输入命令来编写 GUI 界面，那么

编程的效率实在是太低了。读者学过编写 MATLAB 中的 M 文件后（M 文件的具体编写请参阅第 5 章）立即会想到，通过编写脚本式 M 文件或函数式 M 文件来进行 MATLAB 的 GUI 程序设计。本节就是重点介绍如何编写 GUI 的 M 文件。

8.6.1 全局变量与用户数据属性

1．全局变量

在进行 GUI 程序设计时，如果定义了一个全局变量，那么该变量可以被所有的函数与 Callback 例程调用。全局变量在函数的公共区说明，变量的定义规则遵循 MATLAB 的规定。一般地讲，全局变量名应该大写，以便容易辩别。如果程序代码比较简单，全局变量的使用会使程序的编写更简单。下面编写一个画线的函数式 M 文件 Drawing1.m 来说明如何利用全局变量进行 GUI 程序设计。

```
function drawing()
% 输入变量个数判断
if nargin>0
   error(“too many input arguments”)
   return
end

global AX_H1 AX_H2     % 定义的全局变量

t = 0:pi/20:2*pi;
s = sin(t);
c = cos(t);
% 设置坐标轴的默认颜色值
figh = figure('Position',[30 100 800 350],…
                    'DefaultAxesColor',[.8 .8 .8]);
AX_H1 = subplot(1,2,1); grid on
% 设置第一个坐标轴的默认线型
set(AX_H1,'DefaultLineLineStyle','-.');
% 设置第一个坐标轴的线宽
set(AX_H1,'LineWidth',2);
% 设置第一个坐标轴的前景色
set(AX_H1,'Color',[0 1 1]);
% 画两条曲线
line('XData',t,'YData',s)
line('XData',t,'YData',c)
% 标注文本串
text('Position',[3 .4],'String','Sine')
% 利用静态文本框标注文本串
uicontrol(figh,'Style','text','units','normalized',…
     'Position',[0.3 0.5 0.05 0.05],…
```

```
'String','Cosine','HorizontalAlignment','right');

% 第二个坐标轴
AX_H2 = subplot(1,2,2); grid on
% 在第二个坐标轴中，设置标注文本旋转属性的默认值
set(AX_H2,'DefaultTextRotation',90)
% 设置第二个坐标轴的线宽
set(AX_H2,'LineWidth',5)
% 在第二个坐标轴中画线
line('XData',t,'YData',s)
line('XData',t,'YData',c)
% 标注文本串
text('Position',[3 .4],'String','Sine')
text('Position',[2 -.3],'String','Cosine',…
    'HorizontalAlignment','right')
```

需要注意的是，在定义全局变量时，使用“ global ”字符串，千万要记住该字符串中的字母全都是小写。另外，当定义多个全局变量时，各变量间以空格而不是以逗号分开。编写了上述 M 文件后，在 MATLAB 命令窗口输入 drawing1，就画出如图 8-36 的图形。

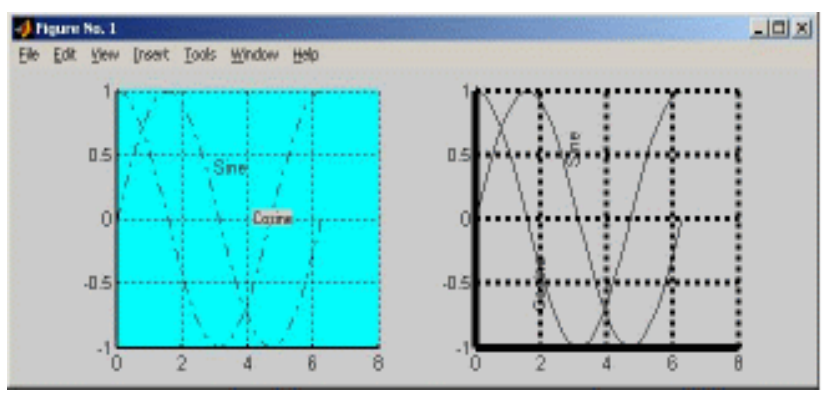


图 8-36　使用全局变量的 drawing 函数画出的图形

虽然使用全局变量可以使编写程序更加方便，但是，如果要编写一个包含多个函数的更复杂的 GUI 程序，此时使用全局变量就很容易引起冲突，MATLAB 也不容易区分各个全局变量。而且，如果用户在 MATLAB 命令窗口输入命令 clear global，那么所有的全局变量都将被清除，这会造成 MATLAB 在进行变量判断时，误认为某些图形界面内的变量没有定义，从而使编写的程序无法运行。对于包含多个图形窗口的更复杂的函数或用独立的回调函数（下面会介绍）实现的情况，用户数据 UserData 属性会更合适。

2．用户数据属性

UserData 属性，即用户数据属性中的属性值，可以在函数之间或递归函数的不同部分之间传递。可以把要使用的多个变量放在一个容易区别的图形对象的 UserData 属性中进行传递。只要获取该对象的对象句柄，别的函数就可以很容易地读取该对象句柄 UserData 属性中的变量值，并使用这些变量。UserData 属性的取值是一个矩阵，通过该属性可以保存与图形

对象有关的信息或数据，利用函数 set 和 get 可以通过对象句柄来调用这些信息。UserData 属性比全局变量使用起来更方便。由于它依赖于不同图形对象的句柄，所以不易引起冲突。另外，当用户在 MATLAB 命令窗口内输入 clear global，不会清除 UserData 属性中的数据，除非图形对象句柄定义为全局变量。但是，使用 UserData 属性编写程序也相应地变得复杂了。

对于上例在程序中使用图形窗口对象的 UserData 属性，编写如下：

```
function drawing()
if nargin>0
   error('too many input arguments')
   return
end
% 此处仍然把 AX_H2 定义为全局变量
global AX_H2
t = 0:pi/20:2*pi;
s = sin(t);
c = cos(t);
% 设置坐标轴的默认颜色值
figh = figure('Position',[30 100 800 350],…
                    'DefaultAxesColor',[.8 .8 .8]);
AX_H1 = subplot(1,2,1); grid on
% 把 AX_H1 存在图形窗口对象的 UserData 属性中
set(figh,'UserData',[AX_H1]);
% 设置第一个坐标轴的默认线型，注意如何调用 UserData 属性中的属性值
set(get(figh,'UserData'),'DefaultLineLineStyle','-.')
% 设置第一个坐标轴的线宽，注意如何调用 UserData 属性中的属性值
set(get(figh,'UserData'),'LineWidth',2)
% 设置第一个坐标轴的前景色，注意如何调用 UserData 属性中的属性值
set(get(figh,'UserData'),'Color',[0 1 1])
line('XData',t,'YData',s)
line('XData',t,'YData',c)
text('Position',[3 .4],'String','Sine')
uicontrol(figh,'Style','text','units','normalized',…
      'Position',[0.3 0.5 0.05 0.05],…
            'String','Cosine','HorizontalAlignment','right');

AX_H2 = subplot(1,2,2); grid on
% 在第二个坐标轴中，设置标注文本旋转属性的默认值，此处仍然使用全局变量
set(AX_H2,'DefaultTextRotation',90)
set(AX_H2,'LineWidth',5)
line('XData',t,'YData',s)
line('XData',t,'YData',c)
text('Position',[3 .4],'String','Sine')
```

```
text('Position',[2 -.3],'String','Cosine',…
    'HorizontalAlignment','right')
```

8.6.2 脚本式 M 文件

脚本式 M 文件中的所有代码，包括命令、变量等，都在 MATLAB 命令窗口内执行。因此，如同全局变量一样，可以随时使用所有在脚本式 M 文件中编写的函数、变量与对象句柄等，并将它们传递给其他函数。

下面是编写一个脚本式 M 文件的例子。该例中，首先在当前图形窗口内创建一个列表框。列表框内有 first，second，third 等 3 个字符串。当用户按任何一个列表项后，就调用编写的脚本式 M 文件，并显示相应的信息。

```
str(1)={'first'};
str(2)={'second'};
str(3)={'third'};
Hm_list=uicontrol('Style','listbox','position'…
,[200 250 80 100],'String',str,…
'Callback','listbox1_callback');
```

上面是在 MATLAB 命令窗口输入的命令。下面是编写的脚本式 M 文件，该文件的文件名为 listbox1_callback.m，供创建的列表框的 Callback 属性调用。

```
index_selected = get(Hm_list,'Value');
list = get(Hm_list,'String');
   listnum=list{index_selected}
switch listnum
case 'first'
    msgbox('This is a first Dialog Box!','Information','non-modal');
case 'second'
    msgbox('This is a second Dialog Box!','Information','non-modal');
case 'third'
    msgbox('This is a third Dialog Box!','Information','non-modal');
otherwise
    msgbox('You have not select list index,Please select a list!','Information','non-modal');
end
```

由于脚本式 M 文件中的所有代码都在 MATLAB 命令窗口内执行，这样很容易引起变量冲突。比如，脚本式文件 M1 中有许多的变量，这其中有一个变量名为 flag，脚本式文件 M2 中也有许多变量，这其中也有一个变量名为 flag。如果 M1 先执行，当它还没有执行完又执行 M2，此时就可能发生 M2 中的变量 flag 覆盖 M1 中的变量 flag。另外，由于所有变量都在工作空间内，会使工作空间内充斥大量的变量。而且，如果用户在 MATLAB 的命令窗口内使用 clear 命令，就可能使很多变量丢失掉。

在 MATLAB 中，脚本式 M 文件与函数式 M 文件相比，脚本式 M 文件的执行速度比函数式 M 文件慢，而且脚本式 M 文件没有输入变量与返回值。因此，采用编写函数式 M 文件来进行 GUI 程序设计应该是最佳的选择。

8.6.3 函数式 M 文件

函数式 M 文件比脚本式 M 文件更方便、更灵活。函数式 M 文件可以有输入参数，可以有返回值。函数式 M 文件中的所有变量，如果没有声明为全局变量，都是临时变量，它们只在本函数内有效，这样就不易引起变量之间的冲突。而且，函数式 M 文件中的变量不会出现在 MATLAB 的工作空间内。

下面是编写的一个函数式 M 文件 listbox2.m 的例子。在该例中，首先打开 GUI 设计工具的对象设计编辑器，然后利用对象设计编辑器创建两个控件对象，它们分别是静态文本框与列表框。其中，静态文本框上的文本字符串是“Double Click ListBox to Open File”。该函数首先把当前目录下的所有文件装载在列表框中。然后，当用户双击某一个文件时，就调用列表框中的 Callback 属性中的 Callback 例程 listbox2('listbox1_Callback',gcbo,[],guidata (gcbo))，打开文件。函数代码如下所示：

```
function varargout = listbox2(varargin)

if nargin <= 1     % 创建 GUI 界面
  if nargin == 0
        initial_dir = pwd;
  elseif nargin == 1 & exist(varargin{1},'dir')
        initial_dir = varargin{1};
  else
        errordlg('Input argument must be a valid directory','Input Argument Error!')
        return
end
% 打开 fig 图形文件
fig = openfig(mfilename,'reuse');
% 创建一个句柄结构，传递给 callback 例程用，并存放该句柄结构
handles = guihandles(fig);
guidata(fig, handles);
% 初始化列表框
load_listbox(initial_dir,handles)
% 作为第一个输出参数，返回图形句柄
if nargout > 0
        varargout{1} = fig;
end

elseif ischar(varargin{1}) % 激活下面的 Callback 例程
try
% 所有的 callback 函数最终都经由 feval 调用
[varargout{1:nargout}] = feval(varargin{:});
   catch
      disp(lasterr);
  end
```

```
end
% --------------------------------------------------------------
% 打开列表框中的文件的 Callback 例程
% --------------------------------------------------------------
function varargout = listbox1_Callback(h, eventdata, handles, varargin)
if strcmp(get(handles.figure1,'SelectionType'),'open')
    index_selected = get(handles.listbox1,'Value');
    file_list = get(handles.listbox1,'String');
    filename = file_list{index_selected};
    if    handles.is_dir(handles.sorted_index(index_selected))
          cd (filename)
          load_listbox(pwd,handles)
    else
       [path,name,ext,ver] = fileparts(filename);
       switch ext
       case '.fig'
            guide (filename)
       otherwise
        try
        open(filename)
        catch
              errordlg(lasterr,'File Type Error','modal')
        end
       end
   end
end
% --------------------------------------------------------------
% 读取当前目录中的文件，并按文件名排序
% --------------------------------------------------------------
 function load_listbox(dir_path,handles)
cd (dir_path)
dir_struct = dir(dir_path);
[sorted_names,sorted_index] = sortrows({dir_struct.name}');
handles.file_names = sorted_names;
handles.is_dir = [dir_struct.isdir];
handles.sorted_index = [sorted_index];
guidata(handles.figure1,handles)
set(handles.listbox1,'String',handles.file_names,…
    'Value',1)
set(handles.text1,'String',pwd)
```

执行该函数式 M 文件的结果如图 8-37 所示。

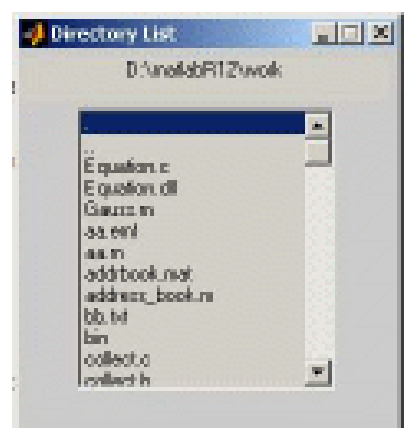


图 8-37　函数式 M 文件执行的界面

8.7 鼠标操作

在 GUI 方式下的应用程序，最重要的输入设备就是鼠标，利用鼠标来驱动消息，利用鼠标来输入数据，利用鼠标来做出判断。事实上，如今任何图形界面应用程序与操作系统都提供了对鼠标操作的支持。MATLAB 也提供了对图形用户界面下应用程序鼠标操作的支持。MATLAB 依据鼠标箭头的位置和鼠标按钮的状态来调用不同的 Callback 例程，作出合理的反应。在 MATLAB 中，鼠标操作可以激活的属性有：对于菜单图形对象，鼠标操作可以激活菜单的 Callback 属性；对于控件对象，鼠标操作可以激活控件对象的 Callback 属性与 ButtondownFcn 属性；对于图形窗口对象，鼠标操作可以激活图形窗口的 ButtondownFcn 属性、WindowButtondownFcn 属性、WindowButtonmotionFcn 属性和 WindowButtonupFcn 属性。当鼠标箭头的位置位于菜单对象或控件对象上时，Callback 属性被激活；当鼠标箭头的位置不在图形对象上，但与图形对象的距离在 5 个像素范围之内时，ButtondownFcn 属性被激活。只要没有在菜单对象或控件对象上进行鼠标操作，都有可能引起 WindowButtondownFcn 属性、WindowButtonmotionFcn 属性或 WindowButtonupFcn 属性。在进行 MATLAB 中的 GUI 设计时，所有创建的图形对象都放在一个特殊的数据结构堆栈中。开始时，图形对象在堆栈结构中的位置，即栈序是按对象的创建顺序来排列的，最先生成的对象在堆栈结构的最底端。但是，当某个图形对象被鼠标操作选中时，该对象就上升到堆栈结构的最顶端。下面根据鼠标按下的动作、鼠标移动的动作和鼠标释放的动作来对鼠标操作进行更详细的介绍。

8.7.1 鼠标按下的处理

当鼠标的箭头位置位于图形窗口对象内，按下鼠标时，根据鼠标箭头位置的不同和与图形对象邻近程度的不同将会发生不同的动作。如果鼠标箭头位置正好位于一个图形对象上（包括控件对象与菜单对象），那么进行鼠标按下的操作后，该图形对象就被选中，因而就成为当前对象，它在堆栈结构中的位置也就上升到堆栈的顶端。然后，启动图形对象的 Callback 属性定义的属性值（M 函数或 MATLAB 表达式）。如果鼠标箭头位置不在图形对象上，但与图形对象的距离在 5 个像素范围内，那么鼠标被按下后，该图形对象被选中，并且图形窗口对象的 WindowButtondownFcn 属性被启动，随后图形对象的 ButtondownFcn 属性被启动。如果鼠标箭头位置位于图形窗口对象上，那么鼠标被按下后，该图形窗口对象成为当前对象，并且图形窗口对象的 WindowButtondownFcn 属性被启动。同时，MATLAB 就会更新图形窗口属性 CurrentObject（当前图形对象）为选中的图形对象句柄值，更新图形窗口属性 CurrentPoint 为当前鼠标箭头位置的坐标值，并更新图形窗口的 SelectionType 属性（SelectionType 属性表示鼠标按键的方式，即是否与组合键 Shift，Ctrl 结合，以及是否单击或双击鼠标按键）的属性值。

8.7.2 鼠标移动的处理

当鼠标的箭头位置位于图形窗口对象内，移动鼠标时，MATLAB 就会更新当前图形窗口的属性 CurrentPoint 为鼠标箭头所在位置的坐标值，并引起当前图形窗口对象的 WindowButtonmotionFcn 属性被启动。但是，如果当前图形窗口对象的 WindowButtonmotionFcn 属性没有被定义，那么 MATLAB 什么也不做，它既不可能启动当前图形窗口对象的 WindowButtonmotionFcn 属性，也不会更新当前图形窗口 CurrentPoint 属性

的属性值。

8.7.3 鼠标释放的处理

当鼠标的箭头位置位于图形窗口对象内，释放鼠标时，MATLAB 就会更新当前图形窗口属性 CurrentPoint 的属性值为鼠标箭头所在位置的坐标值，并引起当前图形窗口对象的 WindowButtonupFcn 属性被启动。但是，如果当前图形窗口对象的 WindowButtonupFcn 属性没有被定义，那么 MATLAB 什么也不做，它既不可能启动当前图形窗口对象的 WindowButtonupFcn 属性，也不会更新当前图形窗口 CurrentPoint 属性的属性值。

8.8 GUI 设计实例

为了使读者对 GUI 的程序设计有更深入的了解，也为了使读者能够从实践中来理解图形用户界面的程序设计，下面举几个综合实例。

【例 8-1】 从列表框中显示当前 MATLAB 工作空间内的变量，并利用这些变量来画图的 GUI 程序。

在 Simple Plotter 中，创建了一个列表框，列出当前 MATLAB 工作空间内的变量，用户可以从中选择几个变量来画图。当用户单击更新命令按钮后，就会更新列表框内的列表项，把当前 MATLAB 工作空间内刚创建的变量显示出来。在初始化列表框时，没有任何列表项被选中。程序中创建的列表框允许进行列表项的多选，但是程序中没有定义列表框的 Callback 属性。为此，在进行程序设计时，必须删除分配给列表框 Callback 属性的字符串。画图命令由创建的 3 个命令按钮来启动。由于使用的几个画图函数只能选择两个变量画图，所以，当用户只选了一个或多于两个变量后，会显示一个出错信息提示对话框来提示用户，让用户选择正确的变量。图 8-38 所示为该程序创建的 GUI 界面。

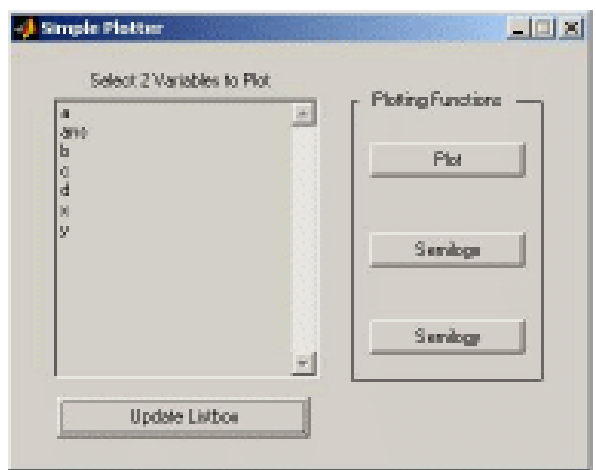


图 8-38 【例 8-1】创建的 GUI 界面

用户从中选择两个变量后，单击任何一个画图命令按钮，如果所选择的变量是正确的，就会画出图形。下面，选择 MATLAB 图形工作空间内创建的变量 c=-2*pi:pi/80:2*pi;d=sin(c);，然后按第一个命令按钮，在区间[-2π 2π]内画出正弦图形，如图 8-39 所示。

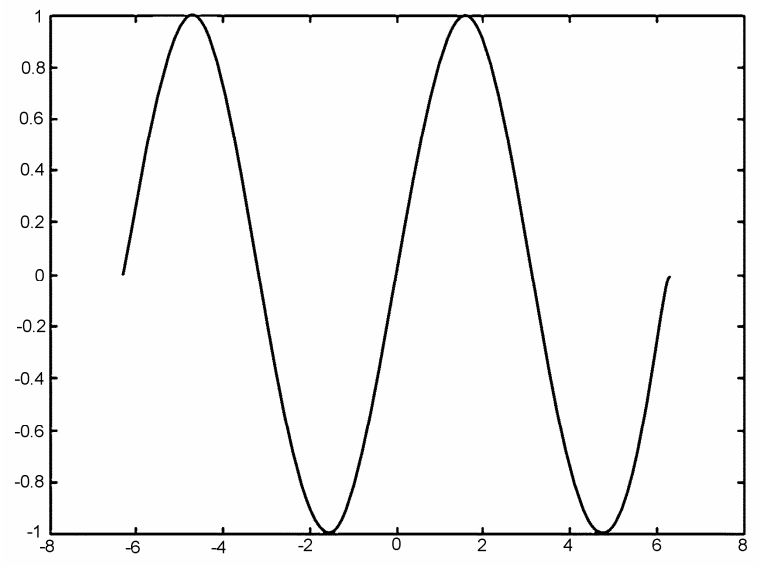


图 8-39　**正弦曲线图**

下面所示为该 GUI **程序的程序源代码。**

```
function varargout = lb(varargin)

% 初始化的代码- DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename,…
                   'gui_Singleton',     gui_Singleton,…
                   'gui_OpeningFcn',    @lb_OpeningFcn,…
                   'gui_OutputFcn',     @lb_OutputFcn,…
                   'gui_LayoutFcn',     [],…
                   'gui_Callback',      []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    varargout{1:nargout} = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% 初始化结束 - DO NOT EDIT

% ---例 SimplePlotter 的 GUI 界面显示前执行的函数
function lb_OpeningFcn(hObject, eventdata, handles, varargin)
% 该函数没有输出参数
```

```
% hObject    图形的句柄（handle to figure）
% eventdata  reserved，该参数保留给以后的 MATLAB 版本使用
% handles    structure with handles and user data (see GUIDATA)
% varargin   命令行的输入参数

%  为例 SimplePlotter 选择默认的命令行输出
handles.output = hObject;

%  更新句柄结构
guidata(hObject, handles);

%  初始化列表框
update_listbox(handles)
set(handles.listbox1,'Value',[])

% ---  该函数的输出结果显示在命令行上
function varargout = lb_OutputFcn(hObject, eventdata, handles)
% varargout  存放输出结果的向量(see VARARGOUT);
% hObject    图形的句柄（handle to figure）
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%  从句柄结构中得到默认的命令行输出结果
varargout{1} = handles.output;

% Update 命令按钮的 Callback 例程
function varargout = update_button_Callback(h, eventdata, handles, varargin)
% hObject    update_button 按钮的句柄
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%  调用初始化列表框函数
update_listbox(handles)

%  初始化列表框函数
function update_listbox(handles)
% hObject    update_button 按钮的句柄
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%  更新列表框中的变量，与当前 workspace 工作区中的变量一致
vars = evalin('base','who');
```

```
set(handles.listbox1,'String',vars)

% 获取列表框内选择的变量值
function [var1,var2] = get_var_names(handles)

list_entries = get(handles.listbox1,'String');
index_selected = get(handles.listbox1,'Value');
if length(index_selected) ~= 2
    errordlg('You must select two variables','Incorrect Selection','modal')
else
    var1 = list_entries{index_selected(1)};
    var2 = list_entries{index_selected(2)};
end

% plot 命令按钮的 Callback 例程
function varargout = plot_button_Callback(h, eventdata, handles, varargin)
% hObject    plot_button 按钮的句柄
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% 调用获取列表框内选择的变量值的函数
[x,y] = get_var_names(handles);
figure(gcf)
evalin('base',['plot(',x,',',y,')'],'errordlg(lasterr,"Error generating plots","modal")')

% semilogx 命令按钮的 Callback 例程
function varargout = semilogx_button_Callback(h, eventdata, handles, varargin)
% hObject    semilogx_button 按钮的句柄
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[x,y] = get_var_names(handles);
figure(gcf)
evalin('base',['semilogx(',x,',',y,')'],'errordlg(lasterr,"Error generating plots","modal")')
% semilogy 命令按钮的 Callback 例程
function varargout = semilogy_button_Callback(h, eventdata, handles, varargin)
% hObject    semilogy_button 按钮的句柄
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[x,y] = get_var_names(handles);
figure(gcf)
```

```
evalin('base',['semilogy(',x,',',y,')'],'errordlg(lasterr,"Error generating plots","modal")')


% --- 该函数在对象创建阶段，设置对象所有的属性后执行
function listbox1_CreateFcn(hObject, eventdata, handles)
% hObject    listbox1 的句柄
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    空的句柄，直到所有的 CreateFcns 函数被调用后才创建

% 列表框控件通常有一个白色的背景色，如果把'usewhitebg' 设置为 0，就用默认值
usewhitebg = ispc;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

【例 8-2】　设置 Simulink 模型参数的 GUI 程序。

本例演示如何创建一个 GUI 程序来设置 Simulink 模型的参数。并且，可以通过单击命令按钮“Simulate and store results>>”来进行模拟，并单击命令按钮 Plot 来画出图形。从本例可以学到如下知识：

（1）如何从 GUI 界面上为 Simulink 模型打开并设置参数。

（2）将滑标控件与可编辑文本框相结合来设置参数，其中可编辑文本框显示滑标控件的当前值，并接受用户输入的值。

（3）根据 GUI 的状态，使命令按钮 Remove 与 plot 的属性 Enable 有效或无效。

（4）通过一个隐藏的图形窗口句柄直接输出图形。

（5）增加一个帮助按钮来显示帮助信息。

图 8-40 所示为创建的 GUI 界面。

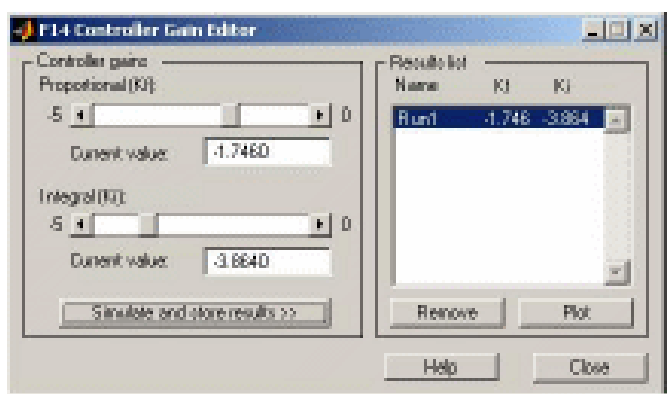


图 8-40　设置 Simulink 模型参数的 GUI 界面

下面所示的就是该 GUI 程序的程序源代码。

```
function varargout = f14ex(varargin)

% 初始化的代码- DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename,…
                   'gui_Singleton',  gui_Singleton,…
                   'gui_OpeningFcn', @f14ex_OpeningFcn,…
                   'gui_OutputFcn',  @f14ex_OutputFcn,…
                   'gui_LayoutFcn',  [],…
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    varargout{1:nargout} = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% 初始化结束 - DO NOT EDIT


% ---Simulink 的模型界面可见之前执行的函数
function f14ex_OpeningFcn(hObject, eventdata, handles)
% 该函数没有输出参数
% hObject    图形的句柄（handle to figure）
% eventdata  reserved
% handles    structure with handles and user data (see GUIDATA)
% varargin   命令行输入参数

fig = handles.F14ControllerEditor;

% ------------------------------------------------------------
% If f14.mdl is not open: open it and bring GUI to front
% Also set model parameters to match GUI settings
% ------------------------------------------------------------
% 调用模型打开函数，如果 simulink 模型没有打开，通过该函数打开模型，并设置模型的参数与
 % GUI 界面相匹配
model_open(handles)

% 为 f14ex 选择默认的命令行输出
```

```
handles.output = hObject;

% 更新句柄结构
guidata(hObject, handles);


% --- 该函数的输出结果显示在命令行上
function varargout = f14ex_OutputFcn(hObject, eventdata, handles)
% varargout   存放输出结果的向量(see VARARGOUT);
% hObject     图形的句柄（handle to figure）
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% 从句柄结构中得到默认的命令行输出结果
varargout{1} = handles.output;

% ------------------------------------------------------------
% 该函数用于确保 Simulink 模型是打开的
% ------------------------------------------------------------
function model_open(handles)
% 确保 Block 对话框仍然是打开的
if  isempty(find_system('Name','f14')),
   open_system('f14'); open_system('f14/Controller')
   set_param('f14/Controller/Gain','Position',[275 14 340 56])
   figure(handles.F14ControllerEditor)
   % 把 GUI 界面上的参数 Kf 与 Ki 的值放入 Block 对话框
   set_param('f14/Controller/Gain','Gain',…
             get(handles.KfCurrentValue,'String'))
   set_param('f14/Controller/Proportional plus integral compensator',…
             'Numerator',…
             get(handles.KiCurrentValue,'String'))
end

% ------------------------------------------------------------
% %调用 KfValueSlider 控件的 Callback 例程
% ------------------------------------------------------------
function varargout = KfValueSlider_Callback(h, eventdata, handles)
% hObject     KfValueSlider  控件的句柄
% eventdata   reserved
% handles     structure with handles and user data (see GUIDATA)

% get(hObject,'Value')  返回 slider 控件的当前位置
```

```
% get(hObject,'Min') 和 0get(hObject,'Max') 获取 slider 控件的范围

% 确保 Simulink 模型是打开的
model_open(handles)

% %从滑标中得到 Kf 参数的新值
NewVal = get(h,'Value');

% 设置 KfCurrentValue 可编辑文本框内的值为滑标的当前值
set(handles.KfCurrentValue,'String',NewVal)

% 设置 Gain 参数中 Kf 的值为滑标的当前值
set_param('f14/Controller/Gain','Gain',num2str(NewVal))

% ------------------------------------------------------------
% % 调用 KfCurrentValue 可编辑文本框的 Callbck 例程
% ------------------------------------------------------------
function varargout = KfCurrentValue_Callback(h, eventdata, handles)
% hObject    KfCurrentValue 控件的句柄
% eventdata  reserved
% handles    structure with handles and user data (see GUIDATA)

% get(hObject,'String') 以字符的形式返回编辑框中的内容
% str2double(get(hObject,'String')) 以 double 的形式返回编辑框中的内容

% 确保 Simulink 模型是打开的
model_open(handles)

% 从可编辑文本框内得到 Kf 的新值
NewStrVal = get(h,'String');
NewVal = str2double(NewStrVal);

% 检验输入可编辑文本框内的参数是否是在允许范围内
if  isempty(NewVal) | (NewVal< -5) | (NewVal>0),
  % 不是，仍然为原来的 KfValueSlider 滑标的值
  OldVal = get(handles.KfValueSlider,'Value');
  set(h,'String',OldVal)

else
  %是，设置 KfValueSlider 滑标新值为输入的值
  set(handles.KfValueSlider,'Value',NewVal)
```

```
  % 设置 Gain 参数中 Kf 的值为滑标的当前值
  set_param('f14/Controller/Gain','Gain',NewStrVal)
end

% ------------------------------------------------------------
% %调用 KiValueSlider  控件的 Callback 例程
% ------------------------------------------------------------
function varargout = KiValueSlider_Callback(h, eventdata, handles)
% hObject        KiValueSlider 控件的句柄
% eventdata    reserved
% handles        structure with handles and user data (see GUIDATA)

% get(hObject,'Value')  返回 slider 控件的当前位置
% get(hObject,'Min')  和 0get(hObject,'Max')  获取 slider 控件的范围

% 确保 Simulink 模型是打开的
model_open(handles)

% %从滑标中得到 Ki 参数的新值
NewVal = get(h,'Value');

% 设置 KiCurrentValue 可编辑文本框内的值为滑标的当前值
set(handles.KiCurrentValue,'String',NewVal)

% 设置 Gain 参数中 Ki 的值为新值，利用函数 num2str 进行数值型与字符型间的转换
set_param('f14/Controller/Proportional plus integral compensator','Numerator',num2str(NewVal))

% ------------------------------------------------------------
% %调用 KiCurrentValue 可编辑文本框的 Callbc 例程
% ------------------------------------------------------------
function varargout = KiCurrentValue_Callback(h, eventdata, handles)
% hObject      KiCurrentValue 控件的句柄
% eventdata    reserved
% handles        structure with handles and user data (see GUIDATA)


model_open(handles)

% 从可编辑文本框内得到 Ki 的新值
NewStrVal = get(h,'String');
NewVal = str2num(NewStrVal);
```

```
% 检验用户输入可编辑文本框内的值是否在允许的范围内
if    isempty(NewVal) | (NewVal< -5) | (NewVal>0),
  % 不是
  OldVal = get(handles.KiValueSlider,'Value');
  set(h,'String',OldVal)

else
  %是
  set(handles.KiValueSlider,'Value',NewVal)

  %% 设置 Gain 参数中 Ki 的值为滑标的当前值
  set_param('f14/Controller/Proportional plus integral compensator','Numerator',NewStrVal)
end



% ---------------------------------------------------------
% 调用 Simulate and store results 按钮命令的 Callback 例程
% ---------------------------------------------------------
function varargout = SimulateButton_Callback(h, eventdata, handles)
% hObject       SimulateButton 按钮的句柄
% eventdata    reserved
% handles       structure with handles and user data (see GUIDATA)

[timeVector,stateVector,outputVector] = sim('f14');

% 读出原先的 results 值的数据结构
if isfield(handles,'ResultsData') & ~isempty(handles.ResultsData)
  ResultsData = handles.ResultsData;
  % %决定当前运行的最大值
  maxNum = ResultsData(length(ResultsData)).RunNumber;
  ResultNum = maxNum+1;
Else
% 开始 results 的一个新的数据结构
  ResultsData = struct('RunName',[],'RunNumber',[],…
        'KiValue',[],'KfValue',[],'timeVector',[],'outputVector',[]);
  ResultNum = 1;
end

if isequal(ResultNum,1),
  %使 Plot and Remove 命令按钮能用
  set([handles.RemoveButton,handles.PlotButton],'Enable','on')
end
```

```
%%得到参数 Ki and Kf 的值，并把它们放在 results 列表框内
Ki = get(handles.KiValueSlider,'Value');
Kf = get(handles.KfValueSlider,'Value');

ResultsData(ResultNum).RunName = ['Run',num2str(ResultNum)];
ResultsData(ResultNum).RunNumber = ResultNum;
ResultsData(ResultNum).KiValue = Ki;
ResultsData(ResultNum).KfValue = Kf;
ResultsData(ResultNum).timeVector = timeVector;
ResultsData(ResultNum).outputVector = outputVector;

% %在列表框内创建一个新的 results 字符串
ResultsStr = get(handles.ResultsList,'String');
if isequal(ResultNum,1)
  ResultsStr = {['Run1                ',num2str(Kf),'     ',num2str(Ki)]};
else
  ResultsStr = [ResultsStr; {['Run',num2str(ResultNum),'                ',num2str(Kf),'     ',num2str(Ki)]}];
end
set(handles.ResultsList,'String',ResultsStr);

% %存储新的 ResultsData
handles.ResultsData = ResultsData;
guidata(h,handles)

% ------------------------------------------------------------
% %调用 Remove 命令按钮的 Callback 例程
% ------------------------------------------------------------
function varargout = RemoveButton_Callback(h, eventdata, handles)
% hObject       RemoveButton 控件的句柄
% eventdata     reserved
% handles       structure with handles and user data (see GUIDATA)

currentVal = get(handles.ResultsList,'Value');
resultsStr = get(handles.ResultsList,'String');
numResults = size(resultsStr,1);

%  删除列表框内被选择的数据
resultsStr(currentVal) =[];
handles.ResultsData(currentVal)=[];

%  如果列表框内没有列表项了，使 Remove and Plot 按钮无效，
```

```
% 并设置列表框的 String 属性为空
if isequal(numResults,length(currentVal)),
  resultsStr = {'<empty>'};
  currentVal = 1;
  set([handles.RemoveButton,handles.PlotButton],'Enable','off')
end

% 确保列表框内的值是有效的，重新设置列表框的 Value and String
currentVal = min(currentVal,size(resultsStr,1));
set(handles.ResultsList,'Value',currentVal,'String',resultsStr)

% 存储新的 ResultsData
guidata(h,handles)

% --------------------------------------------------------------
%% 调用 plot 按钮的 Callback 例程
% --------------------------------------------------------------
function varargout = PlotButton_Callback(h, eventdata, handles)
% hObject       PlotButton 的句柄
% eventdata    reserved
% handles       structure with handles and user data (see GUIDATA)

% Callback of the uicontrol handles.PlotButton.
currentVal = get(handles.ResultsList,'Value');

% Get data to plot
legendStr = cell(length(currentVal),1);
plotColor = {'b','g','r','c','m','y','k'};
for ctVal = 1:length(currentVal);
  PlotData{(ctVal*3)-2} = handles.ResultsData(currentVal(ctVal)).timeVector;
  PlotData{(ctVal*3)-1} = handles.ResultsData(currentVal(ctVal)).outputVector;
  numColor = ctVal - 7*( floor((ctVal-1)/7) );
  PlotData{ctVal*3} = plotColor{numColor};
  legendStr{ctVal} = [handles.ResultsData(currentVal(ctVal)).RunName,';    Kf=',…
                num2str(handles.ResultsData(currentVal(ctVal)).KfValue),';    Ki=',…
                num2str(handles.ResultsData(currentVal(ctVal)).KiValue)];
end

% %如果需要，创建画图窗口，并把它存储在句柄结构内
if ~isfield(handles,'PlotFigure') | ~ishandle(handles.PlotFigure),
  handles.PlotFigure = figure('Name','F14 Simulation Output','Visible','off',…
        'NumberTitle','off','HandleVisibility','off','IntegerHandle','off');
```

```
  handles.PlotAxes = axes('Parent',handles.PlotFigure);
  guidata(h,handles)
end

% Plot data
pHandles = plot(PlotData{:},'Parent',handles.PlotAxes);

% % 增加一个图标
legend(pHandles(1:2:end),legendStr{:})
% 使图可见，并放在最前端
figure(handles.PlotFigure)

% --------------------------------------------------------------
% % 调用 Help 按钮的 Callback 例程
% --------------------------------------------------------------
function varargout = HelpButton_Callback(h, eventdata, handles)
% hObject    HelpButton 按钮的句柄
% eventdata  reserved
% handles    structure with handles and user data (see GUIDATA)

% Callback of the uicontrol handles.HelpButton.
HelpPath = which('f14ex_help.html');
web(HelpPath);

% --------------------------------------------------------------
% % 调用 Close 按钮的 Callback 例程
% --------------------------------------------------------------
function varargout = CloseButton_Callback(h, eventdata, handles)
% hObject    CloseButton 按钮的句柄
% eventdata  reserved
% handles    structure with handles and user data (see GUIDATA)

% % 关闭 GUI 和打开的图形窗口
if isfield(handles,'PlotFigure') & ishandle(handles.PlotFigure),
  close(handles.PlotFigure);
end
close(handles.F14ControllerEditor);

% --- 该函数在对象创建阶段，设置对象所有的属性后执行
function ResultsList_CreateFcn(hObject, eventdata, handles)
% hObject    ResultsList 的句柄
% eventdata  reserved -
```

```
% handles      empty - handles not created until after all CreateFcns called

% 列表框控件通常有一个白色的背景色，如果把'usewhitebg' 设置为 0，就用默认值
usewhitebg = 1;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

【例 8-3】　读地址本的 GUI 程序。

本例演示如何从一个 MAT 文件中读出人的姓名与电话号码。用户可以增加新的姓名与电话号码，将它们存储在同一个 MAT 文件中，或者存储在一个新的 MAT 文件中。从本例可以学到如下知识：

（1）使用保存与打开对话框让用户打开或保存一个 MAT 文件。

（2）创建两种菜单类型，即 uimenu 与 uiocntextmenu，并为菜单对象定义 Callback 例程。

（3）使用 GUI 句柄结构来保存或读出数据。

（4）设置图形窗口对象的 Resize 属性，使用户无法在纵向拉长图形窗口。

（5）跟踪各个变量（包括 MAT 文件名，人名与电话号码，图形窗口的位置与大小等）的变化情况，并且使得各个 Callback 例程适应这种变化。

图 8-41 所示为创建的 GUI 界面。

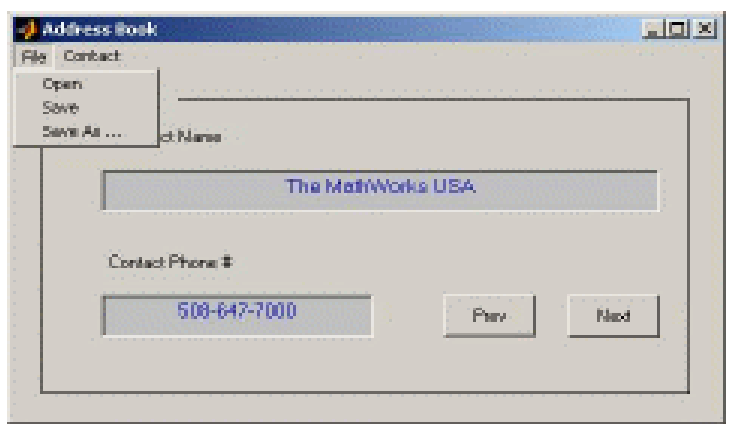


图 8-41　读地址本的 GUI 程序界面

用户可以通过 Prev 与 Next 按钮来读前一条或下一条记录。也可以通过菜单读出别的地址本文件或存储新建的文件。通过 Contact 菜单下的 Create new…或用户单击 Frame 控件对象弹出 Uicontextmenu 菜单，从中选择 New…来创建新的 MAT 文件。

下面所示的就是该 GUI 程序的程序源代码。

```
function varargout = address_book(varargin)

% 初始化的代码- DO NOT EDIT
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename,…
                   'gui_Singleton',     gui_Singleton,…
                   'gui_OpeningFcn',    @address_book_OpeningFcn,…
                   'gui_OutputFcn',     @address_book_OutputFcn,…
                   'gui_LayoutFcn',     [],…
                   'gui_Callback',      []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    varargout{1:nargout} = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% 初始化结束 - DO NOT EDIT


% --- address_book 的 GUI 界面显示前执行的函数
function address_book_OpeningFcn(hObject, eventdata, handles, varargin)
% 该函数没有输出参数
% hObject    图形的句柄（handle to figure）
% eventdata  reserved，该参数保留给以后的 MATLAB 版本使用
% handles    structure with handles and user data (see GUIDATA)
% varargin   命令行的输入参数

% 默认的命令行输出
handles.output = hObject;

%% 更新句柄结构
guidata(hObject, handles);
if nargin < 4
    % 装载默认的 address book
    Check_And_Load([],handles);
% 如果在 varargin 中的第一个元素是'book'，第二个元素是一个 MATLAB 文件，那么装载这个文件
elseif (length(varargin) == 2 & strcmpi(varargin{1},'book') & (2 == exist(varargin{2},'file')))
    Check_And_Load(varargin{2},handles);
else   %返回错误
    errordlg('File Not Found','File Load Error')
    set(handles.Contact_Name,'String','')
    set(handles.Contact_Phone,'String','')
```

```
end

% --- 该函数的输出结果显示在命令行上
function varargout = address_book_OutputFcn(hObject, eventdata, handles)
% varargout   存放输出结果的向量(see VARARGOUT)
% hObject     图形的句柄（handle to figure）
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% 从句柄结构中得到默认的命令行输出结果
varargout{1} = handles.output;


% 检验 MAT 文件，看是否有要求的字段：人名与电话号码。如果是，装载它，并显示
function pass = Check_And_Load(file,handles)

%% 初始化 pass 变量，以标注是否是一个有效文件
pass = 0;

% 如果打开的文件不是一个有效文件，则设置文件名为默认的 MAT 文件名
if isempty(file)
    file = 'addrbook.mat';
    handles.LastFile = file;
    guidata(handles.Address_Book,handles)
end

% 如果是有效文件，装载它
if exist(file) == 2
    data = load(file);
end

% 如果变量是 Address，并有字段 Name 与 Phone，是有效的 MAT 文件
flds = fieldnames(data);
if (length(flds) == 1) & (strcmp(flds{1},'Addresses'))
    fields = fieldnames(data.Addresses);
    if (length(fields) == 2) &(strcmp(fields{1},'Name')) & (strcmp(fields{2},'Phone'))
        pass = 1;
    end
end

% 如果文件有效，显示它
if pass
```

```
  % 把 Addresses 放到句柄结构中
  handles.Addresses = data.Addresses;
  % 显示出人名与电话号码
  set(handles.Contact_Name,'String',data.Addresses(1).Name)
  set(handles.Contact_Phone,'String',data.Addresses(1).Phone)
  % 设置索引为 1
  handles.Index = 1;
  % 保存修改的句柄结构
  guidata(handles.Address_Book,handles)
else
  errordlg('Not a valid Address Book','Address Book Error')
end

% --------------------------------------------------------------
% 调用打开菜单的 Callback，显示打开对话框
% --------------------------------------------------------------
function varargout = Open_Callback(h, eventdata, handles, varargin)
% 使用 UIGETFILE 函数
[filename, pathname] = uigetfile(…
  {'*.mat', 'All MAT-Files (*.mat)';…
        '*.*','All Files (*.*)'},…
  'Select Address Book');
% 如果打开文件对话框的 Cancel 按钮被选择
if isequal([filename,pathname],[0,0])
  return
% 否则，装载文件全名，包括路径，并检验文件，装载它
else
  File = fullfile(pathname,filename);
  % 如果 MAT 文件是有效的，存储文件名
  if Check_And_Load(File,handles)
        handles.LastFIle = File;
        guidata(h,handles)
  end
end
% --------------------------------------------------------------
% 调用 Contact Name 可编辑文本框的 Callback 例程
%--------------------------------------------------------------
function varargout = Contact_Name_Callback(h, eventdata, handles, varargin)
% 从文本框中得到人名与电话号码
Current_Name = get(handles.Contact_Name,'string');
Current_Phone = get(handles.Contact_Phone,'string');
% 如果是空
```

```
if isempty(Current_Name)
  return
end
% 从句柄结构得到当前的 Address
Addresses = handles.Addresses;
% 比较，看当前的名字是否与已经存在的相匹配
for i = 1:length(Addresses)
  if strcmp(Addresses(i).Name,Current_Name)
        set(handles.Contact_Name,'string',Addresses(i).Name)
        set(handles.Contact_Phone,'string',Addresses(i).Phone)
        handles.Index = i;
        guidata(h,handles)
        return
  end
end
% 如果有一个新的名字，问创建一个新的记录吗？
Answer=questdlg('Do you want to create a new entry?',…
  'Create New Entry',…
  'Yes','Cancel','Yes');
switch Answer
case 'Yes'
  Addresses(end+1).Name = Current_Name; % Grow array by 1
  Addresses(end).Phone = Current_Phone;
  index = length(Addresses);
  handles.Addresses = Addresses;
  handles.Index = index;
  guidata(h,handles)
  return
case 'Cancel'
  set(handles.Contact_Name,'string',Addresses(handles.Index).Name)
  set(handles.Contact_Phone,'String',Addresses(handles.Index).Phone)
  return
end


% ------------------------------------------------------------
% 调用 Contact Phone 文本框的 Callback 例程
% ------------------------------------------------------------
% 调用 Contact Phone 文本框的 Callback
function varargout = Contact_Phone_Callback(h, eventdata, handles, varargin)
Current_Phone = get(handles.Contact_Phone,'string');
% 如果有空，返回
```

```
if isempty(Current_Phone)
  return
end
% 从句柄结构中得到当前的 Address
Addresses = handles.Addresses;
Answer=questdlg('Do you want to change the phone number?',…
  'Change Phone Number',…
  'Yes','Cancel','Yes');
switch Answer
case 'Yes'
  % 如果没有名字与新创建的相匹配
  Addresses(handles.Index).Phone = Current_Phone;
  handles.Addresses = Addresses;
  guidata(h,handles)
  return
case 'Cancel'
  set(handles.Contact_Phone,'String',Addresses(handles.Index).Phone)
  return
end

% ------------------------------------------------------------
% 调用 Prev 与 Next 按钮的 Callback 例程
% 传递一个字符串 str 表示哪个对象被选中
% ------------------------------------------------------------
function varargout = Prev_Next_Callback(h, eventdata, handles, str)
% hObject    Prev_Next 按钮的句柄
% eventdata  reserved
% handles    structure with handles and user data (see GUIDATA)

% % 得到记录的索引
index = handles.Index;
Addresses = handles.Addresses;
% 对 Prev 或 Next 按钮采取不同的办法
switch str
case 'Prev'
  % 前一记录
  i = index - 1;
  % 如果索引比 1 小，已到文件头，重新设置索引到文件最后
  if i < 1
      i = length(Addresses);
  end
case 'Next'
```

```
  % 后一记录
  i = index + 1;
        % 如果索引比文件记录数大，已到文件最后，重新设置它到文件头
  if i > length(Addresses)
        i = 1;
  end
end


% 得到被选择索引的记录
Current_Name = Addresses(i).Name;
Current_Phone = Addresses(i).Phone;
set(handles.Contact_Name,'string',Current_Name)
set(handles.Contact_Phone,'string',Current_Phone)

% 更新记录指针，指向新的索引
handles.Index = i;
guidata(h,handles)

% --------------------------------------------------------------
% 调用 Save and Save As 菜单的 Callback 例程
% --------------------------------------------------------------
function varargout = Save_Callback(h, eventdata, handles, varargin)
% 得到选择的菜单对象的 Tag 属性值
Tag = get(h,'Tag');
% 得到 address
Addresses = handles.Addresses;
% 依据不同的菜单，采取不同的措施
switch Tag
case 'Save'
  % 保存为默认的 MAT 文件名
  File = handles.LastFile;
  save(File,'Addresses')
case 'Save_As'
  % 允许用户设置不同的文件名来保存
  [filename, pathname] = uiputfile(…
        {'*.mat';'*.*'},…
        'Save as');
  % 如果按 Cancel 按钮
  if isequal([filename,pathname],[0,0])
        return
  else
```

```
        % 创建文件全名，并保存
        File = fullfile(pathname,filename);
        save(File,'Addresses')
        handles.LastFile = File;
        guidata(h,handles)
    end
end

% --------------------------------------------------------------
% 调用 Contact → Create New 菜单 的 Callback 例程
% --------------------------------------------------------------
function varargout = New_Callback(h, eventdata, handles, varargin)
set(handles.Contact_Name,'String','')
set(handles.Contact_Phone,'String','')

% --------------------------------------------------------------
% % 调用 GUI figure 的 ResizeFcn 属性的 Callback 例程
% --------------------------------------------------------------
function varargout = ResizeFcn(h, eventdata, handles, varargin)
% 获取 figure 的大小与位置
Figure_Size = get(h,'Position');
% 设置 figure 的初始大小
Original_Size = [ 0 0 94 19.230769230769234];
% 如果 figure 的大小比最初的小，那么
if (Figure_Size(3) < Original_Size(3)) | (Figure_Size(4) ~= Original_Size(4))
  if Figure_Size(3) < Original_Size(3)
        % 如果 figure 的宽度比最初的宽度小
        set(h,'Position',[Figure_Size(1) Figure_Size(2) Original_Size(3) Original_Size(4)])
        Figure_Size = get(h,'Position');
  end

  if Figure_Size(4) ~= Original_Size(4)
        % figure 的高度不允许变化
        set(h,'Position',[Figure_Size(1)   Figure_Size(2)+Figure_Size(4)-Original_Size(4)   Figure_Size(3)
Original_Size(4)])
  end
end
% 设置 Contact Name 的属性 units 为'Normalized'
set(handles.Contact_Name,'units','normalized')
% 得到它的位置
C_N_pos = get(handles.Contact_Name,'Position');
% 重新设置它的宽度
```

```
set(handles.Contact_Name,'Position',[C_N_pos(1) C_N_pos(2)    0.789 C_N_pos(4)])
% 再设置 units 的属性值
set(handles.Contact_Name,'units','characters')
% 重新在屏幕上定位 GUI 界面
movegui(h,'onscreen')


% --- 该函数在对象创建阶段，设置对象所有的属性后执行
function Contact_Name_CreateFcn(hObject, eventdata, handles)
% hObject      Contact_Name 的句柄
% eventdata    reserved
% handles      空的句柄，直到所有的 createFcns 函数被调用后才创建

% 列表框控件通常有一个白色的背景色，如果把'usewhitebg' 设置为 0，就用默认值
usewhitebg = ispc;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- 该函数在对象创建阶段，设置对象所有的属性后执行
function Contact_Phone_CreateFcn(hObject, eventdata, handles)
% hObject       Contact_Phone 的句柄
% eventdata    reserved
% handles      空的句柄，直到所有的 createFcns 函数被调用后才创建

% 列表框控件通常有一个白色的背景色，如果把'usewhitebg' 设置为 0，就用默认值
usewhitebg = ispc;
if usewhitebg
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

第 9 章　面向对象编程

本章介绍如何在 MATLAB 中定义自己的类。使用类和对象，可以向 MATLAB 添加新的数据类型和新的操作符。类描述对象的结构，并指示操作符的类型和可以应用于对象的函数，对象是某个类的实例。面向对象编程强调使用类和对象来编写程序。

9.1　概述

可以把类看做是一个具有一定行为的新的数据类型。例如，一个 Polynomial 类重新定义相加操作符（+）以后，可以对多项式完成正确的相加操作。操作符是类的一种方法。也可以把类看做是一个可以当成单一实体的新项目。例如 arrow（箭头）对象，它可以显示在图形上，并像句柄图形对象那样具有属性。可以通过简单地实例化一个 arrow 对象来创建一个箭头，可以为对象指定进行数据存储的 MATLAB 结构，并创建一个 M 文件的类目录来将类添加到 MATLAB 环境中。这些 M 文件基于对象进行操作并提供类的方法。类目录中还可以定义应用于对象的不同 MATLAB 操作符，包括算术操作、索引引用等。可以在类中重新定义内部操作符，即进行操作符重载。

9.1.1　面向对象编程的特点

使用设计良好的类时，面向对象编程可以显著提高代码的重用性。并使程序更易于维护和扩展。用类和对象编程与一般的结构化编程有下面一些主要区别：

（1）能实现函数和操作符重载。

（2）可以覆盖已经存在的 MATLAB 函数的方法。

（3）把用户定义的对象作为变量调用函数时，MATLAB 首先进行校核，看是否有定义对象类的方法，如果有，MATLAB 调用它。

（4）可以实现数据的封装。

（5）对象属性在命令行中不可见，只能用类方法获取它们。

（6）可以创建类的继承层次。子类可以从一个父类（单继承）和多个父类（多继承）那里进行继承。利用继承，可以共享父函数。

（7）可以用聚合创建类，其中，一个对象包含另一个对象。这适用于一种对象是另一种对象的一部分的情况。

9.1.2　MATLAB 的数据类层次

在面向对象的程序中，所有的 MATLAB 数据类都作为类设计成函数。图 4-1 显示了 MATLAB 中定义的 15 种基本的数据类型（或类）。可以通过扩展类层次来给 MATLAB 添加一个继承自结构类的用户类。创建的所有类都基于结构。

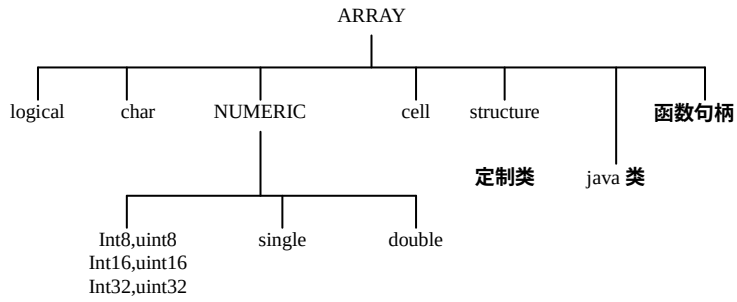


图 4-1　MATLAB 的数据类层次

9.2　在 MATLAB 中创建自己的类

9.2.1　MATLAB 类的方法集合

设计一个 MATLAB 类时，应该提供一个使类能够在 MATLAB 环境中稳定运行的标准的方法集合。利用这些方法，应该可以实现创建类、设置属性以及对对象的引用、赋值、索引和转换等操作。表 4-1 列出了 MATLAB 类中包括的基本方法。

表 4-1　MATLAB 类中的基本方法

方　法	描　述
class constructor	创建一个类对象
display	MATLAB 是否显示对象的内容
set & get	设置和获取类属性
subsref & srbsasgn	用户对象的索引引用和赋值
end	在使用了对象的索引表达式中支持 end 语法，如 A(1:end)
subsindex	支持在索引表达式中使用对象
converters(如 double.char)	将对象转换为一种 MATLAB 数据类型

9.2.2　类目录

1．安装类目录

定义类方法的 M 文件集中在一个目录中，该目录名由类名前加字符“@”组成。例如：本章中用到的一个示例是与单变量中的多项式有关的一个类。类名和构造函数的名称都是 polynom，定义 polynom 类的 M 文件位于名为@polynom 的目录下。类的目录是 MATLAB 搜索路径上目录的子目录。例如，新的@polynom 目录可以是 MATLAB 工作目录的子目录或你自己的已经添加到搜索路径中的个人目录。

2．添加类目录到 MATLAB 路径

创建类目录以后，需要更新 MATLAB 路径，这样，MATLAB 才能找到类源文件。类目录不应该直接把父目录添加到 MATLAB 路径。例如，如果@polynom 类目录的位置是 c:\my_classes\@polynom，用 addpath 命令把类目录添加到 MATLAB 路径中：

```
addpath c:\my_classes;
```

9.2.3 构造函数

构造函数是对类进行操作时第一个运行的函数，它常常用于对象数据的初始化。可以通过调用构造函数并给它传递合适的输入变量来创建类。在 MATLAB 中，构造函数与类具有相同的名称。例如：声明 P＝polynom(10–2–5)；创建一个属于 polynom 类的名为 P 的对象。一旦创建了 polynom 对象，就可以用类定义的方法操作对象。

特定类的@目录必须包含一个作为类的构造函数的 M 文件，除了@前缀和.m 扩展名以外，构造函数的名称与定义类名目录的名称相同。构造函数通过初始化数据结构和实例化类对象来创建对象。

构造函数必须运行一定的函数，这样，对象才能在 MATLAB 环境中正确运行。通常，根据输入参数的个数和类型的不同，构造函数有 3 种，分别对应于没有输入变量、同一个类的一个对象作为输入变量和某种类型的数据作为输入变量的情况。

1．没有输入变量

如果没有输入变量，构造函数将创建一个默认对象。因为没有输入，没有数据创建对象。所以用空值或默认值简单地初始化对象的数据，调用类函数实例化对象，并返回对象作为输出变量。在下面两种情况下需要使用本语法：

- 在工作空间中载入对象时，load 函数调用没有变量的构造函数。
- 创建对象数组时，MATLAB 向数组中添加对象。

2．对象输入变量

如果变量列表中的第一个输入变量是同一类的对象，构造函数将简单地返回对象，使用 isa 函数确定变量是否为类的成员。

3．数据输入变量

如果有输入变量，但没有同一类的对象，则构造函数使用输入的数据，常用于使用 varargin 输入变量有开关控制的程序中。

9.2.4 设置和获取对象数据

需要为类编写方法来提供获取对象数据的途径。set 和 get 方法提供了获取对象数据的便捷途径。

9.2.5 类方法

1．在 MATLAB 中设置类方法

类方法实际上是 M 文件函数，只是该函数的输入变量是对象。类方法必须位于该类的类目录（@class_name 目录）中。这是 MATLAB 查找类方法的第一个位置。调用类方法的语法与调用函数的方法相同，一般地，它与下面的表达类似：

```
out1 ,out2,…=method_name(object,arg1,arg2,…)
```

例如，假设名为 polynom 的类有一个 char 方法。该方法把 polynom 对象转换为一个 character 字符串，并返回该字符串。下面调用 polynom 对象 p 的 char 方法：

```
s=char(p)；
```

使用 class 函数，可以确认返回值 s 是一个 character 字符串。

```
class(s)
```

```
ans =
        char
s
s =
      x^3-2*x-5
```

可以使用 methods 方法来生成类中的所有方法的列表。

2．私有方法

私有方法只能被类中的其他方法调用。通过将相关的 M 文件放到一个@class_name 目录的 private 子目录中来定义私有方法。例如@class_name\private\update_obj.m。update_obj 方法的使用范围局限于 class_name 类。这意味着 update_obj 可被@class_name 目录中定义的任何方法调用，但它不能被 MATLAB 命令行或目录路径以外的方法，包括父方法所调用。私有方法和私有函数之间的区别在于，私有方法有一个输入参数是对象，而私有函数却没有。

3．调试类方法

可以采用与调试其他 M 文件相同的方法来用 MATLAB 调试命令调试对象的方法。惟一的区别在于需要在方法名前添加类的目录，就像下面例子中用 dbstop 显示的那样。

```
dbstop @polynom/char
```

调试类方法时，已经获得了为类定义的所有方法，包括继承方法、私有方法和私有函数。

9.2.6 引用和赋值

用户类在 MATLAB 中实现新的数据类型，通过索引引用获取对象数据很有用。例如，如果 A 是一个 doule 类型的数组，则 A(i)返回 A 的第 i 个元素。作为类设计器，可以确定索引引用对于对象的意义。例如，假设定义一个创建多项式对象的类，各对象包含多项式的系数。多项式对象的一个索引引用 P(3)能返回 x^3 的系数的值、x=3 时多项式的值或某些不同的值，通过创建两个类方法，即 subsref 和 subsasgn 来定义特定的索引行为。无论什么时候引用索引或指定类对象，MATLAB 都会调用这些方法。如果没有为类定义这些方法，则该类的对象的索引就没有定义。

1．索引引用

用对象在赋值语句的右端使用索引或指定字段称为索引引用。在这种情况下，MATLAB 调用 subsref 方法。对象索引引用可以有 3 种形式：数组索引、单元数组索引和结构字段名。B=subsref(A,S)的调用结果为：

```
A(I)
A{I}
A.field
```

MATLAB 给 subsref 传递两个变量：A 和 S。第一个变量 A 是引用的对象，第二个变量 S 是有两个字段的结构数组：S.type 是包含指定索引类型的“ ( ) ”,“ { } ”或“ . ”的字符串。其中，小括号表示数值数组，大括号表示单元数组，点表示结构数组。S.subs 是包含实际子脚本的单元数组或字符串。用做索引的冒号(:)作为字符串的传递。例如，表达式 A(1:2,:)导致 MATLAB 调用 subsref(A,S)，其中，S 是一个 1×1 的结构数组，且

```
S.type='()'
```

```
S.subs={1:2, ': '}
```

类似地，对于 A{1:2}，**有**

```
S.type='{}'
S.subs={1:2}
```

对于 A.field，**调用** subsref(A,S)，**有**

```
S.type='. '
S.subs='field'
```

这些简单的调用可以组合成更复杂的索引表达式。此时，length(S)**是索引水平数。例如，**

```
A(1,2). name(3,4)
```

调用 subsref(A,S)，**其中，**S **是一个** 3×1 **的结构数组，其值为：**

```
S(1).type='()'        S(1).subs='{1,2}'
S(2).type='. '        S(2).subs='name'
S(3).type='()'        S(3).subs='{3:4}'
```

2．编写 subsref **函数**

subsref **方法必须解释** MATLAB **中传递的索引表达式。一个典型用途是使用开关语句确定用到的索引类型，并获取实际的编号。下面** 3 **段代码演示如何解释输入变量，每种情况下，函数必须返回值** B。

对于数组编号：

```
switch S.type
case '()'
    B = A(S.subs{:});
end
```

对于单元编号：

```
switch S.type
case '{}'
    B = A(S.subs{:}); % A 是一个单元数组
end
```

对于结构编号：

```
switch S.type
case '.'
    switch S.subs
    case 'field1'
        B = A.field1;
    case 'field2'
        B = A.field2;
    end
end
```

3. 索引赋值

用对象在赋值语句的左端使用索引或字段指示符称为索引赋值。此时，MATLAB **调用一个名为** subsasgn **的方法。对象索引赋值可以有** 3 **种形式：数组索引、单元数组索引和结构字段名。以** A=subsasgn(A,S,B)**的形式调用** subsasgn **函数，得到下面的结果：**

```
A(I) = B
A{I} = B
A.field = B
```

第一个变量 A **是引用的对象；第二个变量** S **与** subsref **函数具有相同的字段；第三个变量** B **是一个新值。**

9.2.7 对象索引

1．方法中的对象索引

如果为方法中进行了索引引用，MATLAB **使用它内置的** subsref **函数获取类的数据。如果方法从另一个类中获取数据，则** MATLAB **在那个类中调用重载的** subsref **函数。对于索引赋值和** subsasgn**，存在同样的情况。下例显示类** employee **中定义的方法** testref**。本方法用** subsref **函数引用对象内的字段和地址。它还在另一个类中引用相同的字段，这时使用重载的** subsref **函数。**

```
function testref(myclass,otherclass)
myclass.address                    % 使用内置的 subsref 函数
otherclass.address                 % 使用重载的 subsref 函数
```

本例创建一个 employee **对象和一个** company **对象。**

```
empl = employee('Johnson','Chicago');
comp = company('The MathWorks','Natick');
```

调用 employee **类方法** testref**，**MATLAB **只在获取方法所属类以外的数据时才使用重载的** subsref **函数。**

```
testref(empl,comp)
ans =                              % 调用了内置的 subsref 函数
    Chicago
ans =                              % 调用了@company\subsref
Executing @company\subsref…
Natick
```

2．end **索引**

在对象索引表达式中使用 end **时，**MATLAB **调用对象的方法，如果希望在与类对象有关的索引表达式中能使用** end**，必须为你的类定义一个** end **方法。**end **方法的调用格式为：**

```
end(a,k,n)
```

其中，a **为用户对象，**k **为** end **语法用到的表达式中的索引，**n **是表达式中索引指数的总个数。例如，考虑表达式** A(end-1,:)**，**MATLAB **可以这样使用** end **方法：**

```
end(A,1,2)
```

即，共有两个元素，语句发生在第一个元素。end **方法返回第一维的最后一个元素的索引值。**

3．用对象索引对象

MATLAB **把对象作为索引时，它调用** subsindex **方法。例如，假设有一个对象** a**，而且希望使用该对象对另一个对象** b **进行编号，即：**

```
C=b(a);
```

subsindex 方法会尽可能简单地将该对象转换为可以用作编号的 double 格式，如下例所示：

```
function d = subsindex(a)
    d = double(a);
```

注意，subsindex 的值基于 0，而不是基于 1。

9.2.8 识别对象

1．isa 函数

用在构造函数中的函数 class 和 isa 也能用在类目录以外，下面表达式

```
isa(a, 'class_name');
```

检查 a 是否指定类的对象。例如，如果 p 是一个 polynom 对象，下面的每个表达式都返回 True：

```
isa(pi,'double');
isa('hello','char');
isa(p,'polynom');
```

在类目录以外，class 函数只有一个变量（只有在构造函数内部，class 函数才能有一个以上的变量），表达式 class（a）返回一个包含 a 的类名的字符串。例如：

```
class(pi),
class('hello'),
class(p)
```

返回

```
'double',
'char',
'polynom'
```

2．whos 函数

用 whos 函数察看 MATLAB 工作空间中的对象。

```
whos
  Name        Size            Bytes  Class
    p         1x1               156  polynom object
```

3．display 方法

对象为不以分号结尾的表达式的结果时，MATLAB 调用一个名为 display 的方法。例如，创建一个 double 型变量 a，调用 MATLAB 的方法为：

```
a = 5
a =
     5
```

应该在类中定义一个 display 方法，这样，从类中引用对象时，MATLAB 能在命令行中显示值。在许多类中，方法能简单地打印变量名，然后使用转换器方法打印内容或变量的值，因为 MATLAB 把输出显示成字符串，所以必须定义 char 方法，将对象的数据转换为字符串。

9.2.9 转换器方法

转换器方法是一个类方法，它与另一个类具有相同的名称，如 char 或 double 等。转换器方法把某类的一个对象作为输入，返回其他类的一个对象，调用格式为：

```
b=class_name(a)
```

其中，a 是一个不同于 class_name 的类对象。

9.3 重载

在许多情况下，当变量是对象时，可能希望改变 MATLAB 操作符和函数的行为。通过重载相关的函数可以达到目的。重载使一个函数可以有不同的形式、不同数目的输入变量，并能适应对象的不同情况进行操作。

9.3.1 操作符重载

每个 MATLAB 内部操作符都有一个相关的函数名，如‘ + ’操作符的相关函数名为 plus.m。可以在类目录中通过创建具有对应名称的 M 文件来重载任何操作符。例如，如果 p 或 q 是 class_name 类型的对象，表达式 p+q 生成一个对函数@class_name/plus.m 的调用。如果 p 和 q 是不同类的对象，则 MATLAB 采用优先原则确定先用哪个方法（关于优先原则，请参见 9.6 节的内容）。

9.3.2 函数重载

可以通过创建与类目录中名称相同的函数来重载任何函数。函数调用对象时，总是首先在搜索路径中的类目录中进行查找，然后才是在其他位置中查找。例如，要重载 plot 函数，只需要简单地把你自己的 plot.m 放到类目录中就行了。

9.3.3 示例—— 一个多项式类

下面通过定义一个名为 polynom 的新类来为多项式实现一个 MATLAB 数据类型。该类为数据保存指定一个结构，并定义操作 polynom 对象的方法路径(@polynom)。

1．Polynom 类的数据结构

polynom 类表示一个带一行向量的多项式，行向量中包含降序排列的变量幂次项的系数。这样，polynom 对象 p 是一个具有单一字段 p，c 的结构。p，c 中包含各系数。本字段只在@polynom 目录中的方法内可以获得。

2．polynom 类的方法

为了创建一个在 MATLAB 环境中运行良好的类，并为多项式数据类型提供有用的功能，polynom 类实现下面的方法：

- 构造函数 polynom.m。
- double 转换。
- char 转换。
- display 方法。
- subsref 方法。
- 重载的 +、- 和 * 操作符。
- 重载的 roots.polyval.plot 和 diff 函数。

（1）Polynom 类的构造函数

下面是 polynom 类的构造函数@polynom/polynom.m。

```
function p = polynom(a)
if nargin == 0
    p.c = [];
    p = class(p,'polynom');
elseif isa(a,'polynom')
    p = a;
else
    p.c = a(:).';
    p = class(p,'polynom');
end
```

polynom 类有 3 个构造函数，它们具有不同的形式。

- 没有输入变量——如果调用没有变量的构造函数，它返回一个带空字段的 polynom 对象。
- 输入变量为对象——如果调用输入变量为对象的构造函数，MATLAB 返回该输入变量。
- 输入变量为系数向量——如果输入变量是一个变量，该变量不是 polynom 对象，则将它改写为行向量的形式，并将它指定为对象结构的字段，class 函数创建 polynom 对象，并在后面由构造函数返回。下面是使用 polynom 类构造函数的一个例子：

```
p=polynom(10-2-5)
```

这将创建一个指定系数的多项式。

（2）转换器方法

转换器方法将一个类的对象转换为另一个类的对象。包含在 MATLAB 类中的最重要的转换器方法中的两个是 double 和 char。转换为 double 将生成 MATLAB 传统矩阵，尽管它对某些类可能不合适。转换为 char 对于生成打印输出很有用。

- polynom 到 double 的转换　　polynom 类到 double 转换方法是一个简单的 M 文件：@polynom double.m 它只提取对象 p 的系数向量 p=polynom(10-2-5)。

```
function c = double(p)
c = p.c;
```

语句 double(p)返回

```
ans=
    1    0    -2    -5
```

- polynom 到 char 的转换　　polynom 到 char 的转换是一个关键的方法，因为它生成一个与独立变量 x 的幂次有关的字符串。所以，一旦已经指定了 x，返回的字符串是一个语法上正确的 MATLAB 表达式。这里为@polynom/char.m。

```
function s = char(p)
    if all(p.c == 0)
        s = '0';
    else
        d = length(p.c) - 1;
        s = [];
    for a = p.c;
        if a ~= 0;
```

```
                if ~isempty(s)
                    if a > 0
                        s = [s '+ '];
                    else
                        s = [s ' - '];
                        a = -a;
                    end
                end
                if a ~= 1 | d == 0
                    s = [s num2str(a)];
                    if d > 0
                        s = [s '*'];
                    end
                end
                if d >= 2
                    s = [s 'x^' int2str(d)];
                elseif d == 1
                    s = [s 'x'];
                end
            end
            d = d - 1;
        end
    end
```

如果创建 polynom **对象** p(p=polynom(1 0 –2 –5);)**，然后对** p **调用** char **方法：**

```
char(p)
```

则 MATLAB **生成如下结果：**

```
ans =
    x^3 - 2*x - 5
```

char **返回的值是一个字符串，一旦为** x **定义了标量值，可以将它传递给** eval **函数。例如：**

```
x = 3;

eval(char(p))
ans =
    16
```

（3**）**polynom **的** display **方法**

其 M **文件为**@polynom/display.m**。本方法依赖于** char **方法，生成一个表示多项式的字符串，然后显示在屏幕上。本法生成的输出与** MATLAB **的标准输出相同。即，变量名后面跟等号，然后空一行，然后在一个新行中显示值。**

```
function display(p)
disp(' ');
disp([inputname(1),' = '])
```

```
    disp(' ');
    disp(['     ' char(p)])
    disp(' ');
```

语句 p=polynom(1 0 –2 –5)创建一个 polynom 对象，因为语句不以分号结束，结果输出为：

```
p =
     x^3 - 2*x - 5
```

（4）polynom 的 subsref 方法

对于 polynom 对象 p

```
p = polynom([1 0 -2 -5]);
```

下面的子脚本表达式返回 x=3 和 x=4 处多项式的值：p([3 4])

```
ans =
     16     51
```

subsref 方法利用 polynom 类中定义的 char 方法，生成一个可以计算的表达式。

```
function b = subsref(a,s)
    switch s.type
    case '()'
        ind = s.subs{:};
        for k = 1:length(ind)
            b(k) = eval(strrep(char(a),'x',num2str(ind(k))));
        end
    otherwise
    error('Specify value for x as p(x)')
end
```

一旦多项式表达式已经由 char 方法生成，使用 strrep 函数交换字符 x，以传递值的内容。然后 eval 函数评价表达式并在输出变量中返回值。

（5）重载算术操作符

有些算术操作符对于多项式是有意义的，应该在 polynom 类中实现。重载算术操作符时，记住进行操作的数据类型。本例中，为 polynom 类定义了 plus，minus 和 mtimes 方法，以便实现加、减法和乘法等运算。

- 重载“+”操作符

如果 p 或 q 为 polynom 类对象，表达式 p+q 调用@polynom/plas.m 函数，下面的 M 文件重新为 polynom 类定义“+”操作符。

```
function r = plus(p,q)
p = polynom(p);
q = polynom(q);
k = length(q.c) - length(p.c);
r = polynom([zeros(1,k) p.c] + [zeros(1, -k) q.c]);
```

该函数首先将两个输入变量定义为多项式。这样保证至少有一个输入不是多项式的情况下函数也能正常进行。然后函数获取两个系数向量，并且在必要时将其中的某个置 0，以使它们具有相同的长度。实际求和是求两个多项式的系数的和，用向量表示。最后，函数第三次调用 polynom 构造函数，得到结果。

● **重载“ - ”操作符**

可以用相同的方法实现“ - ”操作符。MATLAB **调用**@polynom/minus.m **计算** p-q**。**

```
function r = minus(p,q)
p = polynom(p);
q = polynom(q);
k = length(q.c) - length(p.c);
r = polynom([zeros(1,k) p.c] - [zeros(1,-k) q.c]);
```

● **重载“ * ”操作符**

MATLAB **调用方法**@polynom/mtimes.m **计算** p*q**。因为它从** MATLAB **矩阵乘法**(matrix multiplication)**中继承而来，所以函数名的第一个字母为** m**。**

```
function r = mtimes(p,q)
p = polynom(p);
q = polynom(q);
r = polynom(conv(p.c,q.c));
```

● **使用重载的操作符**

给定 polynom **对象** p=polynom([1 0 -2 -5])**，**MATLAB **在使用语句** q=p+1 **时调用函数** @polynom/plus.m **和**@polynom/mtimes.m**。**

```
q = p+1
r = p*q
```

生成

```
q =
     x^3 - 2*x - 4
r =
     x^6 - 4*x^4 - 9*x^3 + 4*x^2 + 18*x + 20
```

（6**）重载函数**

MATLAB **已经有几个可以操作多项式的函数，它们可以通过重载，用在新的** polynom **对象中。在许多情况下，重载方法可以将原始函数应用于系数字段。**

● **为** polynom **类重载** roots **方法**

方法@polynom/roots.m **计算** polynom **对象所代表的多项式函数的根。**

```
function r = roots(p)
    r = roots(p.c);
    The statement roots(p)
    results in ans =
        2.0946
       -1.0473 + 1.1359i
       -1.0473 - 1.1359i
```

● **为** polynom **类重载** polyval **方法**

polyval **方法计算给定点集的多项式。**@polynom/polyval.m **使用嵌套的乘法或** Horner **法减少计算** x **的不同幂次的乘法操作次数。**

```
function y = polyval(p,x)
y = 0;
```

```
for a = p.c
    y = y.*x + a;
end
```

- **为** polynom **类重载** plot **方法**

用 root **和** polyval **重载** plot **方法。**

```
function plot(p)
    r = max(abs(roots(p)));
    x = (-1.1:0.01:1.1)*r;
    y = polyval(p,x);
    plot(x,y);
    title(char(p))
    grid on
```

- **重载** diff **方法**

方法@polynom/diff.m **对一个多项式进行求导。方法是幂次减** 1**，用原来的幂次乘上对应的系数。**

```
function q = diff(p)
c = p.c;
d = length(c) - 1;    % degree
q = polynom(p.c(1:d).*(d:-1:1));
```

下面用 methods **命令列出类的所有方法。**

```
methods polynom
Methods for class polynom:
chardisplayminusplotpolynomrootsdiffdoublemtimespluspolyvalsubsref
```

下面先对两个 polynom **对象** x **和** p **作运算，然后绘图。**

```
x = polynom([1 0]);
p = polynom([1 0 -2 -5]);
plot(diff(p*p + 10*p + 20*x) - 20)
```

9.4 继承

9.4.1 概念

一个 MATLAB **对象可以从另一个** MATLAB **对象继承属性和行为。当子对象从父对象那里继承时，子对象包括父对象的所有字段，并能够调用父对象的方法，父类的方法可以获取子对象从父对象那里继承的字段，但不能获取子类的新字段。继承是面向对象编程的一个关键特点，它允许子对象利用父对象中已经存在的代码，因而使代码重用更容易。继承使子对象完全像父对象那样工作，便于行为相似、但实现方式不同的相关类的开发。继承有两类：单继承和多继承。前者指子对象只从一个父对象那里继承，后者指子对象从多个父对象那里继承。后面还将讨论聚合，它允许一个对象把另一个对象作为字段之一进行包含。**

9.4.2 单继承

当一个类只从一个父类那里继承属性，并给它自己添加属性时，使用单继承。继承时，属于子类的对象具有与父类相同的字段。所以，父类的方法可以操作属于子类的对象。但是，子类的方法不能操作属于父类的对象，不能直接从子类中获取父类的字段。继承了另一个类的行为的类的构造函数有下面两个特点：

- 为父类调用构造函数，创建继承的字段。
- 类函数的调用语法相对于子类和父类略有不同。用 class 函数建立单继承关系的语法为：

  child_obj = class(child_obj,'child_class',parent_obj);

9.4.3 多继承

多继承通过在构造函数中调用带 3 个以上变量的类方法来实现，即

obj = class(structure,'class_name',parent1,parent2,…)

9.4.4 多层继承

如果父类自己是一个继承的类，则它的子类对象会自动从父类的父类那儿进行继承，因而能实现多个层次的继承。

9.4.5 类属性和方法的可见性

父类不知道子类的属性或方法。子类不能直接获取父类的属性，而必须使用父类的获取方法（如 get 或 subsref 等）来获取父类的属性。使用子类的方法，通过父类在子类结构中的字段来完成。例如，当构造函数创建一个子类对象 c 时，即

c = class(c,'child_class_name',parent_object);

MATLAB 会自动在包含父对象的对象结构中创建一个字段 c.parent_class_name，然后在子类的 display 方法中有一个语句，用来调用父类的 display 方法。

display(c.parent_class_name)

9.5 保存和装载对象

可以使用 MATLAB 的 save 和 load 命令将自定义对象保存到.mat 文件或从.mat 文件中提取出自定义对象。装载对象时，MATLAB 调用对象的构造函数，在工作空间中注册对象正在装载的对象类的构造函数必须能在没有输入变量的情况下被调用，并返回默认值。

对对象使用 save 或 load 命令时，MATLAB 在类目录中寻找名为 saveobj 和 loadobj 的类方法。可以在保存或装载以前通过重载这些方法来修改对象。例如，可以定义 saveobj 方法，它与对象一起保存相关数据，或者在对象类型载入到 MATLAB 工作空间中时编写 loadobj 方法，把对象更新到新的版本中。

9.6 对象优先级

一般地讲，MATLAB 假设对象有相同的优先级，但下面两种情况例外。

- 自定义类具有比 MATLAB 更高的优先级。
- 使用 inferiorto 和 superiorto 函数，自定义类能相对于其他自定义类指定相对优先级。

9.6.1 指定自定义类的优先级

通过在构造函数中调用 inferiorto 或 superiorto 函数来指定自定义类的相对优先级。在优先层次中，inferiorto 函数将一个类放在其他类的下面。该函数的调用语法为：

inferiorto('class1','class2',…)

可以在变量列表中指定多个类，把这些类放在其他类的下面。同样，superiorto 函数将一个类放到优先层次中其他类的上面。该函数的调用语法为：

superiorto('class1','class2',…)

9.6.2 在优先层次中定位

如果在优先层次中，objectA 在 objectB 的上面，则表达式 objectA+objectB 调用@classA/plus.m；相反，如果 objectB 在 objectA 的上面，则该表达示调用@classB/plus.m。

一个目录中可能包含许多函数（M 文件）。只在单个目录中，函数名是惟一的。但多个目录中可以包含有相同名称的函数，在命令行中键入函数时，MATLAB 必须搜索可能调用该函数的所有目录。查找函数时，MATLAB 按路径中罗列的顺序搜索目录，并调用名称与指定函数相匹配的第一个函数。面向对象编程允许有多个同名的方法，对于 MATLAB 来说，这些方法就是类目录中的 MATLAB 函数，根据传递给函数的变量所属的类，MATLAB 确定使用什么方法。例如，如果 p 是一个 portfolio 对象，则 pie3(p)调用@portfolio/pie3.m。因为变量是 portfolio 对象。

使用 which 命令，可以确定 MATLAB 将调用哪个方法。例如：

```
which pie3
your_matlab_path/toolbox/matlab/specgraph/pie3.m
```

但是，如果 p 是一个 portfolio 对象：

```
which pie3(p)
dir_on_your_path/@portfolio/pie3.m          % portfolio 方法
```

如果传递一个 portfolio 对象作为输入变量，which 命令确定将调用哪一个 pie3 版本。要察看 MATLAB 路径上特定函数所有版本的列表，使用-all 选项。

第 10 章　MATLAB 编程技巧

10.1　命令和函数语法

命令和函数语法方面的编程技巧包括语法帮助、命令和函数语法、命令行续行、用 Tab 键完成命令、重新调用命令、清除命令和屏蔽屏幕输出等。

1．语法帮助

要获取 MATLAB 函数和命令的一般语法信息，在命令行键入：

```
help syntax
```

其中，syntax 为函数名或命令名。比如，要显示 plot 命令的语法信息，在命令行输入：

```
help plot
```

输出：

```
PLOT     Linear plot.
    PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
    then the vector is plotted versus the rows or columns of the matrix,
    whichever line up.    If X is a scalar and Y is a vector, length(Y)
    disconnected points are plotted.

    PLOT(Y) plots the columns of Y versus their index.
    If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
    In all other uses of PLOT, the imaginary part is ignored.
```

(后面的内容省略)

2．命令和函数语法

可以用命令格式和函数格式输入命令。两种格式如下所示：

```
functionname arg1 arg2 arg3                    % 命令格式
functionname('arg1','arg2','arg3')             % 函数格式
```

3．命令行续行

在编程过程中有时候要遇到命令行比较长的情况，这时候往往有命令行续行，即换行显示的要求。在 MATLAB 中，在行末添加省略号(…)可以实现续行。续行有时候能增强程序的可读性。例如：

```
sprintf ('Example %d shows a command coded on %d lines.\n',…
   example_number,…
   number_of_lines)
```

注意，不能对字符串的一部分进行续行。例如，试图对字符串“Hello,MATLAB!”从中间开始换行时导致错误：

```
disp 'Hello,…
```

```
??? disp 'Hello,…
error: missing operator, comma, or semicolon.
```

4．用 Tab 键完成命令

在“面向对象编程”一章中，曾经介绍了一种“偷懒”技巧，在不造成混淆的前提下，设置对象属性时可以只输入属性名的前面几个字母。这里介绍的技巧与之相似：输入有些命令、变量或属性等时，只需要输入名称的前面几个字母，然后单击“Tab”键就可以了。例如，下面的语句行中，第二行用 Tab 键(用“T”表示)将命令名称扩充完整，并显示到第三行中。

```
f = figure;
set(f, 'papTuT','cT)
set(f, 'paperunits','centimeters')
```

如果有多个命令、变量或属性的名称与简写方式想匹配，第一次按“Tab”键将得不到响应，再按一次，显示所有可能的选项。例如，在命令行中键入：

```
pl
```

然后单击“Tab”键时，系统有警告音，屏幕无响应。再单击一次“Tab”键时，显示下面的信息：

```
planerot          plot          plotedit          pltmat
playbackdemo      plot3         plotmatrix        plus
playshow          plot_fhandle  plotyy
```

所有前两个字母为 pl 的命令名称均显示出来了。

5．重新调用命令

在命令窗口重新调用前面的命令有下面几种方法：

- **单击向上箭头键一次或多次，可以显示上一次或上几次键入的命令。可以对显示的命令进行修改后单击回车键运行它。**
- **调用指定的命令时，键入命令名的前面几个字母，然后单击向上箭头键。**
- **打开“Command History”窗口并查看以前的所有命令，然后双击希望运行的那一个。**

6．清除命令

在命令窗口键入了一个命令，却又不想运行它时，可以用“BackSpace”键删除它，但有更快的方法——单击“Esc”键。

7．屏蔽屏幕输出

在语句末尾添加分号(;)，可以防止输出结果显示到屏幕上。在创建大型矩阵时，这个技巧很有用。如

```
A = magic(100);
```

10.2 帮助

MATLAB 帮助方面的技巧包括使用帮助浏览器、函数帮助、主题帮助、分页输出、编写自己的帮助、子函数和私有函数的帮助以及方法和重载函数的帮助等内容。

1．使用帮助浏览器

可以用下面各方式中的任何一种打开 MATLAB 命令窗口：

- **在工具条中单击蓝色的问题标记符。**
- **从“Help”菜单中单击“MATLAB Help”选项。**

- 在命令行键入“doc”。

帮助浏览器的特点可以概括为表 10-1 中的内容。

表 10-1　帮助浏览器的特点

特　　点	描　　述
产品过滤器	确定查找哪个产品的帮助
内容	在内容表中查找主题
索引	用文档索引查找帮助
搜索	用一个或多个单词搜索文档
演示	看哪些演示可用，运行选定的演示
收藏	保存经常使用的帮助页的书签

2．在帮助浏览器中查找函数的帮助

在帮助浏览器中查找任何函数的帮助，可以采用下面方式中的一种：

- 选择帮助浏览器中的“Contents”选项卡，打开标识为“MATLAB”的“Contents”入口并找到两个子入口(Functions—By Category 和 Functions—Alphabetical List)，使用其中一个查找寻求帮助的函数。
- 在命令行中键入：doc <functionname>。

3．从命令行中查找函数的帮助

从命令窗口可以获得如下几种函数帮助：

- 输入 help 命令可以列出所有类别的帮助。
- 键入 help <category>，查看某个类别 category 的函数列表和关于每个函数的简单描述。例如：输入

```
help plot
```

- 键入 help <functionname>，获取基于特定函数的帮助。例如：输入

```
help sortrows
```

4．主题帮助

除了可以获取单个函数的帮助外，在命令行中键入 help <topicname>，还可以获取关于表 10-2 中所示的主题的帮助。

表 10-2　主题帮助及其描述

主　题　名	描　　述
arith	算术操作符
relop	关系和逻辑操作符
punct	特殊字符操作符
slash	算术除操作符
paren	括号操作符
precedence	操作符优先级
lists	逗号分隔的列表
strings	字符串
function_handle	函数句柄和@操作符
debug	调试函数
java	在 MATLAB 中使用 Java
fileformats	可读文件格式的列表
change_notification	改变通告信息的 Windows 目录

5．分页输出

显示帮助文本或代码的一个很长的段落以前，在命令行中键入 more on，把 MATLAB 设置为分页输出模式。这样可以分页显示，便于查看。在命令行中键入 more off，取消分页模式输出。

单击空格键继续显示分页；单击回车键，逐行显示，单击“Q”键或同时按下“Ctrl”键和“C”键，取消连续显示。

6．编写自己的帮助

作为比较好的编程习惯，往往在程序的开始部分写一些文本，提供关于如何使用函数的帮助信息。如果格式正确，在命令行中键入 help <functionname>命令时会显示该文本。

MATLAB 把紧接函数定义行的以%符号开头的第一组连续行作为函数的帮助文本。假设有下面的小程序：

```
function y=tentimes(x)
% 本程序计算输入值与 10 的乘积
% 用于演示 MATLAB 中函数帮助的实现方法
y=x*10;
```

把它保存到 MATLAB 的 work 目录中，M 文件名为 tentimes.m。在命令窗口键入

```
help tentimes
```

本程序计算输入值与 10 的乘积。

用于演示 MATLAB 中函数帮助的实现方法，键入

```
y=tentimes(10)
```

得到返回值

```
y =
    100
```

7．子函数和私有函数的帮助

可以用上面的方法给子程序编写帮助文本。要显示文件 myfun.m 中子函数 mysubfun 的帮助信息，键入

```
help myfun/mysubfun
```

要显示私有函数的帮助信息，在函数名前面添加“private/”。比如，要获取私有函数 myprivfun 的帮助信息，键入

```
help private/myprivfun
```

8．方法和重载函数的帮助

可以为面向对象的类方法编写帮助文本，这些类方法用文件实现。键入下面的命令行，显示类方法的帮助信息：

```
help classname/methodname
```

其中，文件 methodname.m 位于子目录@classname 中。

例如，如果为一个 polynom 类编写一个 plot 方法，其中 plot 方法定义在文件 @polynom/plot.m 中，可以通过键入下面的命令行来显示帮助信息：

```
help polynom/plot
```

可以用相同的方式获取重载的 MATLAB 函数的帮助。例如，要显示 matlab/iofun/@serial 中实现的 eq 函数的帮助文本，键入

```
help serial/eq
```

10.3 开发环境

开发环境方面的技巧包括工作空间浏览器、使用查找和替换工具、注释一个代码块、从命令历史窗口中创建 M 文件以及在 EMACS 中编辑 M 文件。

1．工作空间浏览器

使用浏览器，可以利用工作空间中的数据查看、修改、保存、载入和创建图形。在“ View ”菜单中单击“ Workspace ”选项，打开浏览器。

2．使用查找和替换工具

用查找和替换工具在一组文件中查找任何单词或短语。

3．注释一个代码块

按照下面的步骤把 MATLAB 编辑器中一个代码块快速地变成注释文本。

- 在编辑器中亮显要注释的代码块。
- 在“ Text ”菜单中单击“ Comment ”选项。先前被选中的文本块各行前面自动添加了“ % ”符号，代码变成绿色，成为注释项。

选择注释文本，然后在“ Text ”菜单中单击“ UnComment ”选项，可以将注释文本变成非注释项。

4．从命令历史窗口中创建 M 文件

利用“ Command History ”窗口，可以把当前 MATLAB 的一些命令行放到 M 文件中去。按照下面的步骤进行操作：

- 在“ View ”菜单中选择“ Command History ”选项，打开窗口。
- 在“ Command History ”窗口中选择要使用的行，MATLAB 亮显这些行。
- 右击鼠标键，从弹出的菜单中选择“ Create M-File ”选项。MATLAB 创建一个新的编辑器窗口，显示选定的代码。

10.4 M 文件函数

M 文件函数方面的技巧包括 M 文件结构、函数名用小写、获取函数的名称和路径、函数使用什么 M 文件、函数运行所依赖的函数、内部函数和类的信息等内容。

1．M 文件结构

M 文件的比较标准的格式如下：

```
function [x, y] = myfun(a, b, c)              函数定义行
% H1 行 —— 用一行文字来综述函数的功能
% 帮助文本 —— 用一行或多行文本解释如何使用函数
% 在命令行中键入"help <functionname>"时可以使用它


% 函数体——一般从第一个空白行后开始
% 注释 —— 描述函数的行为，输入/输出的类型等
% 在命令行中键入"help <functionname>"时不会显示这些文本
```

```
x = prod(a, b);                    % 开始编写函数代码
```

所以，M 文件应该包括函数定义行、H1 行、帮助文本、函数体、注释和函数代码等项目。

2．函数名用小写

函数名在 MATLAB 帮助文本中大写只是为了好读，但是，在实际运用中，调用函数时最好使用小写。特别是 MATLAB 调用任何内部函数时要求使用小写。对于 M 文件函数而言，是使用大写还是使用小写，取决于正在使用的系统的大小写敏感性。一般地讲，使用小写的 M 文件可以适应更多的操作系统。

3．获取函数的名称和路径

要获取当前正在运行的 M 文件的名称，在 M 文件代码中使用下面的函数：

```
mfilename
```

使用下面的语句包括路径名和 M 文件名：

```
mfilename('fullpath')
```

4．函数使用什么 M 文件

按照下面的步骤显示特定函数用到的所有 M 文件：

- 键入 clear functions，从内存中清除所有函数。
- 运行要检查的函数。注意，此时选用的函数变量很重要，因为对于相同的函数，如果变量不同，可能会得到不同的结果。
- 键入 inmem，显示函数运行时用到的所有 M 文件。如果想看看还使用了哪些 MEX 文件，指定一个额外的输出，如下所示：

```
[mfiles,mexfiles]=inmem
```

5．函数运行所依赖的函数、内部函数和类的信息

使用 depfun 函数，可以显示所依赖的函数的详细信息。除了 M 文件，depfun 函数还显示所依赖的内部函数和类的信息。

10.5 函数变量

本节介绍包括获取输入和输出变量、变量个数、字符串或数值变量、用结构传递变量、用单元数组传递变量等内容。

1．获取输入和输出变量

使用 nargin 和 nargout 在函数调用时确定输入变量和输出变量的个数。使用 nargchk 和 nargoutchk 确认函数用必要个数的输入变量和输出变量进行调用。例如：

```
function [x, y] = myplot(a, b, c, d)
disp(nargchk(2, 4, nargin))          % 允许 2～4 个输入
disp(nargoutchk(0, 2, nargout))      % 允许 0～2 个输出


x = plot(a, b);
if nargin == 4
    y = myfun(c, d);
end
```

2．变量个数

调用函数时，输入、输出变量的个数可以比函数定义中指定的少，但不能多。如果希望调用一个变量个数可变的函数，在函数定义中使用 varargin 和 varargout 函数参数。比如，下面的函数返回 size 向量和一个可选变量。

```
function [s, varargout] = mysize(x)
nout = max(nargout, 1) - 1;
s = size(x);
for k = 1:nout
    varargout(k) = {s(k)};
end
```

下面对它进行调用：

```
[s, rows, cols] = mysize(rand(4, 5))
```

3．字符串或数值变量

如果只传递一个字符串变量到函数，可以使用 MATLAB 命令格式。所有按命令格式输入的变量被解释为字符串。例如：

```
strcmp string1 string1
ans =
     1
```

传递数值变量时，最好使用函数格式，除非希望数字作为字符串传递。下面右侧的例子把数字 75 作为字符串“75”进行传递。

```
isnumeric(75)                    isnumeric 75
ans =                            ans =
     1                                0
```

4．用结构传递变量

在 MATLAB 中可以传递结构。使用结构，可以在不必修改函数的情况下修改变量的个数、内容和次序。在有需要大量近似信息的函数时，结构尤其有用。

5．用单元数组传递变量

还可以把变量组合到单元数组中。使用结构的缺点之一是不能用字段名描述每个变量，而单元数组按编号引用，这样可以通过循环获取函数的每一个输入、输出变量。

10.6 程序开发

本节介绍包括程序规划、使用伪代码、选择正确的数据结构、一般编码实践、惟一的函数名、注释的重要性、按步骤编码、按步骤进行修改、有一个调用函数的函数、测试最终形成的程序等内容。

1．程序规划

规划编程时，首先要将整个问题分解成若干个小的独立的任务，并用单独的函数来实现这些任务。要让函数尽可能的短，每个函数只有一个功能。

2．使用伪代码

在正式编程之前，用自己的语言写一个结构化的程序框架很有用。这个程序框架通常称为伪代码。使用伪代码查看、修改程序比使用正式的程序代码来完成往往要容易一些。而且

有了伪代码以后，可以很容易地将它翻译为某种编程语言的代码。

3．选择正确的数据结构

在 MATLAB 中查找，看哪些数据类型和数据结构是可用的，并确定其中哪些用在自己的程序中保存和传递数据时最合适。

4．编码建议

在具体编程过程中，有下面一些建议：

- 使用描述性函数名和变量名，使得代码的可读性更强。
- 在 M 文件中按子函数名称的字母顺序确定子函数的位置，这样更便于查找。
- 在每个子函数前面放置一段描述函数功能的帮助文本。这样做不仅解释了子函数，还可以把它们很明显地区分开。
- 代码行的宽度不要超过 80 列。否则，把它打印出来时将很难阅读。
- 使用完整的句柄图形属性名和值名。

5．惟一的函数名

为了避免出现新函数名称与一个已经存在的函数的名称冲突的情况发生，使用下面的命令查看已有的函数名。

```
which -all <functionname>
```

6．按步骤编码

不要试图一次把整个程序编完。把整个程序分成很多部分以后，首先编写其中的一部分，编号后进行测试，如果合格了再编写余下的部分。与大程序相比，在小程序中更容易发现错误。

7．按步骤进行修改

修改正在工作的程序时，不要一次修改很多地方，最好每次做比较少的修改，然后测试和调试。没问题以后再进行下一步修改。

8．有一个调用函数的函数

如果有一个只被另外一个函数调用的函数，把它作为调用函数放到相同的 M 文件中，将它作为子函数。

9．测试最终程序

测试一个新程序时，建议按照逐步运行的方式，在 MATLAB 调试器中测试整个程序。并且在测试过程中记下每一行的运行情况。使用不同组合的输入进行测试，直到程序中的每个代码行至少运行了一次。

10.7 调试

调试技巧包括 MATLAB 调试函数、MATLAB 图形调试器、快速检查变量、在命令行中设置断点、找到行号、发生错误或警告时停止运行、从出错信息中定位错误、使用警告来帮助调试、使代码的运行可视以及调试脚本等。

1．MATLAB 调试函数

在命令行键入 help debug，有一个关于调试函数的简洁描述。

2．更多调试函数

表 10-3 所示的函数对于调试比较有用。

表 10-3　其他调试函数

函　数	描　述
echo	运行时显示函数或脚本代码
disp	显示指定的值或信息
sprintf fprintf	显示不同类型的格式化数据
whos	在工作空间中列出变量
size	显示数组的维
keyboard	中断程序运行并允许从键盘输入
return	键盘中断以后继续运行
warning	显示指定的警告信息
error	显示指定的出错信息
lasterr	返回最后出现的出错信息
lasterror	返回最后的出错信息和相关信息
lastwarn	返回最后出现的警告信息

3．MATLAB 图形调试器

学习使用 MATLAB 图形调试器。

可以在调试时察看函数和它的子函数，设置和清除断点。在程序中逐步运行，进入或跳过调用的函数，控制工作空间中的可见性并在文件中查找和替换字符串。

首先，在“ File ”菜单中单击“ Open ”选项或在命令窗口使用 open 函数打开要调试的文件。用工具条上可用的调试函数或下拉菜单设置断点，运行或在程序中逐步运行，并检查变量。

4．快速检查变量

在编辑器或调试窗口中用鼠标光标指着变量名，稍等片刻就可以看到变量的值。

5．从命令行中设置断点

可以用 dbstop 命令，采取下面任意一种方式设置断点：

（1）在指定的 M 文件行号处设置。

（2）在指定函数的开始处设置。

（3）在 M 文件的第一个可执行行处设置。

（4）当生成警告、错误时设置。

（5）遇到任何无限值或空值时设置。

6．查找设置断点的行号

从命令行调试时，一个查找设置断点的行号快捷方法是使用 dbtype 函数。该函数显示 M 文件的全部或一部分，并给每一行编号，用下面的命令显示 copyfile.m：

```
dbtype  copyfile
```

用下面的语句显示第 70 行到第 90 行之间的内容：

```
dbtype  copyfile  70:90
```

7．在错误或警告发生处停止运行

使用 dbstop if error 语句，在发生任何错误时停止运行并进入调试模式。

使用 warning debug 语句，在出现任何警告信息时停止运行并进入调试模式。

8．从出错信息中定位错误

在出错信息中单击加了下划线的文本。MATLAB 在编辑器中打开运行的 M 文件，并把鼠标放在发生错误的地方。

9．用警告帮助调试

在 MATLAB 程序中插入警告信息，可以帮助发现错误或意料之外的行为。

10．让代码的运行结果可见

查看菜单的运行结果的最简便方法是删除该行最后的分号，然后，运行程序时计算结果显示在屏幕上。

11．调试脚本

脚本把它们的变量保存在工作空间中，该工作空间对于脚本的调用者来说是共享的。所以，从命令行调试脚本时，脚本使用源于基本工作空间的变量。为了避免工作空间共享导致的错误，在开始调试脚本以前键入 clear all，清除基本工作空间。

10.8 变量

变量编程技巧包括变量命名规则，确定变量名的合法性，不使用与函数名相同的变量名。检查关键字，避免把 i 和 j 用作变量，避免覆盖脚本中的变量，persistent 变量，防止 persistent 变量以及全局变量。

1．不要把函数名用作变量

命名变量时，确定该变量的名称没有用作函数名。否则，在没有从内存中清除该变量的情况下，将不能调用该函数（除非用 builtin 运行该函数）。

要测试变量名是否已经用作函数名，键入：

```
which –all <name>
```

2．检查预留的关键字

MATLAB 预留了一些关键字并且不允许重载它们，如果调用、使用它们，可能会导致类似于下面的出错信息：

```
Error: Expected a variable, function, or constant, found "=".
Error: "End of Input" expected, "case" found.
Error: Missing operator, comma, or semicolon.
Error: "identifier" expected, "=" found
```

用 iskeyword 函数（不带输入变量）列出所有预留的关键字。

3．避免把 i 和 j 用作变量

MATLAB 用字符 i 和 j 表示虚数单位。如果涉及到复数计算，应避免把 i 和 j 用作变量名。

4．避免覆盖脚本中的变量

MATLAB 把变量保存在工作空间中，该工作空间被脚本的调用者共享。从命令行中调用时，它们共享基本工作空间；从函数调用时，共享函数的工作空间。如果运行一个改变已经存在于调用者工作空间的变量的脚本，该变量被脚本覆盖。

使用 persistent 命令，可以在 MATLAB 中获得静态变量。在函数中把一个变量声明为 persistent 时，与全局变量不同的是，persistent 变量只对声明了它们的函数已知。

5．全局变量

要尽量少用全局变量，全局变量在整个程序范围内都可用。所以，使用的全局变量越多，无意中重复使用变量名的机会就越大，这样，就很容易在无意中改变变量的值，从而导致错误的结果，更糟糕的是，这种错误还很难捕获。

10.9 字符串

字符串方面的技巧包括通过聚合创建字符串，比较聚合的方法，把字符数组以单元数组的形式保存，用正则表达式搜索和替换，在字符串和单元数组之间转换。

1．通过聚合创建字符串

字符串通常可以由更小的元素聚合而成。两个通用的聚合方法是使用 MATLAB 聚合操作符（[]）或 sprintf 函数。

下面程序的第二行和第三行演示了这两个方法，两行给出相同的结果。

```
num_chars = 28;
s = ['There are ' int2str(num_chars) ' characters here']
s = sprintf('There are %d characters here\n', num_chars)
```

2．比较聚合的方法

用聚合方法创建字符串时，用 sprintf 函数比使用[]好，因为：

（1）更好读，特别是在组成复杂表达式的时候。

（2）对输出格式有更多控制。

（3）一般运行更快。

也可以用 strcat 函数聚合字符串，但对于简单的聚合，使用 sprintf 和[]更快。

3．以单元数组的形式保存字符串数组

通常，最好将字符串数组以单元数组的形式保存，而不是以字符数组方式保存，特别是在字符串的长度不同的情况下。

字符数组中的字符串长度必须相等，这常常需要在字符串后面添加空格。如果使用字符串单元数组就没有这个必要了。例如，下面的 cellRecord 单元数组就不需要在字符串后面添加空格。

```
charRecord  = ['Allison Jones'; 'Development  '; 'Phoenix      '];
cellRecord = {'Allison Jones'; 'Development'; 'Phoenix'};
```

4．在字符串和单元数组之间转换

使用 cellstr 函数和 char 函数，可以在标准的字符数组和字符串单元数组之间转换。例如：

```
charRecord = ['Allison Jones'; 'Development  '; 'Phoenix      '];
cellRecord = cellstr(charRecord);
```

同样，可以用字符数组或（和）单元数组进行大量的 MATLAB 字符串操作。例如：

```
cellRecord2 = {'Brian Lewis'; 'Development'; 'Albuquerque'};
strcmp(charRecord, cellRecord2)
ans =
     0
     1
     0
```

5．评价表达式

评价表达式包括替换 eval 函数，指定一系列变量，Short-Circuit 逻辑操作符和在 for 循环中改变计数变量等内容。

6．替代 eval 函数

编程时要尽量少用 eval 函数，主要原因是：使用 eval 函数的代码通常不好读，而且不易调试。第二个原因是，有时候不能被 MATLAB 编译器翻译为 C 或 C++代码。

评价一个函数，使用 feval 函数比使用 eval 函数的效率更高。

7．给一系列变量赋值

创建变量的一个通用模式是在一个变量名后面添加连续的数字后缀（例如，phase1，phase2，phase3 等），建议采用单元数组来创建这种类型的变量名序列。采用这种方式，使代码的可读性更强，运行更快。例如：

```
for k=1:800
   phase{k}=<expression>;
end
```

8．改变 for 循环中的计数变量

不能在 for 循环体中改变循环计数变量的值。例如，即使在每次迭代时把 k 的值设置为 1，循环也只运行 10 次。

```
for k=1:10
   disp(sprintf('pass %d',k))
   k=1;
end
```

尽管 MALTAB 允许在循环体中使用与循环计数变量相同名称的变量，但不建议这样做。

10.10 MATLAB 路径

1．优先法则

给 MATLAB 指定一个名称进行解释时，它按照下面的顺序将该名称进行核对，以确定名称的用途。

（1）变量。

（2）内部函数。

（3）子函数。

（4）私有函数。

（5）类的构造函数。

（6）重载方法。

（7）当前目录中的 M 文件。

（8）路径上的 M 文件。

如果发现该名称是路径上的 M 文件的名称，而且路径中有多个相同的 M 文件的名称，MATLAB 将使用最靠近路径起始位置的那一个。

2．文件优先级

如果只通过文件名来引用文件（没有扩展名），而且目录中有一个以上的文件具有该名

称，MATLAB 将按照下面的次序来使用文件：

（1）MEX 文件。

（2）MDC 文件（仿真模型）。

（3）P 代码文件。

（4）M 文件。

3．给搜索路径添加目录

用下面方式中的一种给搜索路径添加目录：

（1）在工具条中，选择 File-SetPath。

（2）在命令行中，使用 addpath 函数。

还可以用上面方式中的一种在一次操作中添加一个目录和它的所有子目录。在命令行中进行此项操作时，联合使用 genpath 函数和 addpath 函数。

下例是把/control 和它的所有子目录添加到 MATLAB 路径中：

```
addpath(genpath('k:/toolbook/control'))
```

4．不在路径中的函数的句柄

不能给不在 MATLAB 路径中的函数创建函数句柄。但是，通过在该函数所在的目录中添加一个脚本文件来创建函数句柄。用 run<path>/<script>运行脚本以后，就创建了所需要的函数句柄。例如：

（1）在路径外函数所在的目录中创建类似于下面的脚本：

```
file E:/testdir/create_fhandles.m
fhset =@set_items
fhsort =@sort_items
fhdel =@delete_item
```

（2）从当前目录中运行脚本，创建函数句柄：

```
run E:/testdir/create_fhandles
```

（3）用 feval 函数，通过它的句柄运行一个函数：

```
feval(fhset,item,value)
```

5．让工具箱文件的改变对于 MATLAB 可见

与用户自定义目录中的函数不同，＜matlab＞\foolbox 目录中的 M 文件不会马上得到检查。所以，MATLAB 不会自动察觉它们的改变。如果修改其中一个文件，然后返回到目录中，会发现目录中的内容并没随着改变。因为 MATLAB 还在使用先前载入的文件版本。

要迫使 MATLAB 从磁盘中重新载入函数，需要用 clear<frnctionname>语句从内存中清除该函数。

类似地，MATLAB 不会自动发现＜matlab＞\foolbox 目录中的新文件。如果从这些目录中添加或删除文件，用 rehash foolbox 语句可以察看改变后的显示。注意，如果使用 MATLAB 编辑器创建文件，编辑器会自动把这些改变通知给 MATLAB。

6．使非工具箱文件对于 MATLAB 可见

对于工具箱目录以外的 M 文件，MATLAB 通过比较函数 timestamps 来察觉它们的改变，并在下次运行对应函数时重新载入这些文件。

如果 MATLAB 没有察觉这些改变，用 clear<functionname>语句把原函数从内存中清除出去。可以用 inmem 函数列出载入内存的所有函数，以便确认是否已经删除原函数。

10.11 程序控制

1．使用 break，continue 和 return 函数

break，continue 和 return 函数比较容易混淆，表 10-4 中对它们进行了比较详细的比较。

表 10-4 函数 break，continue 和 return 之间的区别

函 数	用 在 何 处	描 述
break	for 或 while 循环	在它出现的时候，退出循环，在嵌套的循环中，进入相邻的外层循环
continue	for 或 while 循环	在本循环中跳过剩余的语句，进入本循环的下一次迭代
return	任意位置	它出现时，立即退出函数，进入函数的调用函数

2．switch 与 if 相比较

表 10-5 比较 switch/case 语句和 if/elseif 语句之间的区别。

表 10-5 switch/case 语句和 if/elseif 语句之间的区别

switch/case 语句	if/elseif 语句
更好读	更难读
可以比较不同长度的字符串	需要 strcmp 函数比较不同长度的字符串
只测试相等性	测试相等性或不等性

3．MATLAB 的 case 语句中可以使用字符串

C 语言和 MATLAB 中都有 switch/case 语句，但二者有所不同，在 MATLAB 中，case 语句可以指定字符串的值，而在 C 中不能。例如：

```
switch(method)
  case'linear'
    disp('Method is linear')
  case'cubic'
    disp('Method is cubic')
end
```

4．Case 语句的多种情况

可以用 switch 语句测试一种以上的情况。下面的例子是在 case 语句中用单元数组测试一个 linear 方法或 bilinear 方法。

```
switch(method)
  case{'linear', 'bilinear'}
    disp('Method is linear or bilinear')
    …
end
```

5．switch/case 语句中的隐含中断

在 C 代码中，如果在每个 case 块中不用 break 语句终止，代码的运行会进入到下一个 case 块。在 MATLAB 中，不会出现这种情况，而只会运行一个 case 块。在 case 块中使用 break 语句，不仅不必要，而且非法，会产生警告信息。

下面的例子中，如果 result 等于 52，则只有第一个 disp 语句运行。虽然第二个 case 也符

合要求，但它不会运行。

```
switch(result)
   case 52
     disp('result is 52')
  case {52,78}
     disp('result is 52or78')
end
```

6．switch 语句中的变量范围

因为 MATLAB 只运行 switch 语句中的一个 case 块，所以一个 case 块中定义的变量在同一个 switch 语句中的另一个 case 块中是不可知的。if/elseif 语句也存在同样的情况。

下面的例子中，当 choice 等于 2 时，产生错误，因为 x 没有定义。

```
--SWITCH/CASE--                    --IF/ELSEIF—
  switch choice
      case 1                       if choice==1
        x=-pi:0.01:pi;             x=-pi:0.01:pi;
      case 2                       elseif choice==2
        plot(x,sin(x));            plot(x,sin(x));
  end                               end
```

7．用 try/catch 捕获错误

如果发现代码中某些语句导致运行结果不正确，可以考虑将这些语句放到一个 try/catch 块中去，这将捕获到所有错误，并可以合理地控制它们。

下面的例子是在一个 2 个矩阵相乘的函数中显示一个 try/carch 块。如果块中的 try 部分中的语句运行失败，进入 catch 部分。此时，catch 语句检查出错信息＜由 lasterr 返回＞并作出相应的反应。

```
try
  X=A*B
catch
  errmsg=lasterr
  if(strfind(errmsg, 'inner matrix dimens I ons'))
  disp('**Wrong dimens I ons for matrix multiply')
end
```

8．嵌套的 try/catch 块

还可以像下面这样嵌套 try/catch 块。

```
try
  statement1                运行语句
catch
  try
  statement2              试图补救错误
catch
  disp'Operation failed'  控制错误
  end
```

```
end
```

9．迫使从函数中提早返回

要迫使从函数中提早返回，在函数中希望退出的地方放置 return **语句。例如：**

```
if<done>
  return
end
```

10.12 保存和载入

1．从工作空间中保存数据

按照下面任何一种方法操作，可从工作空间中保存数据：

（1）从命令窗口中复制，然后粘贴到文本文件中。

（2）在 diary **文件中记录，然后在文本编辑器中编辑。**

（3）用 save **函数保存为二进制或** ASCII **文件。**

（4）用合适的函数保存电子表格、文本文件、图像或声音数据。

（5）用低级 I／O **函数（**fwrite,fprintf **等）保存文件。**

2．把数据载入工作空间

要把新数据或已经保存的数据载入工作空间，可在下面的方法中选择一种来完成：

（1）在命令行输入或粘贴数据。

（2）创建一个脚本文件，初始化大型矩阵或数据结构。

（3）用 load **函数读一个二进制或** ASSII **文件。**

（4）用合适的函数载入电子表格文本、图像或声音数据。

（5）用 I/O **函数（**fread，fsranf **等）载入文件。**

3．在 MAT **文件中察看变量**

用下面的 who **或** whos **函数察看** MAT **文件中保存了哪些变量（不需要**.mat **扩展名）。**who **函数返回一个单元数组，**whos **函数返回一个结构数组。**

```
mydata_variables=who('-file', 'mydata.mat')
```

4．添加到 MAT **文件**

要把一个额外的变量保存到已经存在的 MAT **文件中，使用下面的语句：**

```
save<matfi lename>-append
```

保存 MAT **文件中不存在的变量时，把它添加到文件中。保存任何已经存在的变量时，覆盖原来的变量。**

下面的例子中第二个 save **操作不会把新元素聚合到** A **中（使** A **等于**[1 2 3 4 5 6 7 8]**），而是用一个** 3 **元素向量取代** 5 **元素向量，并保留所有第一次** save **操作中已经保存的其他变量。**

```
A=[1 2 3 4 5];      B=12.5;      C=rand(4);
save savefile;
A=[6 7 8]
Save savefile A-append;
```

5．在启动或退出时载入或保存变量

创建一个 finish.m **文件，在每次退出** MATLAB **时保存基本工作空间中的内容，可以在终**

止 MATLAB 运行时自动保存变量。

同样，通过创建一个 startup.m 文件，使用 load 函数从 MAT 文件中载入变量，可以在启动 MATLAB 时自动把先前保存的变量载入到工作空间中。

6．保存到 ASCII 文件

用 save –ascii 命令把矩阵数据保存到 ASCII 文件时，MATLAB 把单个矩阵组合到一个数字集合中去，不保存变量名。如果应用程序不能接受，用 fprintf 函数保存数据。

7．命名 M 文件

M 文件名必须以字母打头，可以包含任何阿拉伯数字或下划线，并且，长度不能超过 M 文件名的最大长度（用 name lengthmax 函数指定）。例如：

```
N＝name lengthmax
N=
    63
```

因为变量必须遵守相似的原则，可以使用 isvarname 函数检查文件名去掉扩展名以后是不是合法的 M 文件名。即：

```
isvarname<M-File name>
```

8．命名其他文件

MATLAB 中 M 文件以外的文件（如 MAT 文件、MEX 文件、MDL 文件等）的名称与 M 文件名具有相同的命名规则，但长度不能任意。

9．把文件名作为变量

在 MATLAB 中，可以用 MATLAB 命令或函数语法指定一个文件名变量。例如，下面两种情况都是允许的：

```
load mydata.mat          % 命令格式
load('mydata..mat')      % 函数格式
```

如果把输出指定给变量，必须使用函数语法。即：

```
saved_data=load('mydata.mat')
```

10．把文件名传递给 ASSII 文件

以下面的形式指定 ASSII 文件，文件扩展名不可省略。

```
load mydata.dat-asrii            % 命令格式
load('mydata.dat', '-ascii')     % 函数格式
```

11．运行时确定文件名

在不把文件名写入程序代码的情况下，函数有以下几种方式基于指定文件进行工作。

（1）把文件名作为变量传递。

```
function myfun(datafile)
```

（2）用 input 函数提示输入文件名。

```
filename=input('Enter name of file: ', 's')
```

（3）用 uigetfile 函数浏览文件。

```
[filename,pathname]=uigetfile('*.mat', 'Select MAT-file')
```

12．返回文件的大小

有两种方法可以确定文件的大小。

```
-- 方法 1 --                    -- 方法 2 --
s = dir('myfile.dat');          fid = fopen('myfile.dat');
filesize = s.bytes              fseek(fid, 0, 'eof');
                                filesize = ftell(fid)
                                fclose(fid);
```

dir 函数还返回文件名（s.name），最后修改日期（s.date）和它是否是一个目录（s.isdir）。第二个方法需要从文件中读取。

10.13 输入和输出

1．通用 I/O 函数

I/O 函数在 MATLAB 中最常用。而且，最高级的 I/O 函数是 save 函数和 load 函数。键入 doc save 或 doc load，获取帮助。

2．读取文件格式

键入 doc fileformats，察看 MATLAB 中可读的文件格式列表，以及相关的 MATLAB 函数。

3．载入混合格式的数据

用 textread 函数取代 load 函数，载入混合格式的数据。textread 函数允许指定每条数据的格式。

如果文件 mydata.dat 的第一行为

```
Sally     12    34    45
```

用%格式把第一行读成自由格式，即：

```
[name,x,y]=textread('mydata.dat', '%s %f %d', 1)
```

返回

```
name=
    'Sally'
x=
   12.34000000000000
y=
    45
```

4．把 ASSII 数据读入单元数组

下面介绍一个把 ASSII 数据读入单元数组的通用技巧。

```
[a,b,c,d]=textread('data.txt', '%s %s %s %s')
mydata=cellstr([a b c d ])
```

5．交互式输入数据

程序可以在运行时接收交互输入。用 input 函数提示用户输入数据，并暂停运行，输入后用键盘或其他方式告诉计算机（如单击回车键）。然后，计算机读入数据，继续运行。

附录　常用命令与函数

表 1　特殊变量名表

变量名称	变量含义	变量名称	变 量 含 义
ans	MATLAB 中默认变量	i(j)	复数中的虚数单位
pi	圆周率	nargin	所用函数的输入变量数目
eps	计算机中的最小数，PC 机上为 2^{-52}。	nargout	所用函数的输出变量数目
inf	无穷大，如 1/0	realmin	最小可用正实数
NaN	不定值，如 0/0，∞/∞，0*∞	realmax	最大可用正实数

表 2　运算符与操作符

函数分类	函数名	说　　明	函数分类	函数名	说　　明
运算符	+	加法	关系操作函数	eq(A,B)	等于
	-	减法		ne(A,B)	不等于
	*	矩阵乘法		lt(A,B)	小于
	.*	数组乘法		gt(A,B)	大于
	^	矩阵乘方		le(A,B)	小于等于
	.^	数组乘方		ge(A,B)	大于等于
	\	矩阵左除	逻辑运算符	&	逻辑与
	/	矩阵右除		|	逻辑或
	.\	数组左除		~	逻辑非
	./	数组右除	逻辑运算函数	and(A,B)	逻辑与
关系操作符	= =	等于		or(A,B)	逻辑或
	~ =	不等于		not(A)	逻辑非
	<	小于		xor(A,B)	逻辑异或。A 与 B 都非零或都是零时返回 0；有一个非零则返回 1
	>	大于		any(A)	向量 A 中有非零元素时返回 1；矩阵 A 中某一列有非零元素时，此列返回 1
	<=	小于等于		all(A)	向量 A 中所有元素非零时返回 1；矩阵 A 中某一列所有元素非零时，此列返回 1
	>=	大于等于			

表 3　基本数学函数

函数分类	函数名	说　明	函数分类	函数名	说　明
三角函数	sin	正弦函数	三角函数	asech	反双曲正割函数
	sinh	双曲正弦函数		cot	余切函数
	asin	反正弦函数		coth	双曲余切函数
	asinh	反双曲正弦函数		acot	反余切函数
	cos	余弦函数		acoth	反双曲余切函数
	cosh	双曲余弦函数	其他常用计算函数	fix	向零方向取整
	acos	反余弦函数		round	四舍五入到最近的整数
	acosh	反双曲余弦函数		floor	向无穷大方向取整
	tan	正切函数		rem	求两整数相除的余数
	tanh	双曲正切函数		exp	指数函数
	atan	反正切函数		log	自然对数函数
	atanh	反双曲正切函数		log10	以 10 为底的对数函数
	sec	正割函数		sort	开方函数
	sech	双曲正割函数		abs	绝对值函数
	asec	反正割函数			

表 4　测试函数

函　数　名	说　明
isempty(A)	若参量 A 为空，返回 1；否则，返回 0
isglobal(A)	若参量 A 为全局变量，返回 1；否则，返回 0
ishold	当前绘图状态保持是 ON，返回 1；否则，返回 0
isieee	计算机执行 IEEE 运算，返回 1；否则，返回 0
isinf(A)	若参量 A 无穷大，返回 1；否则，返回 0
isletter(A)	若参量 A 为字母，返回 1；否则，返回 0
isnan(A)	若参量 A 为不不定值，返回 1；否则，返回 0
isreal(A)	若参量 A 无虚部，返回 1；否则，返回 0
isspace(A)	若参量 A 为空格字符，返回 1；否则，返回 0
isstr(A)或 ischar(A)	若参量 A 为一个字符串，返回 1；否则，返回 0
isstudent(A)	若 MATLAB 为学生版，返回 1；否则，返回 0
isunix(A)	若计算机为 UNIX 系统，返回 1；否则，返回 0
isvms(A)	若计算机为 VMS 系统，返回 1；否则，返回 0

表 5　矩阵函数

函 数 分 类	函　数　名	说　明
常用矩阵函数	cond	矩阵的条件数值
	condest	1-范数矩阵条件数值
	cross	矩阵的叉积
	det	矩阵的行列式值
	dot	矩阵的点积

续表

函数分类	函数名	说明
	eig	矩阵的特征值
常用矩阵函数	eigs	矩阵的特征值
	inv	矩阵的逆
	norm	矩阵的范数值
	normest	矩阵的 2-范数值
	rank	矩阵的秩
	orth	矩阵的正交化运算
	rcond	矩阵的逆条件数值
	trace	矩阵的迹
	triu	上三角变换
	tril	下三角变换
	diag	对角变换
	exmp	矩阵的指数运算
	exmp1	Pade 法进行矩阵的指数运算
	exmp2	Taylor 级数进行矩阵的指数运算
	logm	矩阵的对数运算
	sqrtm	矩阵的开方运算
	cdf2rdf	复数对角形矩阵转换成实数块对角形矩阵
	rref	转换成逐行递减的阶梯矩阵
	rsf2csf	实数块对角形矩阵转换成复数对角形矩阵
	rot90	矩阵逆时针方向旋转
	fliplr	左、右翻转矩阵
	flipud	上、下翻转矩阵
	reshape	改变矩阵的维数
	funm	一般的矩阵函数
矩阵分解函数	chol	矩阵的 Cholesky 分解
	eig	矩阵的特征值分解
	hess	矩阵的 Hessenberg 分解
	lu	矩阵的 LU 分解
	null	奇异值分解求得的矩阵的零空间的标准正交基
	qr	矩阵的 QR 分解
	qz	矩阵广义特征值的 QZ 分解
	schur	矩阵的 Schur 分解
	svd	矩阵的奇异值分解
	svds	矩阵的奇异值分解

表 6　稀疏矩阵函数

函数名	说明	函数名	说明
sparse	最常用的稀疏矩阵生成函数	spones	用 1 代替非零元素
spdiags	以对角带生成稀疏矩阵	sprank	稀疏矩阵的秩

续表

函数名	说　明	函数名	说　明
spconvert	由外部数据输入生成稀疏矩阵	spy	显示稀疏矩阵的结构图
find	求出非零元素在稀疏矩阵中的索引	issparse	判断是否是稀疏矩阵
speye	生成稀疏单位矩阵	colmmd	列最小度
sprand	生成稀疏的均匀分布随机矩阵	colperm	按列排列向量
sprandn	生成稀疏的正态分布随机矩阵	dmperm	矩阵的 DuITnageMendelsohn 分解
sprandsym	生成稀疏的对称随机矩阵	randperm	随机排列向量
full	把稀疏矩阵转化为满矩阵	symmmd	最小对称度
nnz	稀疏矩阵非零元素的个数	gplot	按图论画图
nonzeros	稀疏矩阵非零元素的值	etree	给出矩阵的消元树
nzmax	存放非零元素所需的内存	etreeplot	画消元树图
spalloc	为非零元素分配内存空间	treeplot	画结构树图
spfun	求非零元素的函数值		

表 7　特殊矩阵

矩阵名称	说　明
[]	空矩阵
zeros	零矩阵
ones	1 矩阵
eye	单位矩阵
rand	均匀分布的随机矩阵
randn	正态分布的随机矩阵
compan	伴随矩阵
magic	魔方矩阵
hilb	Hibert 矩阵
invhilb	逆 Hibert 矩阵

表 8　多项式运算函数

函数名	说　明
conv	多项式乘法
deconv	多项式除法
poly	用根生成多项式
polyder	多项式导数
polyfit	多项式拟合
polyval	多项式求值
polyvalm	多项式求值
ppval	分段多项式求值
residue	部分分式展开
roots	多项式求根

表 9　数据分析函数

函数分类	函　数　名	说　明
通用的数据分析函数	max	求最大元素
	min	求最小元素
	mean	求元素平均值或列元素的平均值
	median	求列元素的中值
	std	求元素标准差
	sum	求各列元素的和
	prod	求列元素的积
	hist	直方图
	trapz	梯形法求数值积分
	quad	Simpson 法数值积分
	quad8	Cotes 法数值积分

续表

函数分类	函 数 名	说 明
	cumprod	求列元素的累计积
	cumsum	求列元素的累计和
	cumtrapz	梯形法求累积数值积分
	sort	按升序对元素进行排序
	sortrows	按升序排列矩阵各行
有限差分函数	dff	求差分和近似导数
	gradient	求近似梯度
	del2	求离散 Laplace 算子
相关关系函数	corrcoef	求相关系数
	cov	求协方差矩阵
	subspace	求 2 个子空间之间的夹角

表 10 傅里叶变换函数

函数名	说 明	函数名	说 明
filter	一维离散时间滤波器	ifft2	二维逆快速傅里叶变换
filter2	二维离散时间滤波器	abs	模
conv	卷积	angle	相角
conv2	二维卷积	unwrap	清除相角突变
fft	快速傅里叶变换	fftshift	平移零点到频率中心
fft2	二维快速傅里叶变换	cplxpair	把数字分类为复共轭对
ifft	逆快速傅里叶变换	nexttpow2	最靠近 2 的次幂

表 11 符号工具箱函数

函 数 分 类	函 数 名	说 明
符号表达式运算	sym	建立符号表达式
	syms	建立符号表达式
	numden	提取分子与分母
	symadd	符号加法
	symsub	符号减法
	symmul	符号乘法
	symdiv	符号除法
	sympow	符号表达式的幂运算
	symop	符号运算
	compose	符号表达式的复合函数运算
	finverse	符号表达式的反函数运算
	symsum	求表达式的符号和
	symvar	求符号变量
	numeric	符号表达式转换为数值表达式
	eval	符号表达式转换为数值表达式

续表

函数分类	函数名	说明
符号表达式运算	sym	数值表达式转换为符号表达式
	poly2sym	将等价系数向量转换成它的符号多项式
	sym2poly	将符号多项式转换成它的等价系数向量
	charpoly	特征多项式
符号可变精度运算	digits	设置可变精度
	vpa	可变精度计算
符号表达式的化简	pretty	显示方便查看的符号表达式
	collect	合并符号表达式的同类项
	horner	把一般的符号表达式转换成嵌套形式的符号表达式
	factor	对符号表达式进行因式分解
	expand	对符号表达式进行展开
	simple	对符号表达式进行化简
	simplify	求解符号表达式的最简形式
符号矩阵的运算	transpose	符号矩阵的转置
	determ	符号矩阵的行列式运算
	det	符号矩阵的行列式运算
	inv	符号矩阵求逆运算
	rank	符号矩阵求秩运算
	eig	求符号矩阵的特征值、特征向量
	eigensys	求符号矩阵的特征值、特征向量
	svd	符号矩阵的奇异值运算
	singvals	符号矩阵的奇异值运算
	jordan	符号矩阵的约当标准型运算
符号微积分	limit	符号极限
	diff	符号微分
	int	符号积分
	taylor	泰勒级数展开
符号画图	ezplot	符号函数画图
	fplot	符号函数画图
符号方程的求解	solve	代数方程的求解
	linsolve	齐次线性方程组的求解
	fsolve	非线性方程组的求解
	dsolve	微分方程的求解

表 12　MATLAB 低级文件 I/O 函数

函数名	说明	函数名	说明
fclose	关闭文件	fscanf	从文件中读格式化数据
feof	检查是否到文件尾	fseek	设置文件指针的位置
ferror	查询文件 I/O 的出错信息	ftell	获取文件指针的位置

续表

函数名	说　明	函数名	说　明
fgetl	从文件中读一行，并忽略回车符	fwrite	向一个文件中写入二进制数据
fgets	从文件中读一行，并包括回车符	sprintf	把格式化数据写入字符串
fopen	打开文件	sscanf	从格式化字符串中读入数据
fprintf	把格式化数据写到文件中	tempname	建立临时文件名
fread	从文件中读二进制数据	tempdir	读出临时文件名
frewind	将打开的文件的指针反绕		

表 13　程序设计与文件调试函数

函 数 分 类	函　数　名	说　明
程序设计标识函数	script	命令式 M 文件
	function	函数式 M 文件
	eval	执行字符串
	feval	执行字符串定义的函数
	global	定义全局变量
程序结构控制流	for	预定的次数重复执行的循环语句
	end	与 for,while,if,switch 语句匹配的结束标志语句
	while	以不定的次数执行的循环语句
	if	条件执行语句
	elseif	与 if 匹配使用
	else	与 if 匹配使用
	switch	分支选择语句
	case	与 switch 匹配使用
	otherwise	与 switch 匹配使用
	continue	结束本次循环，判断是否执行下一次循环
	break	终止本次循环，跳出最内层的循环
	return	返回调用函数
	echo	显示执行的 M 文件的每条命令
	error	出错信息显示
	try	对异常进行处理
	catch	与 try 匹配使用
交互式输入	input	提示用户输入
	keyboard	通过键盘输入数据
	pause	等候用户响应
调试命令	dbclear	取消断点
	dbcont	断点后重新运行
	dbdown	工作空间下移
	dbquit	退出调试模式
	dbstack	列表显示堆栈调用
	dbstatus	列表显示所有的断点

续表

函 数 分 类	函 数 名	说 明
调试命令	dbstep	执行一行或多行
	dbstop	设置断点
	dbtype	列表显示带行号的 M 文件
	dbup	工作空间上移

表 14 字符串函数

函 数 名	说 明	函 数 名	说 明
Isstr	判断是否是字符	abs	把字符串变成 ASCII 码值
Blanks	空格符	setstr	把 ASCII 码值变成字符串
deblank	删除空格符	num2str	把数字变成字符串
eval	执行字符串	str2num	把字符串变成数字
isletter	确定字符串中的字符是否为字母	str2mat	把字符串变成文本矩阵
strcmp	字符串比较	hex2num	把十六进制字符串变成 IEEE 浮点数
findstr	在其他字符串中寻找字符串	hex2dec	把十六进制字符串变成十进制数
strrep	用一个字符串代替另一个字符串	dec2hex	把十进制数变成十六进制字符串
upper	把字符串中的字符变成大写形式	sprintf	把带格式的数字变成字符串
lower	把字符串中的字符变成小写形式	sscanf	把字符串变成带格式的数字

表 15 二维图形函数

函 数 分 类	函 数 名	说 明
基本二维绘图函数	plot	二维线形图
	loglog	双对数坐标图
	semilogx	单对数坐标图，*x* 轴为对数坐标，*y* 轴为线性坐标
	semilogy	单对数坐标图，*y* 轴为对数坐标，*x* 轴为线性坐标
	polar	极坐标图
	plotyy	双(y)轴图
坐标轴控制函数	axis	控制坐标轴比例和外观
	zoom	放大或缩小
	grid	网格显示控制
	box	坐标轴外框显示
	hold	控制是否保持当前图形
	axes	在指定位置创建坐标轴
	subplot	创建子图图区
图形标注函数	plotedit	编辑和标注图形工具
	legend	图形图例
	title	图形标题
	xlabel	*x* 轴标签
	ylabel	*y* 轴标签
	textlabel	文本标签
	text	文本注解
	gtext	用鼠标放置文本

续表

函 数 分 类	函 数 名	说 明
硬拷贝和打印函数	print	打印图形或仿真系统，或保存图形到 M 文件
	printopt	打印机默认设置
	orient	设置纸张方向

表 16 三维图形函数

函 数 分 类	函 数 名	说 明
基本三维图形函数	plot3	三维线形图
	mesh	网格图
	surf	表面图
	fill3	填充的三维多边形
颜色控制函数	colormap	颜色察看表
	caxis	颜色按坐标轴比例设置
	shading	颜色阴影模式
	hidden	网格隐藏
	brighten	加亮或变暗色图
	colordef	颜色默认设置
	graymon	灰度监视器图形默认设置
光照控制函数	surfl	给三维阴影表面添加光照
	lighting	光照模式
	material	材料反射模式
	specular	特殊反射
	diffuse	漫反射
	surfnorm	表面法向
相机控制函数	campos	相机位置
	camtarget	相机目标
	camva	相机视角
	camup	相机抬升向量
	camproj	相机投影
透明控制函数	alpha	透明模式
	alphamap	透明察看表
	alim	透明比例
高水平相机控制函数	camlight	创建和设置光线的位置
	lightangle	光线的极坐标位置
视点控制函数	view	指定三维图的视点
	viewmtx	察看转换矩阵
	rotate3d	交互旋转三维图

表 17　特殊图形函数

函 数 分 类	函　数　名	说　　明
特殊二维图形函数	area	面积图
	bar	二维条形图
	barh	二维水平条形图
	comet	彗星图
	errorbar	误差条图
	ezplot	简易函数绘图器
	ezpolar	简易极坐标绘图器
	feather	羽列图
	fill	填充的二维多边形
	fplot	函数图形
	hist	直方图
	pareto	帕累托图
	pie	饼图
	plotmatrix	散点图矩阵
	scatter	散点图
	stem	火柴杆图
	stairs	阶梯图
等值线图函数	contour	等值线图
	contourf	填充的等值线图
	contour3	三维等值线图
	clabel	给等值线图添加标签
	ezcontour	简易等值线图生成器
	ezcontourf	简易填充等值线图生成器
特殊三维图函数	bar3	三维条形图
	bar3h	三维水平条形图
	comet3	三维彗星图
	ezgraph3	简易一般表面绘图器
	ezmesh	简易三维网格绘图器
	ezmeshc	简易三维网格图叠加等值线图生成器
	ezplot3	简易三维参数曲线绘图器
	ezsurf	简易曲面绘图器
	ezsurfc	简易三维曲面叠加等值线图绘图器
	meshc	网格图叠加等值线图
	meshz	窗帘图
	pie3	三维饼图
	ribbon	带形图
	scatter3	三维散点图
	stem3	三维火柴杆图
	surfc	曲面图叠加等值线图
	trisurf	三角形表面图
	trimesh	三角形网格图
	waterfall	瀑布图

续表

函数分类	函数名	说明
体积和向量可视化函数	vissuite	可视组合
	isosurface	等表面提取器
	isonormals	等表面法向
	isocaps	等表面终端帽盖
	isocolors	等表面和阴影颜色
	contourslice	切片面板中的等值线
	slice	体积切片图
	streamline	二维、三维向量数据的流线图
	stream3	三维流线图
	stream2	二维流线图
	quiver3	三维矢量图
	quiver	矢量图
	divergence	向量场的差异
	curl	向量场的卷曲和角速率
	coneplot	三维流锥图
	streamtube	三维流管图
体积和向量可视化函数	streamribbon	流带图
	streamslice	切片面板中叠加流线图
	streamparticles	流沙图
	interpstreamspeed	内插源于速度的流线顶点
	reducevolume	减少体积数据
	volumebounds	为体积数据返回 x，y，z 和颜色限制
	smooths	平滑三维数据
	reducepatch	减少阴影表面的个数
	shrinkfaces	减少阴影表面的大小
图形显示和文件 I/O 函数	image	显示图片
	imagesc	数据比例化并图形显示
	colormap	颜色察看表
	gray	线性灰度色图
	contrast	灰度色图，用于改进图片的对比度
	brighten	加亮或变暗色图
	colorbar	显示色图
	imread	从图形文件中读取图片
	imwrite	从图形文件中写图片
	imfinfo	图形文件的信息
动画和快照函数	capture	抓取当前图形
	moviein	初始化动画框内存
	getframe	获取动画框
	movie	演示记录的动画框
	rotate	旋转指定了原点和方向的对象
	frame2im	将动画框转换为索引图片
	im2frame	将索引图片转换为动画框

续表

函 数 分 类	函 数 名	说 明
与颜色相关的函数	spinmap	旋转色图
	rgbplot	绘色图
	colstyle	用字符串说明颜色和风格
	ind2rgb	将索引图片转换为 RGB 图片
实体模型函数	cylinder	创建柱体
	sphere	创建球体
	ellipsoid	创建椭球体
	patch	创建阴影
	surf2patch	将表面数据转换为阴影数据

表 18 图形窗口的创建和控制函数

函 数 名	说 明
figure	创建图形窗口
gcf	获取当前图形的句柄
clf	清除当前图形
shg	显示图形窗口
close	关闭图形
refresh	刷新图形
openfig	打开已保存图形的新拷贝或显示已存在的拷贝

表 19 插值与拟合

函 数 名	说 明
interp1	一维线性插值
interp2	二维线性插值
interp3	三维线性插值
interft	快速 Fourier 变换得到一维插值
interpn	多数线性插值
spline	三次样条插值
polyfit	最小二乘法拟合

表 20 非线性数值解法

函 数 名	说 明
ode23	二三阶 Runge-Kutta 求解常微分方程
ode23p	二三阶 Runge-Kutta 求解常微分方程并画出结果图
ode45	四五阶 Runge-Kutta 求解常微分方程
fmin	求一元函数的极小值
fmins	求多元函数的极小值
fzero	求一元函数的零点值

表 21　GUI 图形函数

函 数 名	说　明
set	设置对象属性
get	获取对象属性
reset	把对象属性重设为默认值
delete	删除对象
gcf	获取当前图形对象的句柄
gca	获取当前图形窗口内当前坐标轴的句柄
gco	获得当前图形窗口内当前对象的句柄
gcbo	获取当前回调对像的句柄
gcbf	获取当前回调图形的句柄
findobj	获取指定的属性值的对象的句柄
drawnow	绘图
copyobj	图形对象拷贝
isappdata	核对应用程序定义的数据是否存在
getappdata	获取应用程序定义的数据的值
setappdata	设置应用程序定义的数据的值
rmappdata	删除应用程序定义的数据
figure	创建图形对象
axes	创建坐标轴对象
line	创建线条对象
rectangle	创建矩形
light	创建光线
text	创建文本对象
patch	创建补片对象
surface	创建曲面对象
image	创建图像对象
uimenu	创建下拉式菜单对象
uicontextmenu	创建内容式菜单对象
uicontrol	创建控件对象
dialog	创建对话框
uigetfile	创建文件打开对话框
uiputfile	创建文件保存对话框
uisetcolor	创建颜色设置对话框
uisetfont	创建字体设置对话框
pagesetupdlg	创建打印页面设置对话框
pagedlg	创建打印页面设置对话框
printpreview	创建打印预览对话框

续表

函 数 名	说 明
printdlg	创建打印对话框
helpdlg	创建帮助对话框
errordlg	创建出错信息显示对话框
msgbox	创建信息提示对话框
questdlg	创建问题显示对话框
warndlg	创建警告信息显示对话框
inputdlg	创建变量输入对话框
listdlg	创建列表选择对话框